The Discrete Acyclic Digraph Markov Model in Data Mining

Het Discrete, Acyclische, Digraaf Markov Model in Data Mining

(met een samenvatting in het nederlands)

El Model Markovià Discret Digraf Acíclic en la Prospecció de Dades

(amb un resum en català)

El Modelo Markoviano Discreto Digrafo Acíclico en la Prospección de Datos

(con un resumen en español)

Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit Utrecht
op gezag van de Rector Magnificus, Prof. Dr. W.H. Gispen,
ingevolge het besluit van het College voor Promoties
in het openbaar te verdedigen
op maandag 3 juni 2002 des middags te 12:45 uur

door

Juan Roberto Castelo Valdueza

geboren op 10 juli 1972, te Lleida

promotor: Prof. Dr. A.P.J.M. Siebes

Faculteit Wiskunde en Informatica, Universiteit Utrecht

The research reported in this thesis has been partially carried out at CWI, the Dutch national research laboratory for mathematics and computer science, within the theme *Data Mining and Knowledge Discovery*, a subdivision of the research cluster *Information Systems*.

SIKS Dissertation Series No. 2002-4
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN 90-393-3020-4

To the memory of my mother
A la memòria de ma mare

Contents

Acknowledgements

This thesis is a sum of efforts, collaborations, remarks, and many other sort of interactions with a bunch of great people I have met throughout the last four years. Among them, I'm specially grateful to my supervisor, Arno Siebes, who gave me the opportunity of starting this work. I always had his support and his office door open for sharing ideas, doubts, problems, solutions, successes, failures and nice chats.

I'm grateful to those with whom I had the privilege of writing a paper: Paolo Giudici, Tomáš Kočka, Nick Wormald, Ad Feelders and, of course, Arno. And others with whom I haven't (yet) but my work benefited of insightful discussions with them: Steven Gillispie, Michael Perlman and Milan Studený. The research work in this thesis is for me unconceivable without their collaboration.

The European Science Foundation, through the HSSS research programme, provided the necessary funding for a research visit I gave to Paolo. I would like to thank him for his hospitality during those two wonderful weeks in Pavia and the enlightening work we carried out together. The Fields Institute in Toronto also funded a research visit during three weeks that allowed me to attend a seminar on Conditional Independence structures, organized by Milan Studený and Frantisek Matús, and a course on graphical Markov models given by Steen Andersson. Both events have proven a sound source of inspiration.

I want to express my gratitude to the members of my committee, Linda van der Gaag, Richard Gill, Paolo Giudici, Bert Kappen and Michael Perlman, for their careful reading of the manuscript and all their remarks that have substantially improved it.

All this hard work would have been even harder without all the nice people surrounding me first, at CWI in Amsterdam, and later, at Utrecht University. I enjoyed the relaxed and fun working environment created by all of you. I'm also grateful to all the people I met, and specially to the good friends I made, throughout these years in Amsterdam. Resetting the brain on weekends, and many weekly evenings, made me enjoy my life very much and go back to work mentally fresh. Regular visits of my friends from Spain were also fundamental for healing my bit of homesickness.

Finally, I want to thank my family: *gracies pare, germana i cunyat per estar sempre quan us he necessitat.*

Barcelona, April 2002.

Samenvatting

Grafische Markov modellen zijn een krachtig gereedschap voor de beschrijving van ingewikkelde interacties tussen variabelen van een domein. Zij geven een compacte beschrijving van de gezamenlijke kansverdeling van die variabelen. Deze eigenschap heeft geleid tot de meest succesvolle toepassing van grafische Markov modellen: als het hart van probabilistische expertsystemen. De fascinerende theorie achter deze modellen komt uit drie verschillende disciplines, te weten Statistiek, Grafentheorie en Kunstmatige Intelligentie. Deze interdisciplinaire oorsprong heeft geleid tot diep inzicht vanuit verschillende invalshoeken.

Er zijn twee belangrijke manieren om de structuur van grafische Markov modellen te ontdekken. De structuur kan worden beschreven door een domeindeskundige of we *leren de structuur* uit data. Om de structuur te kunnen leren, moeten we kunnen vergelijken hoe goed verschillende modellen de data beschrijven. Dit is eenvoudig voor bijvoorbeeld acyclische gerichte Markov modellen. Toch is structuurleren moeilijk, omdat het aantal mogelijke modellen exponentieel groeit met het aantal variabelen.

De belangrijkste bijdragen van dit proefschrift zijn de volgende. Ten eerste wordt er een nieuwe klasse van grafische Markov modellen geïntroduceerd, te weten de TCI-modellen. Deze modellen kunnen worden weergegeven als gelabelde bomen en zij vormen de doorsnijding van twee bekende klassen. Ten tweede wordt de deelverzamelingsordening van grafische Markov modellen bestudeerd. Uit deze studie worden twee nieuwe leer-algoritmen afgeleid. Eén om heuristisch te zoeken, de ander voor de Markov Chain Monte Carlo methode. Beide algoritmen verbeteren de prestaties van eerdere algoritmen zonder de computationele kosten van het leerproces te vergroten. Ten slotte worden er nieuwe gereedschappen geïntroduceerd om de convergentie van de Markov Chain Monte Carlo methode voor structuurleren te kunnen beoordelen. De resultaten van dit proefschrift worden geïllustreerd aan de hand van zowel synthetische als echte data.

Resum

Els models Markovians gràfics són una eina poderosa per la descripció d'interaccions complexes entre les variables d'un domini. Permeten una descripció compacta de la distribució conjunta de les variables. Aquesta característica els ha dut a la seva aplicació més pròspera com a component principal als sistemes experts probabilistes. La fascinant teoria darrera d'aquest tipus de models sorgeix de tres disciplines diferents: l'Estadística, la Teoria de Grafs i la Inteligència Artificial. Aquest origen interdisciplinari s'ha traduït en una comprensió més profunda d'aquests models desde perspectives diferents.

Existeixen dues formes principals de trobar l'estructura qualitativa dels models Markovians gràfics. O be l'estructura es especificada per un expert del domini, o be s'aplica l'*aprenentatge estructural*, es a dir, l'estructura és recuperada automàticament a partir de les dades. Per dur a terme l'aprenentatge estructural, un ha de comparar en quina mesura els diferents models descriuen bé les dades. Això és senzill, per exemple, per als models Markovians digrafs acíclics. No obstant, l'aprenentatge estructural és encara un problema difícil perquè el nombre possible de models creix exponencialment en el nombre de variables.

Les contribucions principals d'aquesta tesi són les següents. En primer lloc, s'ha introduït una nova classe de models Markovians gràfics, anomenats models TCI. Aquests models poden ser representats per arbres etiquetats i formen la intersecció de dues classes prèviament conegudes. En segon lloc, s'ha estudiat l'ordre d'inclusió dels models Markovians gràfics. A partir d'aquest estudi, es deriven dos algorismes nous d'aprenentatge. Un per cerca heurística i l'altre pel mètode de Markov Chain Monte Carlo. Tots dos algorismes milloren els resultats de solucions prèvies sense comprometre el cost computacional del procés d'aprenentatge. Finalment, s'han introduït diagnòstics nous per l'avaluació de la convergència del mètode de Markov Chain Monte Carlo a l'aprenentatge estructural. Els resultats d'aquesta tesi han estat il.lustrats utilitzant dades sintètiques i reals.

Resumen

Los modelos Markovianos gráficos son una herramienta poderosa para la descripción de interacciones complejas entre las variables de un dominio. Permiten una descripción compacta de la distribución conjunta de las variables. Esta característica los ha llevado a su aplicación más próspera como componente principal en los sistemas expertos probabilistas. La fascinante teoría detrás de este tipo de modelos surge de tres disciplinas diferentes: la Estadística, la Teoría de Grafos i la Inteligencia Artificial. Este origen interdisciplinario se ha traducido en una comprensión más profunda de estos modelos desde perspectivas diferentes.

Existen dos formas principales de encontrar la estructura cualitativa de los modelos Markovianos gráficos. O bien la estructura es especificada por un experto del dominio, o bien se aplica el *aprendizaje estructural*, es decir, la estructura es recuperada automáticamente a partir de los datos. Para llevar a cabo el aprendizaje estructural, uno ha de comparar en qué medida los diferentes modelos describen bien los datos. Esto es sencillo para, por ejemplo, los modelos Markovianos digrafos acíclicos. Sin embargo, el aprendizaje estructural es aún un problema difícil porque el número posible de modelos crece exponencialmente en el número de variables.

Las contribuciones principales de esta tesis son las siguientes. En primer lugar, se ha introducido una nueva clase de modelos Markovianos gráficos, nombrados modelos TCI. Estos modelos pueden ser representados por árboles etiquetados y forman la intersección de dos clases préviamente conocidas. En segundo lugar, se ha estudiado el order de inclusión de los modelos Markovianos gráficos. A partir de este estudio, se derivan dos algoritmos nuevos de aprendizaje. Uno para búsqueda heurística y el otro para el método de Markov Chain Monte Carlo. Ambos algoritmos mejoran los resultados de soluciones previas sin comprometer el coste computacional del proceso de aprendizaje. Finalmente, se han introducido diagnósticos nuevos para la evaluación de la convergencia del método de Markov Chain Monte Carlo en el aprendizaje estructural. Los resultados de esta tesis han sido ilustrados utilizando datos sintéticos y reales.

Chapter 1

Introduction

1.1 The Data Mining perspective on data analysis

Computation is, nowadays, one of the most influential aspects in most fields of scientific research. Until the moment in which computation has obtained such relevance, the problem of understanding data and its generating mechanisms was, almost exclusively, tackled in the field of statistics.

Computer science is the scientific area where computation is its main concern. Within the scope of computer science, areas like artificial intelligence (AI), machine learning and computational learning theory try to formalize the notion of *knowledge* and find systematic ways to acquire and process it. More concretely, *knowledge acquisition* is the motivation that leads research efforts towards the problem of *learning from data* with the ultimate objective of constructing a machine that *learns* from its environment.

Machine learning developed within the more general framework of AI and, at the beginning, the research community was geared to find ways of modeling how humans learn. In the last 20 years, however, most of the research strive for more specific goals as the development of algorithms that enable computers to, e.g., recognize patterns or make predictions from data. This led to an overlap with some of the work that was developed within the area of statistics, with important differences in particular aspects. These differences rely basically in the mathematical language used and a substantial stress on computational aspects in the works from machine learning.

Most researchers in machine learning work in the scope of computer science, which usually implies that they also have an specific interest in the evolution of computer technology. The growth of the amount of data stored in computers, and of computing power as well, takes place at a extremely high rate throughout the years. In the last ten years, the machine learning community, aware of such circumstance, has identified new challenges and potential benefits of analyzing these massive datasets with our current, and always increasing, computing power.

This realization has pushed the creation of an specific area of applied research and development called *data mining*, with its own meetings, conferences and journals. Part of the research developed in data mining tries to scale up conventional statistical methods and machine learning algorithms to work on massive datasets. Nonetheless, the area of data mining has also contributed novel methods for the general problem of learning from data (see (Smyth, 2001) for a summary of them).

Almost independently, but also under the umbrella of computer science, the area of database research has contributed substantially to the rapid growth of data mining. In a way this is not surprising, since databases is probably the computer science area most sensitive to the extraordinary evolution of computing power and storage. This makes database products almost the unique choice in software technology to store and access massive datasets.

The traditional problems tackled within the database area have been those related exclusively with the efficient storage and access of data on a computer. However, in the last years the storage of massive datasets has led to an increasing interest in finding valuable nuggets of information in the stored data. This interest, has driven the evolution of database query languages from simple query capabilities

> "Select all clients from Amsterdam."

through relational on-line analytic processing (ROLAP)

> "Select clients from Amsterdam between 20 and 30 years old that spend more than 100 euros a month on beer in the first trimester of the year."

to automatic search of interesting patterns in the data

> "What is the profile of the clients who spend more money on beer."

This last step in the evolution of database systems blurs the boundary between data access and data analysis and has made the database community to become very interested and, at the same time, active, in the data mining area. The involvement of databases has had two main positive side-effects: to enable, technologically, data analysis in massive datasets and to make data mining enthusiastically embraced by industry.

Thus clearly, data mining is an interdisciplinary area, boosted primarily by research works in machine learning and databases, but unconceivable without a substantial portion of the statistical language and methodology.

Data mining gives a perspective on data analysis where one not only tries to understand data and its generating mechanisms, but also aims at providing a comprehensive summary of what is interesting. This actually implies that interpretability of the results is of primary interest as well as aspects of the visualization of both the results and the data. Therefore, from a statistical viewpoint, one may see data mining as a computer automated exploratory data analysis (Friedman, 1997).

The data mining perspective also emphasizes in the computational side of the solutions provided. They should scale up in order to be applied to the data stored nowadays. As we mentioned, the size of this data may be huge, but for many problems the dimension, or number of variables, may be an even more challenging aspect.

1.2 The unifying framework of Graphical Markov Models

Graphical Markov models ($\equiv$ GMMs) allow us to describe structural properties of a family of probability distributions using graphs. Most of the evolution of this type of models has taken place in the last twenty years within the fields of statistics and AI.

They are the core component in modern expert systems, which is the reason why the AI community has been involved in their development at large.

The structural properties that GMMs handle are conditional independence restrictions. A restriction, or statement, of conditional independence is an assertion of the form:

> "Cholesterol intake is conditionally independent of the chance of a heart attack given the cholesterol blood level."

This assertion implies that if we know the cholesterol blood level no further information concerning what we eat (cholesterol intake) can enhance our knowledge about the chance of a heart attack. Thus, conditional independence is a formalization of the notion of *irrelevance*.

The usual way in which one describes the mechanism that generates some data set is by enumerating relationships among observed variables, conjecturing their interplay with other plausible hidden (latent) variables and establishing some independent measurement errors. All these aspects can be formally described using conditional independence (Whittaker, 1990). Therefore, the fact that GMMs are a powerful formalism to summarize and handle conditional independence restrictions, makes them an unifying framework for data analysis.

1.3 Graphical Markov Models in Data Mining

GMMs are a good example of successful cross-fertilization of ideas from statistics and computation-related areas. Therefore, one may expect to see them in the toolkit usually deployed in data mining. In fact, they have been considered for such a role already since the first years of largest growth of the area (Glymour, 1995; Spirtes and Meek, 1995; Buntine, 1996a; Heckerman, 1996).

From the many features of GMMs that makes them attractive in data mining (see for instance (Fayyad et al., 1996; Buntine, 1996a)), we highlight three:

- They allow to represent a broad spectrum of problems by handling complex high-order interactions among variables.
- Their graphical counterpart eases the description of such complex interactions and enables the use of graph-theoretic related results from discrete mathematics and computer science.
- Their capability to summarize a probability distribution make them suitable to be used in cooperation with other, different, types of models.

Among the different classes of GMMs that exist, the class of acyclic digraph Markov models, or DAG models, is specially appealing for data mining purposes. The DAG model, and its subclasses, afford an efficient computation of the likelihood of the model for a given dataset, which allows the comparison of many models in a very short time. This feature enables the construction of algorithms that work in a systematic way to learn this class of models from data. Of course, learning these models from data will never replace the domain expert, but it can be a valuable source of information.

However, if we take a look at the data mining literature, or the algorithms that current data mining software offer, we will not find the DAG Markov model as often as we would expect. Other types of models like association rules, decision trees or clustering are much more popular in data mining. From the possible reasons, we find the following three as the more fundamental ones:

- **Interpretability**. Although the graphical representation is quite intuitive, their interpretation becomes difficult when more than a dozen variables are considered.
- **Scalability**. The current learning algorithms for DAG models have serious computational problems with more than 15 variables. This has been usually handled by establishing a causal order among the variables, that substantially reduces the set of competing models, but that is nearly impossible to elicitate in practice. Those approaches that do not rely on a causal order have a serious computational burden to be able to learn a good model.
- **Interoperability**. Their potential use in cooperation with other types of models has been poorly exploited.

1.4 Research Objectives

The research carried out in this thesis tackles the previously mentioned issues of interpretability, scalability and interoperability. In concrete, interpretability is enhanced in two directions. One by identifying a particular subclass of GMMs that affords an easier representation of highly connected GMMs. Another by exploiting the use of learning diagnostics that enrich the output of the learning process. Scalability is enabled by providing algorithms that can work without the need for any causal order among the variables without compromising the computational performance of the learning process. Finally, interoperability is extended by studying the relationship of GMMs and association rules, and by providing an algorithm for learning association rules using GMMs.

1.5 Outline of this thesis

In the next chapter we will examine the DAG Markov model, and its subclasses. In particular, we will discuss a new subclass called the TCI Markov model. In chapter 3 we will survey standard methods for learning the structure of these models from data and two new learning algorithms will be presented. In chapter 4 we will talk again about data mining, and in concrete we will examine one of the most popular types of formalism used in this area, association rules, for which a new algorithm will be devised. In chapter 5, some applications with real data will show the potential use of GMMs in data mining, and finally, in chapter 6 we will summarize the main points of this thesis and we will sketch further lines of research.

Chapter 2

Graphical Markov Models

2.1 Introduction

In this chapter we describe Graphical Markov models (GMMs). We focus on a particular class, the discrete acyclic digraph Markov model. For the interested reader, the books by Pearl (1988); Whittaker (1990); Cox and Wermuth (1996) and Lauritzen (1996) present broader and more thorough overviews of GMMs. Similar to those books, we use the term *model* to specify an arbitrary family of distributions that satisfy a set of conditional independence restrictions.

In this thesis we deal with multinomial data only although many of the concepts and some of the results are independent of this assumption. The discrete acyclic digraph Markov model will be referred to as the DAG Markov model, or simply the DAG model. Some authors use instead the more accurate term ADG, for *acyclic directed graph*, but we have chosen for the more popular, DAG.

In the context of probabilistic expert systems, and e.g. in the book by Pearl (1988), the terms *Bayesian Network*, *Belief Network* or *Bayesian Belief Network* are used instead of DAG Markov model. All terms are perfectly valid but we want to treat the different classes of GMMs described in this thesis uniformly.

The rest of this chapter is organized as follows. First we will formally introduce the concept of conditional independence, and we will sketch how it is related to GMMs. In the following five sections we will describe five different classes of GMMs that belong to the class of DAG Markov models, yet studied. In particular, section 2.7 introduces a new type of GMM determined by labeled trees. Finally, in section 2.8 we will discuss the relationship among the classes and their sizes and in the last section we will highlight the most relevant points of this chapter. Part of the contents in sections 2.7 and 2.8 have been published in (Castelo and Siebes, 2001; Castelo and Wormald, 2001).

2.2 Conditional Independence

Conditional independence, CI hereafter, is a fundamental notion in the analysis of interactions among multiple factors. The intuition behind it is that a dependence relationship between two variables may *vanish* when a third variable is considered in relation with the former two.

For instance, it is widely accepted that there is a strong relationship between the level of cholesterol in the food we eat and the occurrence of a heart attack. A higher cholesterol intake in our eating habits increases the risk of suffering a heart attack. However, if we take a blood sample and analyze the effective cholesterol blood level, we can determine more accurately how large the risk of a heart attack is. In fact, at the moment we consider *cholesterol blood level* (*CBL*), the *cholesterol intake* (*CIN*) becomes irrelevant to the occurrence of a *heart attack* (*HA*). We may say that the dependency between *CIN* and *HA vanishes* when we also consider *CBL*, or more formally, *CIN* is *conditionally independent* of *HA given CBL* and denote it by

$$CIN \perp\!\!\!\perp HA \mid CBL$$

The notion of conditional independence is actually defined for finite sets of variables. The set on which we *condition*, is called the *conditioning set*. When the conditioning set is empty, the conditional independency is a marginal independency which corresponds to the more commonly known notion of independence between two factors. For example our typewriting skills are *independent* of the chance that our favorite national football team will win the next football world cup.

The formal definition of CI is based on the concept of random variables and a joint probability distribution over a set of random variables. We will not introduce these concepts here, but the reader may find them described in depth in any standard introductory textbook on statistics.

We will denote a random variable by an upper case letter, e.g. X, and a set or a vector of random variables with a boldface upper case letter, e.g. $\mathbf{X}$. Whether $\mathbf{X}$ is a set or a vector will be clear from the context. Often, we will be interested in indexing random variables to denote their membership to some vector or set, using a number or a lowercase letter, as in $X_1, \ldots, X_i, \ldots, X_n$.

Similarly, we will index vectors and sets of random variables to distinguish them, as e.g. $\mathbf{X}_A, \mathbf{X}_B, \ldots, \mathbf{X}_Z$. For notational convenience, we sometimes use A as a shorthand for $\mathbf{X}_A$ or i as a shorthand for X_i.

A set of categorical random variables $\mathbf{X}_V$ may form a multivariate distribution that belongs to some family of distributions P defined on a product space $\mathcal{X} = \times(\mathcal{X}_i | i \in V)$. Any family of probability distributions considered in this chapter, and in the rest of this thesis, is defined on $\mathcal{X}$. A particular instantiation of a single random variable will be denoted with a lowercase letter, as $X_i = x$ where $x \in \mathcal{X}_i$. An instantiation of a set of random variables will be denoted with a boldface lowercase letter, as $\mathbf{X}_A = \mathbf{x}$ where $\mathbf{x} \in \mathcal{X}_A$.

Dawid (1979) formalized the concept of CI and introduced the ternary operator $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}[P]$, to denote that $\mathbf{X}$ is *conditionally independent* of $\mathbf{Y}$ *given* $\mathbf{Z}$ *under* P. Where P refers to some family of probability distributions over some larger set of random variables $\mathbf{V}$ that contains $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$. The formal definition is as follows.

Definition 2.1. *Conditional Independence (CI)*
Let $\mathbf{V}$ *be a finite set of categorical random variables where each* $X \in \mathbf{V}$ *has a finite domain. Let* $p(\mathbf{V})$ *be a joint probability distribution over the random variables in* $\mathbf{V}$*, that belongs to some family of probability distributions* P*. Let* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *be any subsets of* $\mathbf{V}$ *such that* $\mathbf{X}, \mathbf{Y}$ *are disjoint*

and non-empty. We say that $\mathbf{X}$ *is* conditionally independent *of* $\mathbf{Y}$ *given* $\mathbf{Z}$*, noted* $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ $[P]$*, if for all configurations* $\mathbf{x}, \mathbf{y}, \mathbf{z}$ *of the variables in* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *satisfying* $p(\mathbf{Z} = \mathbf{z}) > 0$*, it holds that*

$$p(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) = p(\mathbf{X} = \mathbf{x} | \mathbf{Z} = \mathbf{z}).$$

Very often P will be fixed and we will drop it from the notation writing only $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$. The previous definition may be seen as a factorization criterion that tells that the conditional probability of $\mathbf{X}$ given $\mathbf{Y}$ and $\mathbf{Z}$, is in fact a function of $\mathbf{Z}$ alone. Because this may be interpreted as a constraint or restriction, one often uses the term *CI restriction* to refer to a CI statement. When $\mathbf{Z}$ is trivial, $\mathbf{X}$ and $\mathbf{Y}$ are *marginally independent*. This means that

$$p(\mathbf{X}, \mathbf{Y}) = p(\mathbf{X}) p(\mathbf{Y}). \tag{2.1}$$

Where, for clarity, we have not written the assignments of values explicitly. Analogously, when the conditioning set $\mathbf{Z}$ is not trivial and $p(\mathbf{Z}) > 0$, the joint probability factorizes as follows,

$$p(\mathbf{X}, \mathbf{Y} | \mathbf{Z}) = p(\mathbf{X} | \mathbf{Z}) p(\mathbf{Y} | \mathbf{Z}), \tag{2.2}$$

if and only if $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$. Expression (2.2) is a rewrite of (2.1) where the unconditional probabilities are replaced by conditional ones.

The constraint of positive conditioning probabilities it is necessary in the formulation above as otherwise the conditionals would not be unique. An alternative interpretation, without imposing positivity in the conditionals, is that $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ if versions of the conditional distributions $p(\mathbf{X} | \mathbf{Z})$ and $p(\mathbf{Y} | \mathbf{Z})$ exist such that their product is a version of the *law*[1] of $p(\mathbf{X}, \mathbf{Y} | \mathbf{Z})$. More concretely, $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ if and only if one can reproduce the joint law of $p(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ by:

1. sampling from the marginal law of $p(\mathbf{Z})$.
2. sampling $\mathbf{X}$ from $p(\mathbf{X} | \mathbf{Z} = \mathbf{z})$.
3. sampling $\mathbf{Y}$ from $p(\mathbf{Y} | \mathbf{Z} = \mathbf{z})$.

Pearl (1988) provided a sound and complete axiomatization of the concept of CI. It consists of five logical conditions that constrain the relation of CI between any given set of random variables (that forms a multivariate distribution of some family P). The first four logical conditions are as follows:

- *Symmetry*:

$$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \,|\, \mathbf{Z} \iff \mathbf{Y} \perp\!\!\!\perp \mathbf{X} \,|\, \mathbf{Z} \tag{CI1}$$

[1] In the statistical context, a *law* is a probability distribution of the probabilities of another probability distribution, i.e. a second-order probability distribution.

- *Decomposition*:

$$\mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W}) \,|\, \mathbf{Z} \Longrightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \,|\, \mathbf{Z} \;\wedge\; \mathbf{X} \perp\!\!\!\perp \mathbf{W} \,|\, \mathbf{Z} \tag{CI2}$$

- *Weak Union*:

$$\mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W}) \,|\, \mathbf{Z} \Longrightarrow \mathbf{X} \perp\!\!\!\perp \mathbf{Y} \,|\, (\mathbf{Z} \cup \mathbf{W}) \tag{CI3}$$

- *Contraction*:

$$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \,|\, \mathbf{Z} \;\wedge\; \mathbf{X} \perp\!\!\!\perp \mathbf{W} \,|\, (\mathbf{Z} \cup \mathbf{Y}) \Longrightarrow \mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W}) \,|\, \mathbf{Z} \tag{CI4}$$

Any model which satisfies those conditions is called a *semi-graphoid*. Further, if P is strictly positive, then the model is called a *graphoid*, and the following fifth condition holds:

- *Intersection*:

$$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \,|\, (\mathbf{Z} \cup \mathbf{W}) \;\wedge\; \mathbf{X} \perp\!\!\!\perp \mathbf{W} \,|\, (\mathbf{Z} \cup \mathbf{Y}) \Longrightarrow \mathbf{X} \perp\!\!\!\perp (\mathbf{Y} \cup \mathbf{W}) \,|\, \mathbf{Z} \tag{CI5}$$

In general, we will find that for many real world domains, the random variables considered are constrained logically in such a way that the assumption that P is strictly positive may be unrealistic. Therefore, most of the time one is interested in developing results that rely on properties that only depend on the semi-graphoid axioms.

Recall from the introduction of this chapter that we use the term model to specify an arbitrary family of distributions that satisfy a set of CI restrictions. In GMMs, the set of CI restrictions satisfied by the model are encoded in a graph or, as we shall see later in section 2.6, any other equivalent algebraic structure. The criterion used to code, or read off, a CI restriction from a graph is formally defined as a *Markov property*.

Every type of graph has its own set of Markov properties. For any given type of graph, its Markov properties will be related such that they may be equivalent or one may follow from another. Given two Markov properties MP and MP$'$, such that MP $\Rightarrow$ MP$'$, one says that MP is *sharper* than MP$'$. The definition of a particular Markov property will be used to provide the definition of a particular class of GMM, in the following way.

Definition 2.2. *Graphical Markov Model*
Let G be a graph of some type. Let MP *be the sharpest Markov property for G. The family of probability distributions that satisfy the Markov property* MP *relative to G is called the* Graphical Markov Model *determined by G, and it will be noted as* $\mathbf{M}(G)$.

One also says, that the family of probability distributions $\mathbf{M}(G)$ *are Markov over G*, or *are G-Markovian*.

In each of the following sections that describes a type of GMM we will introduce the necessary notation and terminology which is mainly borrowed from Lauritzen (1996) and Andersson et al. (1995).

2.3 The Decomposable Graphical Markov Model

2.3.1 Background concepts

A graph is a pair $G = (V, E)$ where V is the set of vertices and E is the set of edges. In the present context of GMMs, the set of vertices acts as an index set for some collection of random variables $\mathbf{X}_V = \{X_1, \ldots, X_n\}$ that form a multivariate distribution of some family P.

The set of edges E is a subset of the set of ordered pairs $V \times V$ that does not contain loops, i.e. $(x, y) \in E \Rightarrow x \neq y$, nor multiple edges. Given two vertices a, b, we say that they form an *undirected edge* if and only if $(a, b) \in E$ and $(b, a) \in E$. An undirected edge is represented graphically by a solid line joining the two vertices involved, e.g. *a–b*. When all the edges in E are undirected we will say that the graph G is an *undirected graph* (UG).

When two vertices are joined by an undirected edge, these two vertices are called *adjacent*. Given a vertex $v \in V$, the *boundary* of v is $bd(v) = \{u \in V \mid (u, v) \in E\}$. The *closure* of a vertex v is $cl(v) = bd(v) \cup \{v\}$.

A *subgraph* $G_S = (S, E_S)$ is given by a subset $S \subseteq V$ and the induced edge set $E_S = E \cap (S \times S)$. It will be often said that G_S is an *induced subgraph* of G. An undirected graph $G = (V, E)$ is said to be *complete* if and only if every pair of vertices is adjacent. A *subset* of vertices is *complete* if it induces a complete subgraph. A *clique* is a maximal complete subgraph. Note that in the standard terminology of graph theory, our definition of clique is called *maximal clique* and the term clique is used to denote a complete subgraph.

An *undirected path* between two vertices a and b is a sequence $a = v_0, \ldots, v_n = b$ of distinct vertices such that $n > 0$, $(v_{i-1}, v_i) \in E$ and $(v_i, v_{i-1}) \in E$ for $i = 1, \ldots, n$. An *undirected cycle* is an undirected path that begins and ends in the same vertex, i.e. $a = b$.

Given three subsets of vertices $A, B, S \subset V$, it is said that S *separates* A from B in an undirected graph if and only if every undirected path between vertices in A and B intersects S. For instance, in both graphs of Figure 2.1, vertices 3 and 4 *separate* vertex 5 from vertices 1 and 2, while vertex 4 alone *does not* separate vertex 2 from vertex 3.

An *undirected chordal graph*, or *chordal graph* for short, is an undirected graph with no chordless[2] undirected cycles on more than three vertices. Figure 2.1 shows examples of chordal and non-chordal undirected graphs. Chordal graphs are also known as *decomposable*, *triangulated* or *rigid circuit* graphs, and their properties have been exploited in many other areas of research as, e.g., in databases (Beeri et al., 1981).

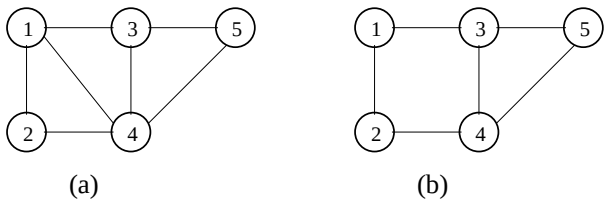


Figure 2.1: Examples of undirected chordal (a) and non-chordal (b) graphs.

As we shall see now, they determine the decomposable graphical Markov model,

[2] A chord in a graph is an edge joining two vertices already connected by a path.

DEC Markov model hereafter. The Markov properties that are used in this type of graph are the ones used for the larger class of UGs.

2.3.2 Markov properties and definition

There are three Markov properties for undirected graphs, and therefore, for undirected chordal graphs.

Definition 2.3. *Undirected pairwise Markov property (UPMP)*
Let $G = (V, E)$ be an undirected graph, a probability distribution P is said to satisfy the undirected pairwise Markov property *(UPMP) if, for any pair $u, v \in V$ of non-adjacent vertices, P satisfies*

$$u \perp\!\!\!\perp v \mid V \backslash \{u, v\} \, [P].$$

The UPMP means that two non-adjacent vertices $u, v \in V$ are conditionally independent given the rest of the vertices.

Definition 2.4. *Undirected local Markov property (ULMP)*
Let $G = (V, E)$ be an undirected graph, a probability distribution P is said to satisfy the undirected local Markov property *(ULMP) if, for any vertex $v \in V$, P satisfies*

$$v \perp\!\!\!\perp V \backslash cl(v) \mid bd(v) \, [P].$$

The ULMP means that a vertex v is conditionally independent of the rest of the variables without its boundary, given its boundary.

Definition 2.5. *Undirected global Markov property (UGMP)*
Let $G = (V, E)$ be an undirected graph, a probability distribution P is said to satisfy the undirected global Markov property *(UGMP) if, for any triple (A, B, S) of disjoint subsets of V such that S separates A from B in G and A, B are non-empty, P satisfies*

$$A \perp\!\!\!\perp B \mid S \, [P].$$

The UGMP means that two non-empty subsets of vertices A, B are conditionally independent given a third subset of vertices S, if and only if S separates A from B.

In (Whittaker, 1990) and (Lauritzen et al., 1990) there is a more thorough discussion of these Markov properties, in particular about the fact that they are related in the following way:

$$\text{UGMP} \Rightarrow \text{ULMP} \Rightarrow \text{UPMP}.$$

This implies that the UGMP is the sharpest possible rule to read off CI restrictions from an undirected (chordal) graph. If P is strictly positive, then the properties are equivalent (Pearl and Paz, 1987). Finally, we can introduce the definition of decomposable graphical Markov model.

Definition 2.6. *DEC Markov model*
Let G be an undirected chordal graph. The set $\mathbf{U}(G)$ of all probability distributions that satisfy the UGMP relative to G is called the decomposable graphical Markov model*, or DEC Markov model, determined by G.*

2.4 The Acyclic Directed Graphical Markov Model

2.4.1 Background concepts

An edge is *directed*, or also called an *arc*, if and only if $(a,b) \in E \Rightarrow (b,a) \notin E$. A directed edge or arc between two vertices a and b, such that $(a,b) \in E$, will be represented graphically by an arrow pointing from a towards b, i.e. $a \rightarrow b$. A graph $G = (V,E)$ is said to be *directed* if all edges in E are *directed edges*. For clarity, sometimes we will write $a \rightarrow b \in E$ to denote that $(a,b) \in E$. As for undirected graphs, two vertices joined by a directed edge in either direction will be called *adjacent*.

For a directed edge $a \rightarrow b$ we distinguish between the two joined vertices by specifying that a is the *parent* of b, and that b is the *child* of a. Those parent vertices that have a common child, will be considered as the *parent set* of this child vertex, and it will be noted as $pa(v)$ for any given child vertex v.

A *path* between two vertices a and b is a sequence $a = v_0, \ldots, v_n = b$ of distinct vertices such that $n > 0$ and either $(v_{i-1}, v_i) \in E$ or $(v_i, v_{i-1}) \in E$ for $i = 1, \ldots, n$. A *cycle* is a path where $a = b$. In a directed graph, a *directed path* is formed by directed edges and is a *direction-preserving* path. This means that every directed edge in the path points towards the same direction. A given vertex a is called the *ancestor* of b if there is a directed path from a to b. A *directed cycle* is a directed path where the first vertex coincides with the last one. An *acyclic directed graph*, or DAG, is a directed graph without directed cycles. The *skeleton* of a DAG is the undirected graph obtained by transforming the set of directed edges into a set of undirected ones that preserves the same adjacencies.

For any given vertex v, one may consider the set of those vertices that are ancestors of v, which will be called the *ancestor set* of v, and noted $an(v)$. Analogously, a vertex b is called the *descendant* of a if there is a directed path from a to b, i.e. a is ancestor of b. The vertices at the end of every directed path that starts at vertex a will form the *descendant* set of a, noted $de(a)$. Given a vertex v the *non-descendant* set of v is defined as $nd(v) = V \backslash \{de(v) \cup \{v\}\}$.

For a given DAG $G = (V,E)$ and a subset of vertices $A \subseteq V$, A is said to be *ancestral* if and only if for every vertex $v \in A$, $an(v) \subseteq A$. Further, for an arbitrary subset of vertices $A \subseteq V$, there is always some larger subset that contains A and it is ancestral. The smallest of the ancestral sets containing A will be called the *smallest ancestral set* of A and noted $An(A)$. To avoid confusion, let's remark the difference between $an(v)$ and $An(A)$. The former one refers to the set of vertices that are ancestors of the vertex v, while the latter refers to the smallest subset $An(A) \subseteq V$ that contains A and is ancestral in G.

An important concept regarding DAGs in this context is the concept of *immorality*. An *immorality* is formed by two non-adjacent vertices with a common child, e.g. $a \rightarrow b \leftarrow c$. In the terminology of Cox and Wermuth (1996), an immorality is known as a *sink-oriented V-configuration*, where a *V-configuration* is defined as a triplet of vertices (a,b,c) such that two of them are adjacent to the third one but they are not adjacent to each other. In the case $a \rightarrow b \leftarrow c$, the vertex b is referred as the *collision* vertex. The terminology of Cox and Wermuth (1996) allows to define further configurations on three vertices as the *source-oriented V-configuration*, e.g. $a \leftarrow b \rightarrow c$, and the *transition-oriented V-configuration*, e.g. $a \rightarrow b \rightarrow c$.

A DAG that has no immoralities is said to be *moral*. In the context of graph theory,

moral DAGs are known as *subtree acyclic digraphs* and were characterized by Harary et al. (1992). A DAG that is not moral can be *moralized* by *marrying* those non-adjacent parents that induce an immorality, i.e. joining them with an undirected edge, and dropping directions on the rest of the edges in G. The moralized version of a directed graph G will be noted as G^m.

2.4.2 Markov properties and definition

There are three Markov properties for DAGs.

Definition 2.7. *Directed pairwise Markov property (DPMP)*
Let $G = (V, E)$ be a DAG, a probability distribution P is said to satisfy the directed pairwise Markov property *(DGMP) if, for any pair $u, v \in V$ of non-adjacent vertices such that $v \in nd(u)$, P satisfies*

$$u \perp\!\!\!\perp v \mid nd(u) \backslash \{v\} [P].$$

The DPMP means that two-nonadjacent vertices u and v, such that v is *non-descendant* of u, are conditionally independent given the non-descendant vertices of u without v.

Definition 2.8. *Directed local Markov property (DLMP)*
Let $G = (V, E)$ be a DAG, a probability distribution P is said to satisfy the directed pairwise Markov property *(DLMP) if, for any vertex $v \in V$, P satisfies*

$$v \perp\!\!\!\perp \{nd(v) \backslash pa(v)\} \mid pa(v) [P].$$

The DLMP means that a vertex is conditionally independent of its non-descendants, without its parents, given its parents.

Definition 2.9. *Directed global Markov property (DGMP)*
Let $G = (V, E)$ be a DAG, a probability distribution P is said to satisfy the directed global Markov property *(DGMP) if, for any triple (A, B, S) of disjoint subsets of V, where A, B are non-empty, such that S separates A from B in the moralized version of the subgraph induced by the vertices in $An(A \cup B \cup S)$, i.e. in $G^m_{An(A \cup B \cup S)}$, P satisfies*

$$A \perp\!\!\!\perp B \mid S [P].$$

The DGMP means that two non-empty subsets of vertices A, B are conditionally independent given a third subset S if and only if S separates A and B in the moralized subgraph induced by the smallest ancestral set of $A \cup B \cup S$.

An alternative way of reading conditional independencies in a DAG is using the *d-separation* criterion of Pearl and Verma (1987), which we review now. Given two vertices $u, v \in V$ and a subset $S \subseteq V$ where $u, v \notin S$, one says that a path between u and v is *active* with respect to S if

1. every non-collision vertex in the path is not in S, and
2. every collision vertex in the path is in S or has a descendant in S.

When a subset S creates an active path between two vertices u and v, then u and v *cannot* be conditionally independent given S in G. When a path between two vertices u, v *is not* active with respect to S, one says that the path is *blocked* by S. Given these notions of active and blocked path, the d-separation criterion is defined as follows.

Definition 2.10. *d-separation*
Let $G = (V, E)$ be a DAG. For any triple (A, B, S) of disjoint subsets of V, where A, B are non-empty, A and B are d-separated by S if every path between the vertices in A and B is blocked by S.

Lauritzen et al. (1990) prove that the *d-separation* criterion encodes in a DAG exactly the same CI restrictions as the DGMP. They also prove that the local and global Markov properties coincide:

$$\text{DGMP} \Leftrightarrow \text{DLMP}.$$

We may see more clearly the intuition behind the DGMP in Figure 2.2. Given the DAG G on the lefthand side of the figure, let's try to find out whether the DGMP holds for $1 \perp\!\!\!\perp 6 \,|\, 3$ in G. The smallest ancestral set of 1,6 and 3 is $An(\{1, 3, 6\}) = \{1, 2, 3, 5, 6\}$.

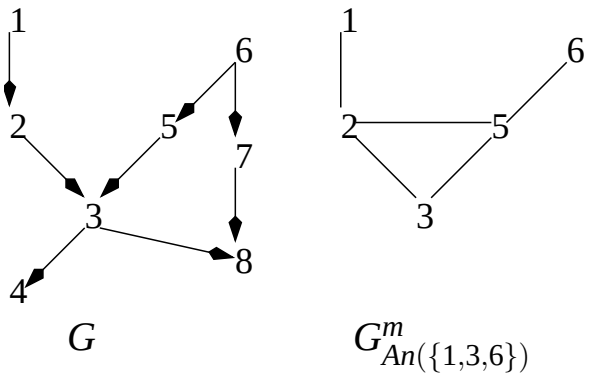


Figure 2.2: A DAG on the left, and on the right its moralized version over the smallest ancestral set of $\{1, 3, 6\}$.

The moralized version of the graph induced by this subset of vertices is on the right-hand side of the figure. Since the subgraph over $An(\{1, 3, 6\})$ contains the immorality $2 \rightarrow 3 \leftarrow 5$, 2 and 5 become adjacent at the moment we moralize the graph, creating therefore, a new path between 1 and 6, which does not intersect the conditioning set $\{3\}$, thus $1 \perp\!\!\!\perp 6 \,|\, 3$ *does not* hold. In the terminology of Pearl and Verma (1987) we would say that the conditioning set $\{3\}$ makes the path 1,2,3,5,6 active in G. In order to find vertex 1 separated from vertex 6, one should add 2 or 5 to the conditioning set, or remove 3.
The definition of DAG Markov model is finally as follows.

Definition 2.11. *DAG Markov model*
Let G be a DAG. The set $\mathbf{D}(G)$ of all probability distributions that satisfy the DGMP relative to G is called the acyclic directed graphical Markov model*, or DAG Markov model, determined by G.*

2.4.3 Markov equivalence

Two DAG Markov models $\mathbf{D}(G)$ and $\mathbf{D}(G')$ determined by two different DAGs G and G' may represent the same restrictions of conditional independence as we can see from the following example for three random variables $\mathbf{X}_V = \{X_a, X_b, X_c\}$. Let's consider a DAG model determined by the following DAG:

$$G = \{a \to b \to c\}.$$

According to the DGMP, this DAG encodes the CI restriction $X_a \perp\!\!\!\perp X_c \mid X_b$. But, another different DAG over the same vertex set that would encode this CI restriction would be

$$G' = \{a \leftarrow b \to c\},$$

and yet another one

$$G'' = \{a \leftarrow b \leftarrow c\}.$$

This actually implies that the sets of discrete probability distributions that are Markov over G, G' and G'', are actually the same, thus $\mathbf{D}(G) = \mathbf{D}(G') = \mathbf{D}(G'')$. When this happens, we say that the DAG models $\mathbf{D}(G)$, $\mathbf{D}(G')$ and $\mathbf{D}(G'')$ are *Markov equivalent*, or simply, *equivalent*. Note, however, that they are not equivalent to the immorality

$$a \to b \leftarrow c,$$

since this DAG induces the marginal independency between X_a and X_c, i.e. $X_a \perp\!\!\!\perp X_c \mid \emptyset$.

Since an equivalence relation is reflexive, symmetric and transitive, the search space of DAG Markov models is organized in equivalence classes. Markov equivalence for DAG models was characterized by Verma and Pearl (1990) in the following theorem.

Theorem 2.1. *(Verma and Pearl, 1990)*
Two DAGs are equivalent if and only if they have the same skeletons and the same immoralities.

From this theorem it is possible to devise an operational criterion to determine which edges in a DAG are *compelled* and which are *reversible*. The criterion is that a directed edge that remains in the same direction for all graphs in an equivalence class is compelled. The others are reversible.

In fact, the canonical representation of an equivalence class of DAG Markov models is an acyclic partially directed graph, where the edge set may contain directed and undirected edges with certain characterizing properties. A directed edge represents a compelled edge, and an undirected edge represents a reversible one. This representation is known in the literature under different terms as, *completed pattern* (Verma and Pearl, 1990), *pattern* (Spirtes et al., 1993), *completed PDAG* (Chickering, 1995), or *essential graph* (Andersson et al., 1997a).

Yet, the previous definition of the canonical representation is broader than the type of graph that we are actually talking about. It corresponds to the definition of *chain graphs* that determine the larger class of *chain graph Markov models* (Frydenberg, 1990a; Andersson et al., 2001) which will not be treated in this thesis. Throughout this thesis, we will adopt the term *essential graph* which was introduced by Andersson et al. (1997a) and which is described in detail in the following section.

The previous operational criterion to obtain the canonical element of one of the Markov equivalence classes of DAG models is, in fact, impractical. Luckily several authors (Chickering, 1995; Andersson et al., 1997a) have provided polynomial time algorithms to create the corresponding essential graph of any given DAG.

2.5 The Essential Graph Markov Model

We have seen in the previous section that DAG Markov models are organized in equivalence classes. In this section we will examine the class of GMMs determined by a particular representation of these equivalence classes. The set of CI models one can represent in this class are obviously the same as for DAG models, but the graphical representation is different.

As we shall see in the next chapter, learning procedures for GMMs use the type of graph that determine the GMM to define the search space. Therefore, alternative graphical representations for a class of GMMs allow the use of different search spaces for the same learning problem. Often, the learning task will perform differently in each search space, thus it becomes important to know all possible representations to devise later better learning algorithms.

Let's remark that from a causal point of view all DAG Markov models are different as every arc denotes a cause-effect relationship. However, from a non-causal perspective, the interpretation of a single DAG becomes somewhat deceptive when we realize that there are other equivalent DAGs. This problem is solved by using a canonical representation of the equivalence class, the essential graph.

In order to describe essential graphs in detail we need to generalize the notions of directed path and directed cycle in the following way. A *semi-directed* path is either a directed path or a path which has one or more directed edges pointing towards the same direction. A *semi-directed cycle* is a semi-directed path that starts and ends in the same vertex.

An essential graph (EG), as well as an UG or a DAG, is a specific case of the broader class of *chain graphs*. A chain graph (CG) is an acyclic partially directed graph (PDAG) whose edges set may contain both directed and undirected edges such that there are no semi-directed cycles. In Figure 2.3 we see examples of CGs as (a) and (b), and examples that are not CGs as (c) and (d).

Recall that a connected graph is such that has a path between every pair of vertices. A graph G of any given type is formed by a non-empty set of *connected components*. A connected component of a graph G, is an induced connected subgraph G' such that there is no other connected subgraph induced from G that contains G'.

Analogously, a CG G is formed by a non-empty set of *chain components* $\mathcal{T}(G)$. The set of chain components of a CG corresponds to the set of connected components left after the removal of all directed edges in the CG.

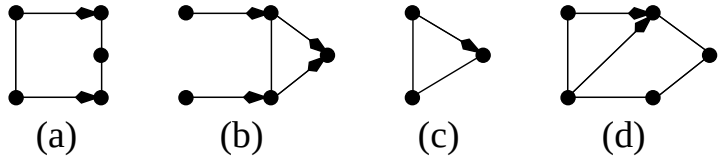


Figure 2.3: The graphs in (a) and (b) are chain graphs while the graphs in (c) and (d) are not.

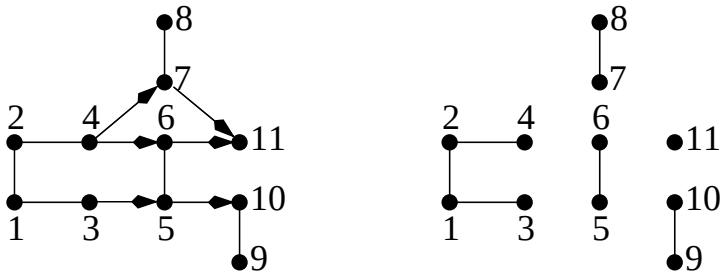


Figure 2.4: A chain graph on the left and its set of chain components on the right.

In Figure 2.4 we see a CG on the left and on the right its set of chain components $\mathcal{T}(G) = \{\{1,2,3,4\},\{5,6\},\{7,8\},\{9,10\},\{11\}\}$. As we may observe from this example, the set of chain components $\mathcal{T}(G)$ forms a *chain* where the directed edges establish a partial order among the chain components, hence the name *chain graph*. In the GMM that is determined by a CG, the random variables within a chain component are assumed to occur on equal footing. Conversely, the directed edges in the CG represent some stepwise mechanism generating the data, among the chain components. We will not examine further this class of GMMs; the interested reader may consult (Frydenberg, 1990a; Andersson et al., 2001).

An EG is a CG with additional characterizing properties. These properties have been investigated by several authors (Spirtes and Meek, 1995; Chickering, 1995; Andersson et al., 1997a). We will review only the properties as they were formalized by Andersson et al. (1997a) in the following theorem:

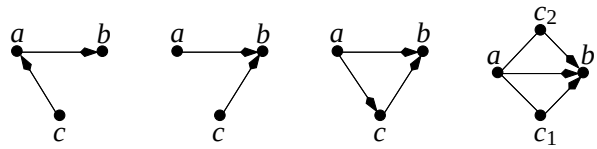
Theorem 2.2. *Essential Graph (EG) (Andersson et al., 1997a, Theorem 4.1)*
A graph $G = (V, E)$ is the EG for some DAG D with vertex set V if and only if G satisfies the following four conditions:

(1) G is a chain graph.

(2) For each chain component $\tau \in \mathcal{T}(G)$, the undirected graph G_τ is chordal.

(3) G has no induced subgraph of the form $a \rightarrow b$–c.

(4) Each arrow $a \rightarrow b$ in G is strongly protected*, that is, it occurs in at least one of the following configurations as an induced subgraph of G:*



The notion behind the previous characterizing properties is that there are directed edges which remain in the same direction throughout all the DAGs that form an equivalence class. One says that these directed edges are *essential* (Andersson et al., 1997a). This characterization allows to devise a polynomial time algorithm to convert a DAG into an EG and viceversa (Spirtes and Meek, 1995; Chickering, 1995; Andersson et al., 1997a). Note, for instance, that the CG on the left of Figure 2.4 is not an EG because the subset of vertices $\{4, 5, 6\}$ induces a subgraph that violates condition (3).

2.5.1 Markov properties and definition

Recall the concepts of boundary $bd(v) = \{u \in V \,|\, (u, v) \in E\}$ and closure $cl(v) = bd(v) \cup \{v\}$ introduced in the context of undirected graphs. Note that in the context of DAGs and CGs $pa(v) \subseteq bd(v)$, while $bd(v) \cap de(v) = \emptyset$, where $de(v)$ corresponds to a more general definition given below.

In a more general sense than for DAGs: the ancestors $an(v)$ of a given vertex v is the set of vertices from which there is a semi-directed path that ends at v; the descendants $de(v)$ of a given vertex v is the set of those vertices at the end of every semi-directed path that starts at v.

Note that these broader definitions of descendants $de(v)$ and ancestors $an(v)$ implicitly modify those of non-descendants $nd(v)$ and smallest ancestral set $An(A)$.

Further, we need two new concepts. Let $G = (V, E)$ be a CG with set of chain components $\mathcal{T}(G)$. Let $\tau(v) \in \mathcal{T}(G)$ be the set of vertices that forms the chain component in G that contains v. A subset $A \subseteq V$ is *anterior* in G iff A is ancestral in G and $v \in A \Rightarrow \tau(v) \subseteq A$. The subset $At(A) \subseteq V$ denotes the smallest anterior set in G that contains A.

Markov properties for CGs were originally introduced by Lauritzen and Wermuth (1989) and Frydenberg (1990a), where the corresponding global Markov property formalized in those works is commonly known as the *Lauritzen-Wermuth-Frydenberg* (LWF) Markov property. Later Andersson et al. (1996, 2001) provided alternative Markov properties for CGs, where the corresponding global Markov property is known as the *Alternative Markov Property* (AMP[3]) Markov property. In the case of EGs, the LWF and AMP Markov properties are equivalent (Andersson et al., 2001, Theorem 4.3). Therefore we will review here only the definition for one of them, the LWF, and we refer the interested reader to the mentioned sources.

Definition 2.12. *Chain graph pairwise Markov property (CGPMP)*
Let $G = (V, E)$ be a CG, a probability distribution P is said to satisfy the chain graph pairwise Markov property *(CGPMP) if, for any pair $u, v \in V$ of non-adjacent vertices such that $v \in nd(u)$, P satisfies*

$$u \perp\!\!\!\perp v \,|\, nd(u) \backslash \{v\} [P].$$

[3]Note that it admits also the spelling of its author's names *Andersson-Madigan-Perlman*.

The notion of the CGPMP is exactly the same as for the DPMP but in the more general sense introduced by the definition of $de(v)$ for CGs.

Definition 2.13. *Chain graph local Markov property (CGLMP)*
Let $G = (V, E)$ be a CG, a probability distribution P is said to satisfy the chain graph local Markov property *(CGLMP) if, for any vertex $v \in V$, P satisfies*

$$v \perp\!\!\!\perp nd(v) \backslash cl(v) \mid bd(v)[P].$$

The CGLMP is analogous to the DLMP but here the role of the parents $pa(v)$ is taken over by the boundary $bd(v)$.

Definition 2.14. *Chain graph global Markov property (CGGMP)*
Let $G = (V, E)$ be a CG, a probability distribution P is said to satisfy the chain graph global Markov property *(CGGMP) if, for any triple (A, B, S) of disjoint subsets of V, where A, B are non-empty, such that S separates A from B in the moralized version of the subgraph induced by the vertices in $At(A \cup B \cup S)$, i.e. in $G^m_{At(A \cup B \cup S)}$, P satisfies*

$$A \perp\!\!\!\perp B \mid S[P].$$

Again the CGGMP is analogous to the DGMP but the role of the smallest ancestral set $An(\cdot)$ is taken over by the smallest anterior set $At(\cdot)$. Frydenberg (1990a) shows that the CGGMP is the sharpest possible rule to read off CI restrictions from a CG:

$$\text{CGGMP} \Rightarrow \text{CGLMP} \Rightarrow \text{CGPMP}.$$

Finally, the definition of EG Markov model is as follows.

Definition 2.15. *EG Markov model*
Let G be an EG. The set $\mathbf{E}(G)$ of all probability distributions that satisfy the CGGMP relative to G is called the essential graph Markov model*, or EG Markov model, determined by G.*

2.6 The Lattice Conditional Independence Markov Model

The *Lattice Conditional Independence* (LCI) Markov model was introduced by Andersson and Perlman (1993) in the context of the analysis of non-nested multivariate missing data patterns and non-nested dependent linear regression models. Later, Andersson et al. (1997a, Theorem 4.1) showed that the class of LCI models coincides with the class of *Transitive Acyclic Directed Graph* (TDAG) Markov models. The overview of this class of GMMs is necessary in order to fully understand the material in the next section. We begin by reviewing the most important background concepts and why finite distributive lattices (that determine LCI models) are the same mathematical objects as transitive acyclic digraphs (that determine TDAG models).

2.6.1 Background concepts

A DAG is *transitive* $\equiv$ TDAG if for every vertex v, $pa(v) = an(v)$. Recall that a subset of vertices $A \subset V$ is ancestral iff for every vertex $v \in A$, $an(v) \subseteq A$. Since the union and intersection of ancestral sets is again ancestral, all the different ancestral sets contained in a DAG $G = (V, E)$ form a ring of subsets of V, which is denoted by $\mathcal{A}(G)$.

A partially ordered set ($\equiv$ poset) $(S, \leq)$ is a set S equipped with an order relation[4] $\leq$. If the poset is *totally ordered*, i.e. $\forall a, b \in S \;\; a \leq b \;\; or \;\; b \leq a$, then it is a *chain*. A chain C in a poset S is called *maximal* iff, for any chain $D \in S$, $C \subseteq D$ implies that $C = D$. Let S be a poset and let $x, y \in S$. We say x is *covered* by y, and write $x \prec y$ if $x < y$ and $x \leq z < y \Rightarrow z = x$.

Grätzer (1978, pg. 10) shows that this covering relation determines the partial ordering in a given poset in the following way. Let S be a finite poset. Then $a \leq b$ iff $a = b$ or there exists a finite sequence of elements $x_0, \ldots, x_{n-1}$, such that $x_0 = a, x_{n-1} = b$, and $x_i \prec x_{i+1}$, for $0 \leq i < n - 1$.

A poset $(S, \leq)$ has an associated undirected graph (V, E) in which $(x, y) \in E$ if $x \prec y$ (y covers x). This associated undirected graph is called the *covering graph* of the poset S. A *Hasse* diagram of a poset S is a representation of the covering graph of S in the plane such that if $x < y$, then x is below y in the plane.

Analogous to the concept of an ancestral set in a DAG, one may define an *ancestral poset*. Let $(S, \leq)$ be a poset, a subset (which is again a poset) $A \subseteq S$ is ancestral in $(S, \leq)$ iff $\forall a \in A$ it follows that $b \in S$ and $b < a \Rightarrow b \in A$.

Given a poset S, a subset $H \subseteq S$ and an element $a \in S$, it is said that a is an *upper bound* (*lower bound*) of H iff for every $h \in H$, $h \leq a$ ($h \geq a$). An upper bound (lower bound) a of H is the *least upper bound* (*greatest lower bound*) of H or *supremum* (*infimum*) of H iff, for any upper bound (lower bound) b of H, we have $a \leq b$ ($a \geq b$), and denote it by $a = \sup H$ ($a = \inf H$).

It is possible to define a *lattice* in different ways. We will introduce here just one of them, as follows. A poset L is a *lattice* iff $\sup H$ and $\inf H$ exist in L for any finite nonvoid subset H of L. Grätzer (1978) shows that the concept of a lattice as a poset is equivalent to the concept of a lattice as an algebra $\mathcal{K} \equiv \mathcal{K}(\wedge, \vee)$, where $\wedge$ and $\vee$ are binary operations on pairs of elements $a, b \in \mathcal{K}$, corresponding to $\inf\{a, b\}$ and $\sup\{a, b\}$ respectively. The operations $\wedge, \vee$ are idempotent, commutative and associative, and satisfy two absorption identities. It has been already mentioned that LCI models are determined by finite distributive lattices. Birkhoff (Grätzer, 1978, pg. 62) characterized finite distributive lattices as those isomorphic to a ring of sets.

A finite distributive lattice $\mathcal{K}$ has a unique irredundant representation in terms of a finite poset $(J(\mathcal{K}), \leq)$, where $J(\mathcal{K}) \subseteq \mathcal{K}$ is the subset of *join-irreducible* elements (see Grätzer, 1978, pg. 62). This poset often is substantially smaller than $\mathcal{K}$, and its elements are defined in the following way

$$J(\mathcal{K}) = \{a \in \mathcal{K} \,|\, a \neq \emptyset, a = b \vee c \Rightarrow a = b \;\; or \;\; a = c\}.$$

In this context, the lattice $\mathcal{K}$ can be constructed by unions ($\vee$) and intersections ($\wedge$) of the elements of the set of join-irreducible elements $J(\mathcal{K})$. Davey and Priestley (1990) characterize a join-irreducible element of a finite distributive lattice as an element which

[4] Reflexive, antisymmetric and transitive.

has exactly one lower cover, i.e. it covers exactly one other element. Note that the partial order $\leq$ of the finite poset $(J(\mathcal{K}), \leq)$ is inherited from $\mathcal{K}$: $a \leq b$ iff $a \wedge b = a$.

It is possible to establish a one to one correspondence between finite posets and TDAGs. Given the finite poset $(S, \leq)$ we can build a TDAG $G = (S, E^<)$, where

$$E^< = \{(a, b) \in S \times S | a < b\}.$$

Given a TDAG $G = (V, E)$, for every pair of vertices $a, b \in V$, $a \in an(b) \Leftrightarrow a < b$. Note that all ancestral subsets of a poset $(S, \leq)$ form a ring $\mathcal{A}((S, \leq))$ which is identical to the ancestral ring $\mathcal{A}((S, E^<))$ of the TDAG $G = (S, E^<)$ defined before.

This correspondence between TDAGs and finite distributive lattices is used by Andersson et al. (1997c) to prove that TDAG models and LCI models coincide.

2.6.2 Markov properties and definition

There is only one specific Markov property for LCI Markov models.

Definition 2.16. *Lattice conditional independence Markov property (LCIMP)*
Let $G = (V, E)$ be a TDAG, a probability distribution P is said to satisfy the lattice conditional independence Markov property *(LCIMP) if, for every pair of ancestral subsets $A, B \in \mathcal{A}(G)$, P satisfies*

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

Andersson et al. (1997c) defined the LCIMP for the ancestral sets of a DAG. In this more general case, they prove:

Theorem 2.3. *Andersson et al. (1997c, Theorem 2.2,p. 32)*
Let G be a DAG. For any probability distribution P, DGMP $\Rightarrow$ LCIMP.

The previous theorem is sharpened for TDAG Markov models as follows.

Theorem 2.4. *Andersson et al. (1997c, Theorem 3.1,p. 33)*
Let G be a TDAG. For any probability distribution P,

$$\text{DGMP} \Leftrightarrow \text{DLMP} \Leftrightarrow \text{LCIMP}.$$

The definition of an LCI Markov model is the following.

Definition 2.17. *LCI Markov model*
Let G be a TDAG. The set $\mathbf{L}(G)$ of all probability distributions that satisfy the LCIMP relative to G is called the lattice conditional independence Markov model*, or* LCI Markov model*, determined by G.*

The fact that there is a one to one correspondence between TDAGs and finite distributive lattices, and the latter are isomorphic to a ring of sets, leads to an alternative reformulation of LCI Markov model in terms of posets. Consider a ring $\mathcal{K}$ of subsets of V, such that for every pair of subsets $L, M \in \mathcal{K}$, a probability distribution P satisfies

$$L \perp\!\!\!\perp M | L \cap M,$$

as in the LCIMP. The subsets L, M refer to subsets of random variables $\mathbf{X}_L, \mathbf{X}_M \subseteq \mathbf{X}_V$ that take values from a larger product space $\mathcal{X} = \times(\mathcal{X}_i | i \in V)$ and $L, M \subseteq V$. Over this product space, a family of probability distributions P satisfies the LCIMP, and gives rise to an LCI model $\mathbf{L}(\mathcal{K})$ that, as the notation suggests, is determined by a ring $\mathcal{K}$. For more details about LCI models determined by rings of subsets, the reader may consult Andersson and Perlman (1993); Andersson et al. (1997c).

In Figure 2.5a we may see an empty DAG, which represents the fully restricted DAG model, on the left, and its representation by a Hasse diagram on its right as the fully restricted LCI model. In Figure 2.5b we may see a complete DAG, which represents the unrestricted or saturated DAG model, and its representation by a Hasse diagram on its right as the unrestricted LCI model. Let's note that for the LCI model on Figure 2.5a, $J(\mathcal{K}_a) = \{1, 2, 3\}$ and for the LCI model on Figure 2.5b, $J(\mathcal{K}_b) = \{1, 12, 123\}$. Note that both DAGs are also TDAGs.

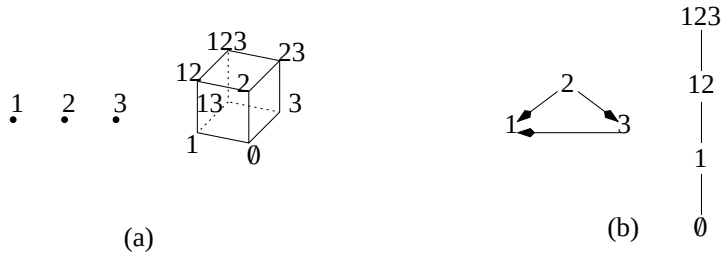


Figure 2.5: Comparison between DAG (TDAG) models and LCI models.

While for the graphical Markov model in Figure 2.5a the restrictions are characterized by all three vertices being marginally independent: $1 \perp\!\!\!\perp 2 \perp\!\!\!\perp 3$, the set of restrictions of the model in Figure 2.5b is empty. In order to read conditional independencies from the Hasse diagram, we have to take into account that any two elements from this diagram are conditionally independent given their intersection (LCIMP). For instance, two trivial cases are those from Figure 2.5. Note that the LCI model in 2.5b is not unique, as we also have other equivalent complete DAGs (always from a non-causal perspective).

In the next figure, we may see two more sophisticated models. The one on Figure 2.6a corresponds to the immorality that induces the two non-adjacent vertices marginally independent, and the one on Figure 2.6b makes the two non-adjacent vertices conditionally independent given the middle one. On the LCI model of Figure 2.6a, $J(\mathcal{K}_a) = \{1, 3, 123\}$ and on LCI model of Figure 2.6b, $J(\mathcal{K}_b) = \{2, 12, 23\}$ (recall that an element belongs to $J(\mathcal{K})$ iff it covers only one other element).

2.7 The Tree Conditional Independence Markov Model

It follows from the known relationships among the different classes of GMMs that the intersection of the classes of DEC Markov models and TDAG Markov models (or LCI Markov models) is non-empty. In this section we show that the GMMs in the intersection can be characterized as labeled trees. This fact leads to the definition of a specific Markov property for labeled trees and therefore to the introduction of labeled trees as part of the

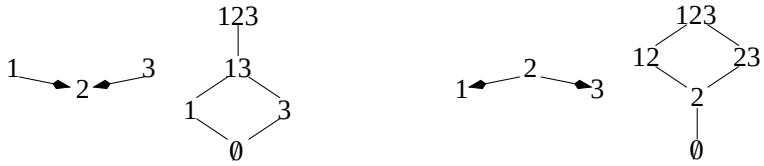


Figure 2.6: On the left hand side of (a) and (b), two DAGs (TDAGs) representing $1 \perp\!\!\!\perp 3|\emptyset$ and $1 \perp\!\!\!\perp 3|2$ respectively, and on the right hand side of (a) and (b) their corresponding Hasse diagrams.

family of GMMs.

With the exception of LCI models, GMMs are usually determined by a graph (e.g. UG, DAG and DEC models) which is interpreted in terms of *separation*. Separation (see beginning of this chapter) is a graphical notion that allows one to split the vertex set of the graph into a triplet that maps to the ternary relationship of conditional independence. As we have seen in the previous section, LCI models are not graphical, thus they do not have a graph-separation interpretation, but another one that manipulates directly the lattice that determines the model. However, we have also seen in the previous section that the LCI representation is equivalent to a graphical representation via TDAGs that does have a graph-separation interpretation.

In this section we consider another special class of GMMs, namely DEC∩LCI. Andersson et al. (1995) proved that DEC∩LCI $\neq \emptyset$ and for this particular class, we introduce an alternative graphical representation which is not interpreted in graph-separation terms. The new representation is more economical, in the sense that it affords an easier interpretation of the model of conditional independence, in contrast to its equivalent graphical counterpart in terms of either TDAGs or chordal graphs, or its equivalent non-graphical counterpart in terms of finite distributive lattices.

This new representation is based on a characterization of moral TDAGs as labeled trees, which will be presented first. Afterwards, a Markov property for labeled trees will be introduced. Finally, the relationship between this new Markov property and the rest of the existing Markov properties is investigated. From this study follows the new formalization of the graphical Markov models in DEC∩LCI. Because of the relation between trees and models for conditional independence, we will refer to DEC∩LCI models as *tree conditional independence* ≡ TCI models.

The direct consequence of such a formalization is that it provides a different way to read the structural information (≡ the conditional independencies) contained in the model, by using the new associated Markov property.

2.7.1 Background concepts

Most of the background concepts we need here were already introduced in the previous section for LCI models. We need however, some notions related to trees and a few remarks regarding TDAGs and posets.

A *tree* is a connected undirected graph without undirected cycles. In this case there is

always a unique path between any two different vertices. A *rooted tree* is a tree in which a hierarchy among the vertices is created. One of the vertices of a rooted tree is the *root* and it is placed at the bottom of the hierarchy. The *leaves* of a rooted tree are those vertices connected to just one other vertex; they are placed at the top of the hierarchy. Under this convention we will say that the root is *below* the leaves, and the leaves are *above* the root. Note that this convention, inspired in the concept of tree used in natural sciences, is different from the one used in computer science where the root is at the top, and the leaves are at the bottom, thus upside-down.

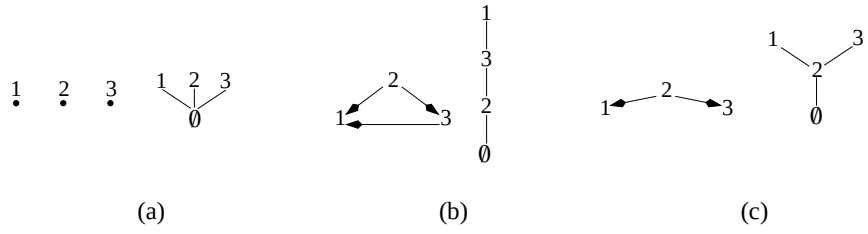
Given a tree $T = (V, E)$ and a vertex $u \in V$, a *subtree* rooted at u, denoted as T_u, is the pair $T_u = (U, E_U)$, where the vertex set $U \subseteq V$ contains all vertices involved in every path from u to the leaves above, and the edge set $E_U = E \cap (U \times U)$.

A TDAG is *moral* if it contains no immoralities. For any moral TDAG and every vertex v, the induced subgraph $\{v\} \cup pa(v)$ $(\{v\} \cup an(v))$ is complete.

An *envelope*[5] E of a poset $(S, \leq)$ is a subset $E \subseteq S$ such that for every $s \in S$, there exists $e \in E$ such that $s \leq e$. A *minimal envelope*[5] E^* of a poset $(S, \leq)$ is an envelope of $(S, \leq)$ such that there is no subset $E \subseteq E^*$ that is an envelope of $(S, \leq)$ too.

2.7.2 Moral TDAGs as labeled trees

In this section we build an isomorphism between moral TDAGs and labeled trees, which will allow us to represent any given moral TDAG with a unique corresponding labeled tree. In order to get a first intuition of such mapping, we may see in Figure 2.7 an example of three simple moral TDAGs with their corresponding labeled tree representation.



(a) (b) (c)

Figure 2.7: From left to right, trees constructed from an empty moral TDAG, a complete moral TDAG and a moral TDAG where one vertex renders the two other non-adjacent vertices conditionally independent.

In this section we do not yet discuss Markov models, only purely graph-theoretic issues. The results in this section will be used later to introduce the new class of graphical Markov models based on labeled trees.

Lemma 2.1. *Let $G = (V, E)$ be a moral TDAG corresponding to the finite distributive lattice $\mathcal{K}$, which has set of join-irreducible elements $J(\mathcal{K})$. Let $\mathcal{A}(G)$ be the ring of ancestral subsets of V in G, which is identical to the ring of ancestral posets of $J(\mathcal{K})$, $\mathcal{A}((J(\mathcal{K}), \leq))$. The set of*

[5]Note that *envelope* and *minimal envelope* are concepts analogous to those of *cover* and *minimal cover* as defined in Hearne and Wagner (1973).

join-irreducible elements $J(\mathcal{K})$ is the collection of maximal chains

$$J(\mathcal{K}) = (H_i \,|\, H_i \subseteq \mathcal{A}(G) \ \wedge \ H_i \text{ maximal chain}). \tag{2.3}$$

Proof. A poset $(S, \leq)$ is a *tree poset* iff for $x, y, z \in (S, \leq)$, $x, y < z \Rightarrow x < y$ or $y < x$.

Recall from the previous section that the relation between $G = (V, E)$ and its corresponding poset $(J(\mathcal{K}), \leq)$ is such that for every $a, b \in V$, $a \in an(b) \Leftrightarrow a < b$ in $(J(\mathcal{K}), \leq)$. The fact that G is a moral TDAG implies that for $a, b, c \in V$ such that $a, b \in an(c)$, either $a \in an(b)$ or $b \in an(a)$. Therefore the poset $(J(\mathcal{K}), \leq)$ is a tree poset.

Consider a decomposition of a poset $(S, \leq)$ as a collection of smaller posets $H_1, H_2, \ldots, H_k$, $H_i \subseteq (S, \leq)$, $1 \leq i \leq k$, as follows. For every element x_i of the minimal envelope of $(S, \leq)$, create $H_i = \{y \in S : y < x_i \text{ in } (S, \leq)\} \cup \{x_i\}$.

Applying the previous decomposition to the poset $(J(\mathcal{K}), \leq)$ we will obtain a collection of k posets $H_i = \{y_1, \ldots, y_q, x_i\}$, $1 \leq i \leq k$. The tree poset condition of $(J(\mathcal{K}), \leq)$ and the fact that $y_j < x_i$, $1 \leq j \leq q$, implies that each H_i is a maximal chain. □

Let $\mathcal{K}$ be a finite distributive lattice isomorphic to some moral TDAG G. Then the set of join-irreducible elements $J(\mathcal{K})$ is of the form (2.3) and forms a poset $(J(\mathcal{K}), \leq)$. Let $\mathcal{L}$ denote the class of such finite distributive lattices. Consider a correspondence μ between the set of such posets $P(\mathcal{L}) = \{(J(\mathcal{K}), \leq) | \mathcal{K} \in \mathcal{L}\}$ and the set of labeled trees $T(\mathcal{L}) = \{(J(\mathcal{K}) \cup \{\emptyset\}, E^{\prec}) | \mathcal{K} \in \mathcal{L}\}$, defined as follows:

$$\begin{array}{rccc} \mu: & P(\mathcal{L}) & \longleftrightarrow & T(\mathcal{L}) \\ & (J(\mathcal{K}), \leq) & \longleftrightarrow & (J(\mathcal{K}) \cup \{\emptyset\}, E^{\prec}) \end{array} \tag{2.4}$$

where for every labeled tree $t(\mathcal{K}) \equiv (J(\mathcal{K}) \cup \{\emptyset\}, E^{\prec}) \in T(\mathcal{L})$ the vertex set is formed by the elements in $J(\mathcal{K})$ plus an extra vertex labeled $\emptyset$, which acts as the root. Note that there is a one to one correspondence between $J(\mathcal{K})$ and the set of vertices from the equivalent moral TDAG. The set of edges of the labeled tree $t(\mathcal{K})$ is defined as follows:

$$E^{\prec} = \{(a, b) \in J(\mathcal{K}) \times J(\mathcal{K}) | a \prec b\} \cup \{(\emptyset, a) \in \{\emptyset\} \times J(\mathcal{K}) | \ \nexists\, b \in J(\mathcal{K}) \ b \prec a\},$$

where $\prec$ is the covering relation on the poset of join-irreducible elements $J(\mathcal{K})$. From the next three propositions it will follow that the correspondence μ is a bijection between moral TDAGs and labeled trees.

Proposition 2.1. *The correspondence μ is well-defined, i.e., let $\mathcal{K}$ be a finite distributive lattice that coincides with some moral TDAG, the graph $\mu(\mathcal{K})$ is a labeled tree.*

Proof. From Lemma 2.1 we can decompose $J(\mathcal{K})$ into its maximal chains H_C. Every $\mu(H_C)$ is a path in $\mu(\mathcal{K})$ from the root to a leaf and viceversa. For any two such chains H_{C_1} and H_{C_2}, $\mu(H_{C_1}) \cap \mu(H_{C_2})$ is a unique path from the root to a vertex. It follows directly that $\mu(\mathcal{K})$ has no cycles and therefore is a labeled tree. □

Proposition 2.2. *The correspondence (2.4) is injective.*

Proof. Let $\mathcal{K}_1, \mathcal{K}_2$ be two finite distributive lattices that coincide with two moral TDAGs. If $\mu(\mathcal{K}_1) = \mu(\mathcal{K}_2)$, then for every path $h \in \mu(\mathcal{K}_1)$, for which we can create a maximal chain H, there exists a path $d \in \mu(\mathcal{K}_2)$, for which we can create a maximal chain D, such that $h = d$ and $H = D$. Since then the collection of all H will be the same as the collection of all D, it follows that $J(\mathcal{K}_1) = J(\mathcal{K}_2)$, and the correspondence (2.4) is injective. □

Proposition 2.3. *The correspondence (2.4) is surjective.*

Proof. For convenience, consider labeled trees T where the root is labeled as $\emptyset$ and the rest of the vertices using natural numbers $\{1, \dots, n\}$. From the fact that T is a labeled tree, there is always a unique path from the root to each of its leaves. For every path p of the tree T, such that $p = \{\emptyset, x_1, \dots, x_n\}$, take out the root $\emptyset$, and from the rest of the path $\{x_1, \dots, x_n\}$ construct a chain H_C such that $H_C = \{x_1, \{x_1, x_2\}, \dots, \{x_1, \dots, x_n\}\}$. From Lemma 2.1 we know that the collection of these chains H_C produces a set of join-irreducible elements $J(\mathcal{K})$ corresponding to a lattice $\mathcal{K}$ that coincides with a moral TDAG. □

Finally, we can establish the following result.

Theorem 2.5. *The class of moral TDAGs is isomorphic to the class of labeled trees.*

Proof. This follows directly from the fact that the mapping (2.4) is a bijection between moral TDAGs and labeled trees. □

2.7.3 Moral TDAG models as tree conditional independence ≡ TCI models

This section introduces a new class of graphical Markov models called TCI models, based on labeled trees. Moreover, it is shown that TCI coincides with the class of DEC∩LCI graphical Markov models.

A graphical Markov model member of the class DEC∩LCI is determined by a TDAG with no immoralities (Andersson et al., 1995). By Theorem 2.5 we can represent a moral TDAG using a labeled tree.

One of the features that distinguishes tree structures from other types of graph, used in the context of graphical Markov models, is that they are connected. In this sense, they are quite similar to the Hasse diagrams used to represent lattices in LCI models. Thus, we may observe in Figure 2.7a how a complete disconnected graph turns into a connected structure, a labeled tree, by using this new, artificially introduced, vertex labeled $\emptyset$.

The intuition behind the root node $\emptyset$ will become clear from the Markov property for labeled trees. To define this formally, we need two new concepts regarding labeled trees and the following proposition.

Proposition 2.4. *Let $T = (V \cup \{\emptyset\}, E)$ be a tree rooted at $\emptyset$. Given any two vertices $u, v \in V$ there is always at least one common vertex (the root $\emptyset$) in the two unique paths that lead from u and v to the root $\emptyset$.*

Proof. It follows directly from the fact that every vertex in a tree is reachable from the root by a unique path. □

Given the existence of at least one common element for every two paths from two given vertices to the root, consider the next two definitions.

Definition 2.18. *Meet*
Let $T = (V \cup \{\emptyset\}, E)$ be a tree rooted at $\emptyset$. Let T_u, T_w be two subtrees of T, rooted at vertices u and w respectively, such that neither is a subtree of the other. The meet *is the first common vertex in the two unique paths from u, w to the root $\emptyset$. It will be noted as $\varphi_{u,w}$.*

Definition 2.19. *Meet path*
Let $T = (V \cup \{\emptyset\}, E)$ be a tree rooted at $\emptyset$. Let $u, w \in V$ be two vertices inducing subtrees T_u, T_w, such that neither is a subtree of the other. Let $\varphi_{u,w}$ be their meet. The meet path *is the set of vertices that forms the common path from the meet to the root, and denoted as $mp(\varphi_{u,w}) = \{\varphi_{u,w}, \ldots, \emptyset\}$.*

As we can see, *meet* and *meet path* are intuitive concepts that follow naturally from the definition of a tree. It is straightforward to identify the *meet* in a tree for two given vertices, even if this tree is large. Finally, the new Markov property can be introduced.

Definition 2.20. *Tree conditional independence Markov property (TCIMP)*
Let $T = (V \cup \{\emptyset\}, E)$ be a tree rooted at $\emptyset$. A probability distribution P is said to satisfy the tree conditional independence Markov property *(TCIMP) if, for every pair of vertices $u, w \in V$ inducing two subtrees $T_u = (U, E_U)$ and $T_w = (W, E_W)$ with meet $\varphi_{u,w}$ and meet path $mp(\varphi_{u,w})$, P satisfies*

$$U \perp\!\!\!\perp W \mid mp(\varphi_{u,w}).$$

This Markov property leads to the following new type of graphical Markov model.

Definition 2.21. *TCI Markov model*
Let G be a labeled tree rooted at $\emptyset$. The set $\mathbf{T}(G)$ of all probability distributions that satisfy the TCIMP relative to G is called the TCI model *determined by G.*

To illustrate, consider the three TCI models determined by the trees in Figure 2.7. By the TCIMP, the tree in (a) renders the three vertices marginally independent: $1 \perp\!\!\!\perp 2 \perp\!\!\!\perp 3$. From the tree in (b) it is not possible to read off any conditional independencies, thus the set of restrictions of the model is empty. In (c) we may see that the vertex 2 is the meet of vertices 1 and 3, thus $1 \perp\!\!\!\perp 3|2$. These restrictions may be read from their corresponding moral TDAGs using the directed global Markov property (DGMP). Further examples are provided by Figures 2.8 and 2.9.
In Figure 2.8 we see the different graphical representations for three simple models of conditional independence, with the independencies as specified. In Figure 2.9 we find a larger model which may help to understand the TCIMP. For instance, if we pick the vertices 15 and 21, and apply the TCIMP, we see that the set $\{15, 18, 19\}$ is conditionally independent of $\{21, 22, 23, 24\}$ given $\{2, 3, 14\}$. While if we pick the vertices 12 and 13, the TCIMP renders the singletons $\{12\}$ and $\{13\}$ conditionally independent given $\{1, 7, 11\}$.

To show that DEC$\cap$LCI coincides with TCI, we first have to investigate the relationship between TCIMP and the well-known Markov properties. To do this, we need some definitions.

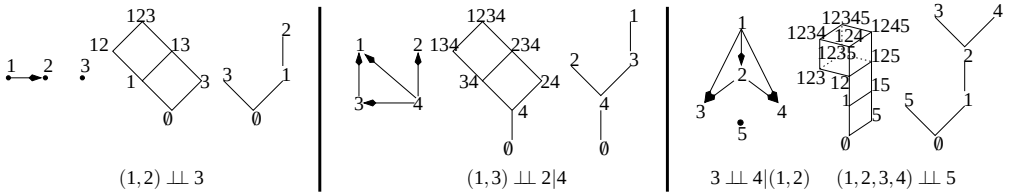


Figure 2.8: Three different Markov models represented by a moral TDAG model, a LCI model and a TCI model.

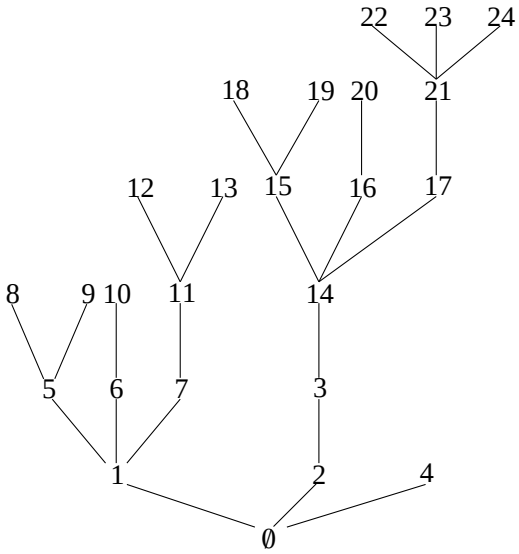


Figure 2.9: Example of a TCI model for 24 variables.

Definition 2.22. *Moral ancestral set*
Let $G = (V, E)$ be a DAG. Given a subset $A \subseteq V$, A is said to be moral ancestral *iff for every vertex $v \in A$, $an(v) \subseteq A$ and $an(v) \cup \{v\}$ is complete in G.*

Proposition 2.5. *Let $G = (V, E)$ be a DAG. Given two moral ancestral subsets $A, B \subseteq V$, the* union *$A \cup B$ is again moral ancestral in G.*

Proof. It is already known that the union of ancestral sets is ancestral. Thus, it is only necessary to determine whether the union of moral ancestral sets is moral.

Let $a, b, c \in A \cup B$ be such that $a \to c \leftarrow b$ where a and b are non-adjacent. Since $a, b \in an(c)$, either $a, b, c \in A$ or $a, b, c \in B$, which would contradict the initial assumption that A and B are moral ancestral. □

Proposition 2.6. *Let $G = (V, E)$ be a DAG. Given two moral ancestral subsets $A, B \subseteq V$, the* intersection *$A \cap B$ is again moral ancestral in G.*

Proof. This follows firstly from the fact that the intersection of ancestral sets is again ancestral. And secondly, since $A \cap B \subseteq A$ and $A \cap B \subseteq B$ then $A \cap B$ should be moral. Otherwise it would contradict the assumption of A and B being moral ancestral. □

From the previous propositions it follows that the moral ancestral sets contained in a DAG $G = (V, E)$ form a ring of subsets of V, which will be called the *moral ancestral ring* of G, and denoted as $\mathcal{A}^m(G)$. The moral ancestral ring allow us to define the TCIMP in terms of DAGs.

Definition 2.23. *Directed tree conditional independence Markov property (DTCIMP) Let $G = (V, E)$ be a DAG. A probability distribution P is said to satisfy the* directed tree conditional independence Markov property (DTCIMP) *if, for every pair of moral ancestral subsets $A, B \in \mathcal{A}^m(G)$, P satisfies*

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

Theorem 2.6. *Let $\mathbf{D}(G)$ be a DAG model. For any probability distribution P,*

$$\text{DGMP} \Rightarrow \text{LCIMP} \Rightarrow \text{DTCIMP} \Rightarrow \text{TCIMP}.$$

Proof. The first implication follows from Theorem 2.3. For the second, let $A, B \in \mathcal{A}^m(G)$. The LCIMP implies the DTCIMP if $A, B \in \mathcal{A}(G)$, and this follows because $\mathcal{A}^m(G) \subseteq \mathcal{A}(G)$.

The third implication is proved as follows. For any pair $A, B \in \mathcal{A}^m(G)$, the set $A \cup B$ induces a moral TDAG $G_{A\cup B}$ from G, such that it coincides with a tree $T_{A\cup B}$ by Theorem 2.5. The DTCIMP will imply the TCIMP if for each pair of vertices $a \in A \backslash B$ and $b \in B \backslash A$, $mp(\varphi_{a,b}) = A \cap B$ in $T_{A\cup B}$. This equality follows from the fact that the meet path in $T_{A\cup B}$, for any pair of vertices $a \in A \backslash B$ and $b \in B \backslash A$, is formed by those vertices that are common to A and B, therefore $A \cap B$. □

Theorem 2.7. *Let G be a moral TDAG. For any probability distribution P,* TCIMP $\Rightarrow$ DGMP. *Thus, for a moral TDAG,*

$$\text{TCIMP} \Leftrightarrow \text{DGMP} \Leftrightarrow \text{DLMP} \Leftrightarrow \text{LCIMP} \Leftrightarrow \text{DTCIMP}$$

and $\mathbf{D}(G) = \mathbf{T}(T)$, for some tree T that coincides with the moral TDAG G.

Proof. By Theorem 2.5, there is a unique labeled tree $T = (V, E)$ that coincides with the moral TDAG G. For any two vertices $u, w \in V$, that induce subtrees $T_u = (U, E_U)$, $T_w = (W, E_W)$, the TCIMP in T implies the DGMP in G if U and W are separated by $mp(\varphi_{u,w})$ in

$$(G_{An(U\cup W\cup mp(\varphi_{u,w}))})^m = G_{An(U\cup W\cup mp(\varphi_{u,w}))}.$$

This equality follows since G is assumed to be moral. Now, we should find out which set separates U and W in the graph specified on the right hand of this equality. Consider a path between any two vertices $a \in U$ and $b \in W$. Since U, W were induced by vertices u, w, this path will intersect the sets $pa(u)$ and $pa(w)$, because of transitivity.

More concretely, this path will always intersect those vertices $x \in pa(u) \cap pa(w)$. The set $pa(u) \cap pa(w)$ in a moral TDAG is equivalent to the definition of meet path, hence $mp(\varphi_{u,w})$ separates U, W in $G_{An(U \cup W \cup mp(\varphi_{u,w}))}$. The second part of the theorem follows from Theorems 2.4 and 2.6. □

Finally, we can establish the following theorem that determines the location of TCI models, within the family of graphical Markov models.

Theorem 2.8. *The class of TCI models coincides with the class of DEC∩LCI models.*

Proof. It follows from the fact that $\mathbf{D}(G) = \mathbf{T}(T)$, for some moral TDAG G and some labeled tree T, which is proved in Theorem 2.7. □

2.7.4 Markov equivalence among TCI models

Recall from subsection 2.4.3 that DAG models are organized in classes of equivalence. This situation also can occur in the case of TCI models: two different trees T_1, T_2 may determine the same TCI model $\mathbf{T}(T_1) = \mathbf{T}(T_2)$. We will investigate now the notion of Markov equivalence among TCI models. Recall the notion of Markov equivalence for DAG models, which we already saw in this chapter.

Theorem *(Frydenberg, 1990a; Verma and Pearl, 1990)*
Two DAG models are Markov equivalent if and only if they have the same skeleton and the same immoralities.

It is possible to decide Markov equivalence for TCI models by simply creating the corresponding moral TDAG using Theorem 2.5 and applying the previous theorem. The notion specifically for TCI models is as follows.

Definition 2.24. *Let $T = (V \cup \{\emptyset\}, E)$ be a labeled tree rooted at vertex $\emptyset$. Let $l \in V$ be a leaf in T. We define a* branch ending at *l as the set of vertices $\pi(l) = \{x_1, \ldots, x_n\}$, where $x_1 = \emptyset$ and $x_n = l$, present in the unique path between the root $\emptyset$ and the leaf l. The set of all branches of T will be denoted as $\lambda(T)$.*

Theorem 2.9. *Two TCI models $\mathbf{T}(T)$ and $\mathbf{T}(T')$ are Markov equivalent, i.e. $\mathbf{T}(T) = \mathbf{T}(T')$, if and only if they have the same sets of branches, i.e. $\lambda(T) = \lambda(T')$.*

In order to illustrate the notion of Markov equivalence among trees, look at Figure 2.10. The pairs of trees on part (a) are Markov equivalent because although two vertices are swapped between the trees, the paths from the leaves to the root remain the same. The two trees on part (b) are not Markov equivalent because given the swap of vertices 2 and 4, although it does not change the paths from vertices 5 and 6 to the root $\emptyset$, it does it from vertex 3 to the root $\emptyset$.

Proof. Theorem (2.9).
(Necessity). Assume that $\mathbf{T}(T) = \mathbf{T}(T')$. Then, any TCIMP read from T, holds also in T'. A TCIMP involves the vertex sets of two subtrees and the vertex set of their meet path. If a TCIMP holds in T and in T', then the two vertex sets of the two subtrees and the vertex set of their meet path in T, should be derived also from T'.

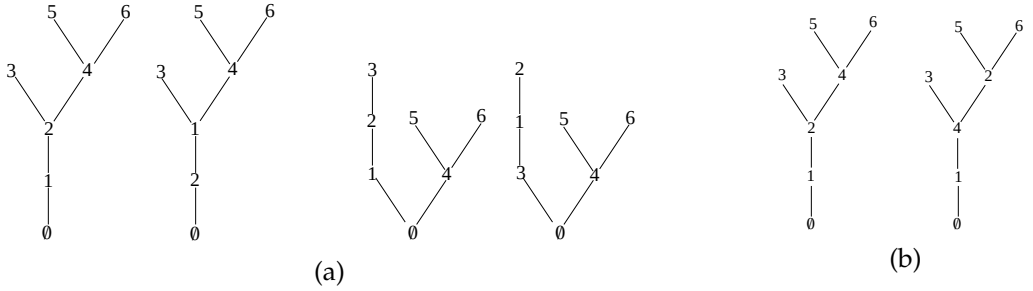
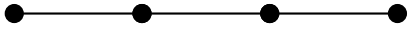


Figure 2.10: Markov equivalence between TCI models. The two pairs on (a) are Markov equivalent, while the pair on (b) is not.

Let u, w be two vertices of the tree $T = (V, E)$, with meet $\varphi_{u,v}$, inducing subtrees $T_u = (U \cup \{\emptyset\}, E_U), T_w = (W \cup \{\emptyset\}, E_U)$ such that none of them is subtree of the other. In order to find the same meet path $mp(\varphi_{u,v})$ for u, v in both trees T and T', the paths from u, v to the root must intersect in the same vertices in T and T'. Because this must happen for every pair of vertices in T and T', it follows that the only way that the intersections of all the paths are the same in T and T', is when $\lambda(T) = \lambda(T')$.
(Sufficiency). Assume that $\lambda(T) = \lambda(T')$. This implies that every possible meet path from any given two subtrees in T must exist in T'. Because, if there were a meet path that differs in at least one vertex in T and T', there would exist some $\pi(l) \in \lambda(T)$ and $\pi'(l) \in \lambda(T')$ such that $\pi(l) \neq \pi'(l)$. Therefore, if for every two given subtrees in T, their meet path is the same in T', it follows from the TCIMP that the collection of Markov properties of T hold also in T', and viceversa. □

2.7.5 A canonical representation of an equivalence class of TCI models

Andersson et al. (1997c) characterized TCI Markov models as those DEC models $\mathbf{U}(G)$ determined by a chordal graph G such that G does not contain the following induced undirected subgraph:



which is a path on four vertices and denoted as P_4. The term P_4-free is used in this context to denote the absence of such induced subgraph. In the graph theory literature Wolk (1962) gave a first characterization of these graphs, although it was Golumbic (1978) who described them later in the terms of forbidden subgraphs as above, hence the name "P_4-free chordal". From this characterization follows a lemma (Wolk, 1965) and a proposition we derive in section 2.8. We need to introduce them here for our current purposes.

Lemma 2.3 *Wolk (1965)*
Let G be a connected P_4-free chordal graph. Let G have more than one clique. The intersection of all the cliques of G is non-empty.

Proposition 2.8 *Castelo and Wormald (2001)*
Let $G = (V, E)$ be a connected undirected graph. Let D be the set of vertices of degree $|V| - 1$. Then G is P_4-free and chordal if and only if it is complete or $G - D$ is a disconnected P_4-free chordal graph.

The P_4-free chordal graph characterization of TCI models suggests a canonical representation of Markov equivalence classes of TCI models. This representation will have again the form of a tree, but its nodes will contain, possibly, more than one single vertex (i.e. more than one single random variable). First we show how the cliques of a connected P_4-free chordal graph lead to a tree organization of their intersections. This allows a representation for the canonical element of an equivalence class of TCI models. Finally, it will be shown how to extract all the members of the equivalence class from this canonical representation.

By Lemma 2.3, a connected P_4-free chordal graph containing more than one clique has a non-empty subset of vertices D which correspond to the intersection of all cliques of the graph. By Proposition 2.8, the resulting graph, after the removal of D, will consist of $k > 1$ disconnected components that are again P_4-free chordal graphs. Now repeat the previous operation recursively until no disconnected component contains more than one clique. At each step of this operation, we will keep track of the different intersecting sets, and we will draw undirected edges from a given intersecting set, to those intersecting sets derived from the disconnected components that were created. It follows that such an undirected structure cannot have undirected cycles, thus has the form of a tree. In Figure 2.11b we may see the P_4-free chordal graph corresponding to the TCI model of Figure 2.11a, which corresponds to one of the subtrees of the TCI model of Figure 2.9. In Figure 2.11c we may see a first step of the procedure we just described, and in 2.11d we may see the second and last step, from which we already obtain the canonical representation.

In graph-theoretic terms, the canonical representation of a TCI model, as for instance the one in Figure 2.11d, corresponds to *homeomorphically irreducible trees* (Harary and Palmer, 1973). Homeomorphically irreducible trees are those trees in which no vertex has degree of adjacency equal to two.

In order to find the members of the equivalence class, one only needs to perform all possible permutations on those nodes of the canonical element that contain more than one vertex. Then build a path for a given permutation, on which the vertices on the extremes of the path will connect to the adjacent nodes in the canonical element. For a given TCI model with more than one branch growing from the root $\emptyset$ one just applies this process to each of the branches separately, and plants the bottom roots (the first intersection set removed) on the root $\emptyset$. For a given canonical element with $s_1, \ldots, s_k$ nodes that contain more than one vertex, the number of trees on that equivalence class will amount to $|s_1|! \ldots |s_k|!$. The reason is obvious since it just corresponds to the number of possible permutations of those nodes that may be exchangeable on the tree.

2.7.6 Marginalization and collapsability in TCI models

GMMs are a powerful tool to understand better the mechanism that generates the data. However, we are only able to see a small part of this mechanism through the data we

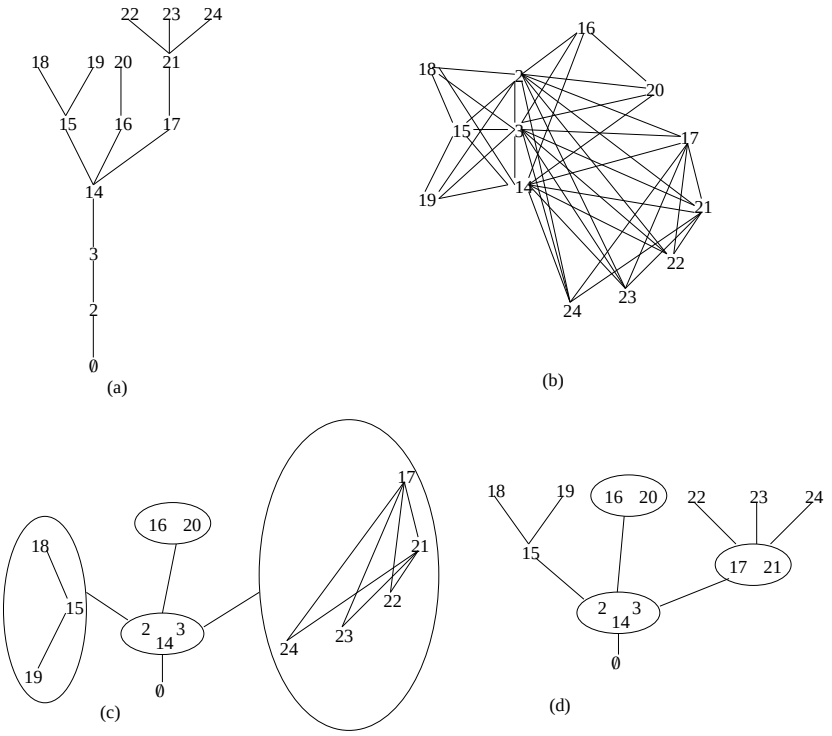


Figure 2.11: (a) A TCI model. (b) Its corresponding P_4-free chordal graph. In (c) and (d) the two steps to obtain the canonical form of the equivalence class are shown.

observe. We may easily neglect to store certain variables, or simply it may be impossible to measure them. Those variables that we do not observe are usually referred to as *hidden variables*. The data generated by the set of variables we observe is the result of *marginalizing* over those that we do not observe. Very often, the observed variables interact differently when some newly discovered hidden variables are incorporated. Think, for instance, of the introductory example of conditional independence we saw at the beginning of this chapter. Therefore it is useful to have some idea in advance on how the mechanism *marginalizes* over the hidden variables. For such goal, one may study how a GMM changes after marginalization over an arbitrary subset of variables.

Analogously, one may be interested in finding out whether there is some subset of variables for which the way they interact is not affected by any marginalization. One says that such set of variables are *collapsible*.

Marginalization and collapsability have been already studied in the context of UG Markov models for discrete data (Asmussen and Edwards, 1983; Frydenberg, 1990a; Studeny, 1997; Castillo et al., 1998). We will first recall the most relevant results to show how collapsible subsets of variables in a TCI model can be identified. Then we will show how a tree *behaves* under marginalization. We begin by reviewing the notions of *simplicial subset* and *simplicial collection*.

Definition 2.25. *Let $G = (V, E)$ be an UG. A subset $B \subset V$ is called* simplicial *in G iff $bd(B)$ is complete and a* simplicial collection *in G iff every connected component in B is simplicial in G.*

In this definition we have used the notion of boundary with respect to a set, while so far we had done that only for a single vertex. The boundary of a given subset of variables is analogously defined as $bd(A) = \{v \in V \,|\, (v, u) \in E, u \in A, v \notin A\}$. The graphical characterization of a collapsible subset of vertices in an UG is as follows.

Proposition 2.7. *(Frydenberg, 1990b)*
Let $G = (V, E)$ be an UG. Let P be any probability distribution Markov over G and P_A the marginalized distribution of P over $V \backslash A$. The UG G is collapsible onto the subset $A \subset V$ and P_A is Markov over G_A, iff $V \backslash A$ is a simplicial collection.

This characterization follows from the fact shown in Asmussen and Edwards (1983) by which in a subgraph G_A, obtained by collapsing G onto $A \subset V$, if S separates A_1 and A_2 in G_A, then they are separated by S in G too.

The characterization of collapsible sets of variables in TCI Markov models is given by the following theorem.

Theorem 2.10. *Let $\mathbf{T}(G)$ be a TCI Markov model determined by a labeled tree $G = (V \cup \{\emptyset\}, E)$ rooted at $\emptyset$. Let $\mathbf{T}_A(G)$ be the set of probability distributions $\mathbf{T}(G)$ marginalized over $V \backslash A$. The tree G is collapsible onto the subset $A \subset V$ and $\mathbf{T}(G_A) = \mathbf{T}_A(G)$, iff for every branch $\pi(l) \in \lambda(G_A)$ there is a branch $\pi(h) \in \lambda(G)$ such that $\pi(l) \subseteq \pi(h)$.*

Proof. Let $A \subset V$ be the subset of all collapsed vertices. Let $\pi(v) \subseteq V$ be the collapsed vertices of a branch (see Definition 2.24) ending at some vertex v. Let $\pi(l)$ be the entire set of vertices of the same branch ending at the corresponding leaf vertex l, thus $\pi(v) \subseteq \pi(l)$. The boundary of $\pi(l) \backslash \pi(v)$ on G will be exactly $\pi(v)$ because the vertices in $\pi(l) \backslash \pi(v)$ are higher in the hierarchy than those in $\pi(v)$. Therefore, in the corresponding chordal graph $\pi(l) \backslash \pi(v)$ are connected to those vertices lower in the hierarchy (see Figure 2.12). The subset $\pi(v)$ is complete because otherwise it would imply that the corresponding TDAG is not moral. Further, for every leaf l, the subset $\pi(l) \backslash \pi(v)$ induces a connected component in the P_4-free chordal graph, which is isolated of every other $\pi(l') \backslash \pi(v')$ $l \neq l'$, $v \neq v'$. Thus, the union of $\pi(l) \backslash \pi(v)$ for every leaf l, i.e. $V \backslash A$, forms a simplicial collection. □

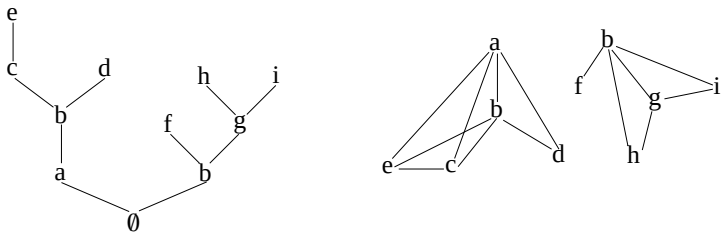


Figure 2.12: A TCI model on the left and its equivalent P_4-free chordal graph on the right.

In order to discuss how a TCI model *changes* after marginalizing over an arbitrary set of random variables we need to review the following notion. Studeny (1997) introduced the *precollapsability* property for discrete UGs models. This property, retained by UG models, describes how for any given UG $G = (V, E)$ and any subset $A \subset V$, we can obtain a *marginal graph* $G_{V\setminus A}$ such that $\mathbf{U}(G_{V\setminus A}) \supseteq \mathbf{U}_{V\setminus A}(G)$, where $\mathbf{U}_{V\setminus A}(G)$ denotes the set of probability distributions $\mathbf{U}(G)$ marginalized over A.

Since TCI models are a subclass of UG models, they also retain the precollapsability property. Castillo et al. (1998) provide an equivalent notion of precollapsability for UG and hypergraph[6] models.

Let $G = (V, E)$ be an undirected graph and $A \subset V$ a subset of variables over which we marginalize. Let $A' = V\setminus A$ be the variables of which we obtain the marginal model. After marginalization we will obtain the graph $G_{A'} = (A', E_{A'})$ where (Studeny, 1997; Castillo et al., 1998),

$$u - v \in E_{A'} \Leftrightarrow \neg\,(u \perp\!\!\!\perp v \,|\, A'\setminus\{u, v\}[G])\,.$$

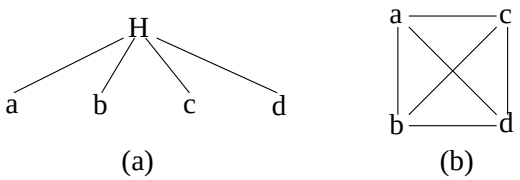


Figure 2.13: In (a) an UG Markov model. In (b) the UG Markov model (a) marginalized over H.

This means, that every pair of variables that were connected by a path intersecting the marginalized set A, should be joined by an edge in $G_{A'}$. In other words, the boundary $bd(A)$ is made complete in the new graph $G_{A'}$. In Figure 2.13a we have an UG[7] model where a variable H renders the rest conditionally independent between them. This happens, for instance, when there is some hidden class variable H that makes the rest of variables independent for a fixed class level. In Figure 2.13b we have the same UG model after marginalizing over H. In order to introduce an analogous result for TCI models we need to define the concept of pruning a set of vertices in a tree.

Definition 2.26. *Pruning*
Let $\mathbf{T}(G)$ *be a TCI Markov model determined by a labeled tree* $G = (V \cup \{\emptyset\}, E)$ *with set of branches* $\lambda(G)$. Pruning *a subset of vertices* $A \subset V$ *in the tree* G *is the construction of a new tree* $G_{V\setminus A} = (\{V\setminus A\} \cup \{\emptyset\}, E_{V\setminus A})$, *in the following way.*

Let $E_{V\setminus A} = E$. *For every branch* $\pi(l) \in \lambda(G)$, *where* $\pi(l) = \{x_1, \ldots, x_m\}$, $x_1 = \emptyset$, $x_m = l$ *and* l *is a leaf in* G, *such that* $\pi(l) \cap A \neq \emptyset$, *consider the vertex* $x_i \in \pi(l) \cap A$ *for which any other* $x_j \in \pi(l) \cap A$ *with* $j \leq i \Rightarrow j = i$. *Let* $G_{x_i} = (V_{x_i}, E_{x_i})$ *be the subtree induced by the vertex* x_i. *Remove* E_{x_i} *from* $E_{V\setminus A}$. *Without loss of generality, let* $V_{x_i}\setminus A = \{z_1, \ldots, z_q\}$. *Add the undirected edge* $\{(z_k, z_{k+1}), (z_{k+1}, z_k)\}$ *to* $E_{V\setminus A}$ *for* $1 \leq k < q$. *Finally, add the undirected edge* $\{(x_{i-1}, z_1), (z_1, x_{i-1})\}$ *to* $E_{V\setminus A}$.

[6] We do not talk about hypergraph models in this thesis, but we refer the reader to the given source.
[7] Note that it coincides with a DEC, a DAG, a LCI and a TCI model.

The intuition behind Definition 2.26 is that pruning entire subtrees implies only the removal of the vertices and the edges involved. While pruning arbitrary vertices throughout the tree implies not only their removal, but also a transformation of the remaining branches. In the first case, we obtain a new tree which is a subgraph of the one we prune, and for that reason we call it a *closed pruning*. In the second case, we obtain a new tree which is not a subgraph of the one we prune, and for that reason we call it an *open pruning*.

In Figure 2.14 we may see examples of open and closed pruning. The vertices inside the circles denote the vertices we want to prune (remove).

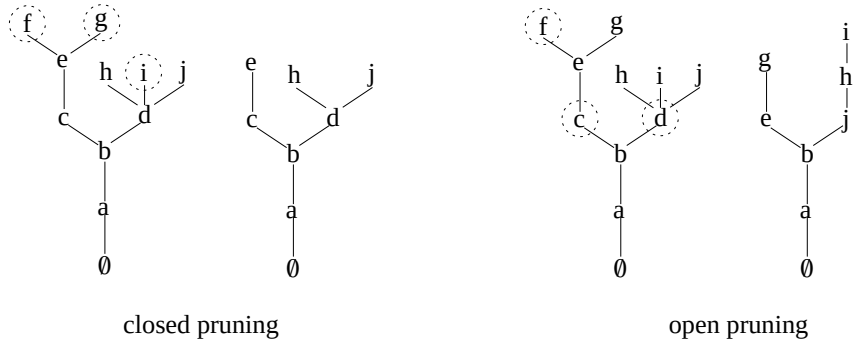


Figure 2.14: The transformation on the left is a closed pruning and the one on the right is an open pruning.

The following theorem shows that pruning a tree is equivalent to marginalize the TCI model over an arbitrary set of variables.

Theorem 2.11. *Let* $\mathbf{T}(G)$ *be a TCI model determined by a labeled tree* $G = (V, E)$*. Let* $\mathbf{T}_{V\setminus A}(G)$ *be the set of probability distributions* $\mathbf{T}(G)$ *marginalized over some subset* $A \subset V$*. Let* $G_{V\setminus A}$ *be the labeled tree after pruning the vertices* A *from* G*. Then,* $\mathbf{T}_{V\setminus A}(G) = \mathbf{T}(G_{V\setminus A})$*.*

Proof. If the pruning is closed then by Theorem 2.10, $V\setminus A$ is a collapsible subset of vertices in G. From the definition of collapsible subset it follows immediately that $\mathbf{T}_{V\setminus A}(G) = \mathbf{T}(G_{V\setminus A})$.

If the pruning is open, then every pair of vertices higher in the hierarchy than some pruned vertex will form part of a single branch. In the corresponding P_4-free chordal graph this implies they will become adjacent. This corresponds to the transformation performed when marginalizing an UG model, therefore $\mathbf{T}_{V\setminus A}(G) = \mathbf{T}(G_{V\setminus A})$. □

2.8 Organization and size of the classes

In this chapter we have surveyed some of the different subclasses of GMMs within the class of the DAG Markov model. In this section we will overview how these classes are organized and their sizes. The size of a class of GMMs is a valuable information for two reasons. First, the size up to equivalence allows to distinguish which classes are more

expressive than others in terms of how many different CI models can they represent. Second, the size according to the type of the graph that determines the model allows to distinguish which search spaces will be larger when performing structural learning. This is relevant as a larger search space leads usually to a harder learning problem. We will see in the next chapter how the knowledge about the sizes of the classes leads to certain design choices in the development of a new learning algorithm.

The organization of the classes is an interesting information as it reveals also details about the search space that may be relevant later in the learning task. In Figure 2.15 we can see a picture that Andersson et al. (1995) devised in order to describe the location of LCI models within the scope of models represented by undirected and directed graphs. Although the class of LCI models appears on the picture as an isolated subclass, Andersson et al. (1995, pg. 38) show that they are in fact interlaced[8] through the class of DAG models. An important characterization also depicted in this figure corresponds to the definition of those DAG models that are equivalent to some UG model (Wermuth, 1980; Kiiveri et al., 1984). Thus, undirected and directed graphs in this intersecting class describe the same model of conditional independence. More concretely those DAG models determined by moral DAGs are equivalent to some DEC model.

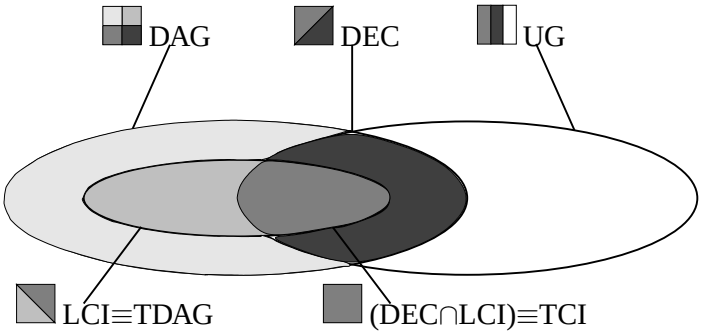


Figure 2.15: Relation among the classes acyclic directed graph models (DAG), undirected graph models (UG), decomposable models (DEC) and lattice conditional independence models (LCI).

As we may also see in Figure 2.15, TCI models correspond to DEC∩LCI, which was formally showed in the previous section. Andersson et al. (1995, pg. 38) claimed that since every conditional independence statement $A \perp\!\!\!\perp B|C$ is equivalent to a simple LCI model, then any DAG model is the intersection of all LCI models that contain it. We can see further that every conditional independence statement $A \perp\!\!\!\perp B|C$ is equivalent to a simple TCI model, therefore any DAG model is the intersection of all TCI models that contain it. A remaining question is how TCI models can be combined graphically to determine the DAG structure of the intersection of TCI models.

The notion of Markov equivalence among GMMs leads to the problem of having to search for an alternative graphical representation that has a one to one correspondence with the equivalence classes. When the equivalence class of a given type of graphical Markov models has a precise graphical definition, one may try to use standard tools

[8]Every LCI model *includes* and it is *included* in at least one DAG model. See section 3.4 in next chapter.

of graph theory and graphical enumeration to count how many models of conditional independence may be represented.

The most straightforward case is that of UG models, which are represented by undirected graphs, since there is a one to one correspondence between undirected graphs and models of conditional independence. Two UG models $\mathbf{U}(G_1), \mathbf{U}(G_2)$ are Markov equivalent, $\mathbf{U}(G_1) = \mathbf{U}(G_2)$, iff $G_1 = G_2$. Thus, there is the same number of different models of conditional independence, as different undirected graphs, i.e. $2^{\binom{n}{2}}$ for labeled graphs with n vertices.

The case of DEC models is also straightforward, since chordal graphs also have a one to one correspondence with Markov equivalence classes. Connected and disconnected chordal graphs were counted by Wormald (1985). The sum of these two quantities provides the number of all chordal graphs, i.e. the number of all different DEC models. We will treat in this section the derivation of the expression that allows us to compute the number of all chordal graphs from the numbers of connected chordal graphs. As we shall see, this expression is related to the computation of Markov equivalence classes of TCI models.

The case of DAG models is a difficult one. As we saw in this chapter, Markov equivalence classes of DAG models are represented by essential graphs. An efficient way of enumerating such graphs is not yet known. In (Andersson et al., 1997a) they were counted up to 5 vertices. Recently, Gillispie and Perlman (2001) developed a computer program which has calculated the number of essential graphs up to 10 vertices, by enumerating DAGs and taking into account the equivalence class to which each DAG belongs. We have taken the liberty to extrapolate these numbers up to 12 vertices, such that we can compare cardinalities of different Markov equivalence classes in Table 2.2.

The basic mathematical tool used in the enumeration of graphs is that of *generating functions*. A generating function is a power series. The coefficients of the polynomial that forms these power series store the counts of the object we intend to enumerate. The exponents of this polynomial describe some structural feature associated to its attached coefficient, as for instance, the number of vertices of a graph. In the case of labeled enumeration, one uses an *exponential generating function* of the form (2.5). For full insight into this subject the reader should consult the book of Harary and Palmer (1973).

$$g(x) = \sum_{n=1} a_n \frac{x^n}{n!}. \tag{2.5}$$

Let $g(x)$ be the generating function for connected labeled chordal graphs. Then a_n corresponds to the number of such graphs with n vertices. Consider now another exponential generating function to count, not only *connected* labeled chordal graphs, but *all* of them,

$$G(x) = \sum_{n=0} A_n \frac{x^n}{n!}. \tag{2.6}$$

In this generating function, the coefficient A_n is the number of *all* chordal graphs with n vertices, which corresponds to the number of Markov equivalence classes of DEC models. These two exponential generating functions are related through the following theorem.

Theorem 2.12. *Harary and Palmer (1973, pg. 8)*
The exponential generating functions $G(x)$ and $g(x)$ for labeled graphs and labeled connected graphs satisfy the following relation

$$1 + G(x) = e^{g(x)}. \tag{2.7}$$

Here the constant 1 refers to the null graph, i.e. the graph with no vertices. In the way we have expressed the generating function $G(x)$, the constant 1 is included in $G(x)$ since n starts on 0 vertices. Therefore we may discard the constant 1 in expression (2.7). Note that Theorem 2.7 holds for every type of labeled graph. As we shall see now, by differentiating the previous equation and equating coefficients, it is possible to find a recurrence for both the number of all labeled chordal graphs A_n and labeled connected chordal graphs a_n. First, $g(x)$ is isolated, by taking logarithms on both sides, and afterwards we can differentiate the equation, which leads to the following relation:

$$\frac{\sum_{n=0}^{\infty} n \frac{A_n}{n!} x^{n-1}}{\sum_{n=0}^{\infty} \frac{A_n}{n!} x^n} = \sum_{n=1}^{\infty} n \frac{a_n}{n!} x^{n-1}.$$

Now multiply both sides by the polynomial at the bottom-left of the equation, to obtain

$$\sum_{n=0}^{\infty} n \frac{A_n}{n!} x^{n-1} = \sum_{n=0}^{\infty} \left(\sum_{k=0}^{n} (k+1) \frac{a_{k+1}}{(k+1)!} \frac{A_{n-k}}{(n-k)!} \right) x^n.$$

In order to equate coefficients, the exponents of both polynomials should match. Therefore we are going to move the running indexes on the right-hand side of the equation. First, move the index k of the inner sum, and then move the index n. Further, the first term of the sum in the left-hand side may be discarded since it cancels for $n = 0$. Thus we obtain

$$\sum_{n=0}^{\infty} n \frac{A_n}{n!} x^{n-1} = \sum_{n=0}^{\infty} \left(\sum_{k=1}^{n+1} k \frac{a_k}{k!} \frac{A_{n+1-k}}{(n+1-k)!} \right) x^n,$$

$$\sum_{n=1}^{\infty} n \frac{A_n}{n!} x^{n-1} = \sum_{n=1}^{\infty} \left(\sum_{k=1}^{n} k \frac{a_k}{k!} \frac{A_{n-k}}{(n-k)!} \right) x^{n-1}.$$

We can now equate coefficients, and for our purposes, we will isolate from the sum the term for $k = n$. In this term, we can substitute afterwards $A_0 = 1$ since the null-graph is unique, to obtain

$$n \frac{A_n}{n!} = n \frac{a_n}{n!} \frac{A_0}{0!} + \sum_{k=1}^{n-1} k \frac{a_k}{k!} \frac{A_{n-k}}{(n-k)!}.$$

Finally, by multiplying the whole expression by $n!$ and dividing it by n, we obtain the recurrence for all chordal graphs for n vertices:

$$A_n = a_n + \frac{1}{n}\left(\sum_{k=1}^{n-1} k\binom{n}{k} a_k A_{n-k}\right). \tag{2.8}$$

Table 2.1: Number of Markov equivalence classes of DEC models

A_n	n
1	1
2	2
8	3
61	4
822	5
18,154	6
617,675	7
30,888,596	8
2,192,816,760	9
215,488,096,587	10
28,791,414,081,916	11
5,165,908,492,061,926	12

Wormald (1985) provides the numbers a_n for labeled connected chordal graphs; thus by using these and the formula (2.8), we obtain the numbers of all labeled chordal graphs; which equals the number of Markov equivalence classes of DEC models, given in Table 2.1.

Next we count the Markov equivalence classes of TCI models. We already mentioned that TCI models may be determined by P_4-free chordal graphs, where P_4-free chordal graphs were defined as those chordal graphs with no path on four vertices as an induced subgraph. From such characterization Wolk (1962, 1965) derived the following two lemmas. For a better understanding of this class of graphs, we have rewritten and included the proofs.

Lemma 2.2. *Wolk (1962, Lemma 1)*
Let G be a connected P_4-free chordal graph. Let G have more than one clique. For every clique C of G, the intersections of all other cliques with C are nested.

Proof. Let's assume that the intersections of a given clique C with its adjacent cliques are not nested. Then, there exist cliques C_1 and C_2, where $C \neq C_1 \neq C_2$, and vertices $0, 1, 2, 3$, such that: $0 \in C_1$ but $0 \notin C$ and $0 \notin C_2$; $1 \in C_1$ and $1 \in C$, but $1 \notin C_2$; $2 \in C$ and $2 \in C_2$ but $2 \notin C_1$; $3 \in C_2$ but $3 \notin C_1$ and $3 \notin C$. In this way, a forbidden path 0–1–2–3 on four vertices, is introduced. □

Lemma 2.3. *Wolk (1965)*
Let G be a connected P_4-free chordal graph. Let G have more than one clique. The intersection of all the cliques of G is non-empty.

Proof. Let's consider two cliques C_1, C_2 from G which do have non-empty intersection. Let's consider a third clique C_3 that intersects with C_2. Since the intersections of C_2, with

C_1 and C_3, are nested (Proposition (2.2)), they form a poset which is, in fact, a chain and therefore C_1 and C_3 have a non-empty intersection. Hence, C_1, C_2 and C_3 have non-empty intersection.

Inductively, let's consider k cliques having non-empty intersection. Each of them has associated a chain $H_i, 1 \leq i \leq k$ that contains the intersections with respect to the i clique. Let's consider now any $k+1$ clique which intersects one of the previous k cliques. This latter intersection, namely p, must form part of the H_k chain, as well as of the H_{k+1}. From the definition of a chain, given any other element q such that $q \in H_k$ and $q \in H_{k-1}$, it follows directly that either $p \subset q$ or $q \subset p$. Therefore p intersects, not only the k clique, but all of them. Hence this gives $k+1$ cliques with non-empty intersection. By induction, all cliques can be included. □

These two properties of P_4-free chordal graphs lead to the following proposition.

Proposition 2.8. *Castelo and Wormald (2001)*
Let $G = (V, E)$ be a connected undirected graph. Let D be the set of vertices of degree $|V| - 1$. Then G is P_4-free and chordal if and only if it is complete or $G - D$ is a disconnected P_4-free chordal graph.

Proof. We proceed by induction on $n = |V|$. For $n = 1$ it is immediate. The complete case is trivial, so assume that G is a connected P_4-free chordal non-complete graph. By Lemma 2.3, G must have at least one vertex of degree $|V| - 1$. Let u be a vertex of maximum degree, and let v be adjacent to u. Let v be adjacent to w which is not adjacent to u. Then, for any vertex x adjacent to u, the fact that $\{w, v, u, x\}$ does not form a path, or a chordless cycle, on four vertices implies that x and v are adjacent. Since $G' = G - v$ is an induced subgraph of a P_4-free chordal graph, G' is P_4-free chordal too.

If G' is disconnected, we are done. If G' is connected, then by induction $G' - D'$ is a disconnected P_4-free chordal graph. But clearly $G - D = G' - D'$. The proposition follows. □

Consider again the exponential generating functions (2.5) and (2.6) as the ones corresponding to connected P_4-free chordal graphs, and all P_4-free chordal graphs, respectively. The coefficients a_n will refer to the number of connected P_4-free chordal graphs, and A_n will refer to the number of all P_4-free chordal graphs. Let G be a connected P_4-free chordal graph on n vertices. The removal of the intersection set D from G produces $k > 1$ disconnected components, therefore the subset of vertices D has at most $n - 2$ vertices, otherwise, if D had $n - 1$ vertices there would be left only $k = 1$ component.

A subset D of k vertices may be taken in $\binom{n}{k}$ ways. For each different set D of size k, one should count all possible disconnected P_4-free chordal graphs on $n - k$ vertices, which amounts to,

$$A_{n-k} - a_{n-k},$$

thus, all P_4-free chordal graphs minus the connected ones on $n - k$ vertices. Finally, from the fact that there is always one single complete graph on n vertices, and the above discussion, it follows a recurrence for computing the number of connected P_4-free chordal graphs:

$$a_n = 1 + \sum_{k=1}^{n-2} \binom{n}{k} (A_{n-k} - a_{n-k}). \tag{2.9}$$

Table 2.2: Numbers of Markov equivalence classes of DAGs, UGs, DECs and TCIs with n vertices.

DAG (essential)	UDG (undirected)	DEC (chordal)	TCI (P_4-free chordal)	n
1	1	1	1	1
2	2	2	2	2
11	8	8	8	3
185	64	61	49	4
8,782	1,024	822	402	5
1,067,825	32,768	18,154	4,144	6
312,510,571	2,097,152	617,675	51,515	7
21,213,3402,500	268,435,456	30,888,596	750,348	8
326,266,056,291,213	68,719,476,736	2,192,816,760	12,537,204	9
1,118,902,054,495,975,141	35,184,372,088,832	215,488,096,587	236,424,087	10
$\approx$ 8.53e+21	36,028,797,018,963,968	28,791,414,081,916	4,967,735,896	11
$\approx$ 1.40e+26	73,786,976,294,838,206,464	5,165,908,492,061,926	115,102,258,660	12

By using expression (2.8), we may compute those A_n we need in (2.9). To initialize the recurrence, one should know that the number of connected and all P4-free chordal graph on one vertex is just one ($a_1 = A_1 = 1$), and that on two vertices there is one connected P4-free chordal graph and two P_4-free chordal graphs ($a_2 = 1$ and $A_2 = 2$). We may see the counts in Table 2.2.

As noted before, TCI models may be defined as those graphical Markov models determined by P_4-free chordal graphs. From the characterization presented in the previous section of moral TDAGs as trees, it follows as well that there are $(n+1)^{n-1}$ different moral TDAGs on n vertices. As it has been already said, these quantities on different graphs may serve to quantify, roughly, difficulty of model selection and expressiveness of the graphical Markov model. Thus, it may be interesting to look at the plot of the cardinalities of the different graphs that determine in several forms different types of graphical Markov models, that we may find in Figure 2.16.

2.9 Concluding Remarks

In this chapter we have surveyed the class of DAG Markov models and some of its subclasses. The overview of these few GMMs shows how ideas from different research areas as statistics, artificial intelligence and graph theory, mingle together to provide a useful tool for the analysis of complex interaction systems through the concept of CI.

In this chapter a new class of GMMs, the TCI Markov model, determined by labeled trees has been introduced. It has been shown that the class of TCI models coincides with the intersection class of DEC$\cap$LCI. Moreover, a new Markov property, specific for trees, has been introduced, and its relationship with the other Markov properties, has been investigated.

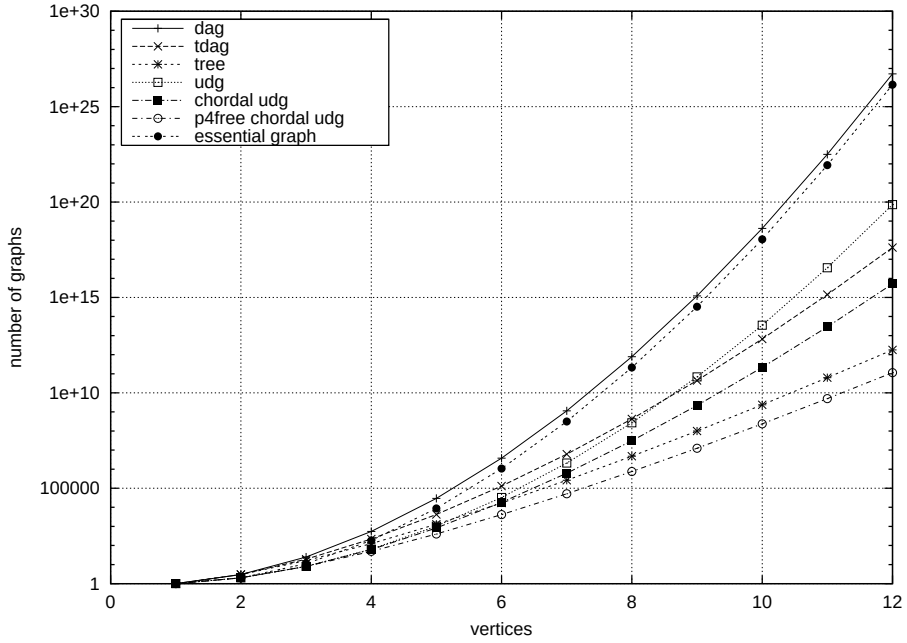


Figure 2.16: Cardinalities of the types of graphs that determine different subclasses of Markov models.

We have sketched the organization of the different classes of GMMs treated here and studied their sizes. From this study we have provided a recurrence for the number of P_4-free chordal graphs, that have a one-to-one correspondence with equivalence classes of TCI models. Moreover, we have found out the following facts that, as before, are interesting graph-theoretic results in their own right, and possibly have some consequence, in our context, from a learning perspective. The canonical representation of an equivalence class of TCI models shows a correspondence between P_4-free chordal graphs and homeomorphically irreducible trees. There are $(n+1)^{(n-1)}$ moral TDAGs (or in graph-theoretic terms, transitive subtree acyclic digraphs) on n vertices. From the number of rooted labeled trees, which is n^{n-1}, it also follows directly that there are n^{n-1} connected moral TDAGs (or connected transitive subtree acyclic digraphs).

Chapter 3

Structural Learning

3.1 Introduction

The term *structural learning* refers to the task of learning the structure of the GMM from data, i.e. the graphical representation of the model that encodes a set of CI restrictions. Another important aspect of learning GMMs from data is to learn the *parameters* of the model, i.e. the (conditional) probabilities attached to the (in)dependencies that the model specifies. Learning the parameters will not be treated here; the interested reader may consult (Lauritzen, 1996; Heckerman et al., 1995; Spiegelhalter et al., 1996).

One approach to structural learning of GMMs is by using an *oracle* that answers queries on the validity of a given CI restriction in a domain. Starting from the fully connected graph, the results of such queries allow the stepwise removal of edges. The procedure stops when no further CI restrictions can be represented. Some examples of this approach are given in (Spirtes et al., 1993; D. Geiger and Pearl, 1993; de Campos and Huete, 1992, 1997).

We will not use oracle-based methods, but rather those we could call *model-based* methods. In model-based methods, structural learning is a three component setting formed by a *scoring function*, a *traversal operator* and a *search strategy*.

The scoring function is a criterion that in the light of data ranks two or more alternative models. The traversal operator is a function that given an input model, creates a set of "neighboring" models. That is, models that differ to some extent from the input model. The search strategy is an algorithm that applies the traversal operator to a particular subset of models. The members of this particular subset are considered in relation to the ranking that the scoring function provides, and some policy that the search strategy follows.

There are three main model-based approaches to structural learning: simple heuristic search, evolutionary computation and the Markov Chain Monte Carlo (MCMC) method. All have the same goal, learning GMMs from data, but the latter is used for a different purpose than the former two. With simple heuristic search or evolutionary computation, one usually aims at finding the set of models that maximize a certain criterion (fit with the data, predictive power, etc.), with the MCMC method one tries to get an accurate picture of the search space in order to account for the uncertainty of every model considered and derive quantities of interest from the data by averaging across the models.

We will not discuss the approach based in evolutionary computation here but the reader may find related material in (Larrañaga et al., 1996b,a; Etxeberria et al., 1997).

There are different score metrics used as scoring function and we survey in detail those developed from a Bayesian perspective in the next section. Next, we discuss how the score metrics are related to the description of GMMs given in chapter 2. Then we introduce the concept of neighborhood and traversal operators. Next we treat a fundamental notion in the class of DAG Markov models, the notion of *inclusion*, that establishes a particular order among the GMMs. In this section we recall some recent results about this particular aspect that we exploit in the following two sections to devise two efficient new learning algorithms for the DAG Markov model. Finally, we talk about how to incorporate prior knowledge in the learning process, and we finish with some concluding remarks in the last section. Parts of the contents in sections 3.3 to 3.6 have been published in (Giudici and Castelo, 2001b; Kočka and Castelo, 2001; Castelo and Kočka, 2002).

3.2 Bayesian Score Metrics

In chapter 2 we have seen some of the different subtypes of GMMs within the class of DAG Markov models. A first clear distinction of the search spaces within this class is between the search space of those models determined by a directed graph, and the search space of those models determined by an undirected graph.

In either case, the class of DAG Markov models provides a unique factorization of the joint probability distribution of the random variables involved, such that closed formulas for the score metrics are available (Whittaker, 1990; Dawid and Lauritzen, 1993; Heckerman et al., 1995). The derivation of such formulas is possible under a set of assumptions, some of them we enumerate below. A precise introduction of these assumptions as well as the entire development of the score metrics is beyond the scope of this chapter, but the interested reader may find them in the works of Cooper and Herskovits (1992); Heckerman et al. (1995); Dawid and Lauritzen (1993). The most important of those assumptions are the following:

- The given dataset D, from which the score metric is computed, is an independent and identically distributed (iid.) multinomial sample.
- The records in D are *exchangeable*, that is, shuffling the records in D does not alter its joint probability distribution. This assumption implies that the mechanism that generates the data does not change over time.
- The records in D do not contain missing values. One often also says that the dataset D is *complete*.

The dataset D contains records from a set of nominal attributes, which we will treat as a set of categorical random variables $\mathbf{X}_V = \{X_1, \ldots, X_n\}$, where each X_i takes values from a finite domain $\mathcal{X}_i = \{x_{i1}, \ldots, x_{ir}\}$ and $r = r_i$. Thus, an assignment of the k_{th} value to the variable X_i will be denoted as $X_i = x_{ik}$. As in chapter 2, each random variable X_i corresponds to a vertex $i \in V$ in some graph $G = (V, E)$ that determines a GMM denoted by $\mathbf{M}(G)$.

Every record in D corresponds to some element of the product space $\mathcal{X}_V = \times_{i \in V} \mathcal{X}_i$. Analogously, we may consider a subset of variables $\mathbf{X}_A \subseteq \mathbf{X}_V$ which takes values in the corresponding product subspace $\mathcal{X}_A = \times_{i \in A} \mathcal{X}_i$. The product subspace $\mathcal{X}_A$ has

$$q = q_A = \prod_{i \in A} |\mathcal{X}_i| = \prod_{i \in A} r_i, \tag{3.1}$$

elements. Consider an index set $\mathcal{I} = \{1, \ldots, q\}$, where q is defined as in (3.1), such that there is a one-to-one correspondence between $\mathcal{I}$ and $\mathcal{X}_A$. This correspondence allows us to consider the assignment of the k_{th} instantiation of $\mathcal{X}_A$ to $\mathbf{X}_A$, which will be denoted as $\mathbf{X}_A = \mathbf{x}_{Ak}$, and $\mathbf{x}_{Ak} \in \mathcal{X}_A$.

Assume D has N records. Let $\mathbf{x}_A^l$ denote the assignment of the set of variables $\mathbf{X}_A$ in the l_{th} record of D. We will use the term *counts* to denote the number of times a certain assignment occurs throughout the N records of D. More formally, we define the counts N_{Ak} for the set of variables $\mathbf{X}_A$ as the following sum

$$N_{Ak} = \sum_{l=1}^{N} \delta(\mathbf{x}_A^l, \mathbf{x}_{Ak}) \quad \text{where} \quad \delta(\mathbf{x}_A^l, \mathbf{x}_{Ak}) = \begin{cases} 1 & \text{if } \mathbf{x}_A^l = \mathbf{x}_{Ak} \\ 0 & \text{otherwise} \end{cases}$$

In other words, N_{Ak} corresponds to the number of records in D where $\mathbf{X}_A = \mathbf{x}_{Ak}$. Whenever A is clear from the context we will simply use N_k, and δ_{lk} as a shorthand for $\delta(\mathbf{x}_A^l, \mathbf{x}_{Ak})$. The vector of counts $\eta = (N_1, \ldots, N_q)$, determined by a given subset $\mathbf{X}_A \subseteq \mathbf{X}_V$, is usually referred to as the *contingency table* for $\mathbf{X}_A$. From the assumption that D is an iid. multinomial sample it follows that the counts of a contingency table from D

$$\eta = (N_1, \ldots, N_q),$$

follow a multinomial distribution with parameters (N, Θ), where every N_k is a nonnegative integer such that

$$N = \sum_{k=1}^{q} N_k \quad N_k > 0,$$

and Θ is a vector of probabilities, called itself *parameters* too, $\Theta = (\theta_1, \ldots, \theta_q)$, where $0 < \theta_k < 1$, such that

$$\sum_{k=1}^{q} \theta_k = 1.$$

Given that the multinomial distribution is closed under marginalization, let's assume without loss of generality that D refers to the data coming from the set of random variables $\mathbf{X}_A$ for some $\mathbf{X}_A \subseteq \mathbf{X}_V$. The parameters of the multinomial distribution specify the probability for a particular record of D, corresponding to the k_{th} entry in the contingency

table. In particular, the assumption about the exchangeability of the records in D implies that for a fixed set of parameters Θ, the records are independent, from which follows that

$$p(\mathbf{x}_A^l = \mathbf{x}_{Ak} | \mathbf{x}_A^1, \ldots, \mathbf{x}_A^{l-1}, \Theta) = \theta_k. \tag{3.2}$$

Therefore, the likelihood for a given dataset D with N records, of a set of parameters Θ, is

$$p(D|\Theta) = \prod_{l=1}^{N} \prod_{k=1}^{q} \theta^{\delta_{lk}} = \prod_{k=1}^{q} \theta_k^{N_k}. \tag{3.3}$$

Let η denote the contingency table of a dataset D of N records, parametrized by Θ. The probability function (pf.) of the multinomial distribution of D is

$$p(\eta|N, \Theta) = \frac{N!}{N_1! \cdots N_q!} \theta_1^{N_1} \cdots \theta_q^{N_q}.$$

In this chapter, as in the rest of this thesis, we use $p(\cdot)$ to denote either the probability of an event, a joint probability distribution, a probability function (pf.) or a probability density function (pdf.). Its particular interpretation will be clear from the context. The score metrics we survey here use a Bayesian approach in order to score a model. In this context, the uncertainty about the parameters Θ is handled through a *prior distribution* which, in combination with the contingency table, results in a *posterior distribution* using Bayes' theorem. Thus, it is necessary to make an assumption about the type of prior distribution for the parameters Θ. In concrete, as we shall see below, one is interested in using what is known as a *conjugate prior*. A prior distribution is *conjugate* if, when combined with the data, the posterior has the same functional form.

The usual choice, as made by (Dawid and Lauritzen, 1993; Cooper and Herskovits, 1992; Heckerman et al., 1995), is to assume that the parameters Θ follow a Dirichlet distribution, which is conjugate with a complete multinomial sample. A Dirichlet distribution over the vector of parameters Θ, requires the specification of a vector of *hyperparameters*, also known as *pseudo-counts*, $\vartheta = (N'_1, \ldots, N'_q)$, where $N'_k > 0$, and its pdf. is

$$p(\Theta|\vartheta) = \frac{\Gamma(N')}{\Gamma(N'_1) \cdots \Gamma(N'_q)} \theta_1^{N'_1 - 1} \cdots \theta_q^{N'_q - 1} \quad \text{where} \quad N' = \sum_{k=1}^{q} N'_k. \tag{3.4}$$

The somewhat recurrent concept of distribution of a probability distribution, as we have here in $p(\Theta|\vartheta)$, is commonly known in the statistics literature as a *second order distribution*, or a *law* for Θ (Dawid and Lauritzen, 1993), which in this particular case is a *hyper-Dirichlet* law. Analogously, the term *prior law* (Dawid and Lauritzen, 1993) denotes the prior distribution of a probability distribution.

More detailed descriptions of the use and properties of the multinomial and Dirichlet multivariate distributions may be found in the book by DeGroot (1970). One of such properties is the simple expression that gives the expectation for a particular assignment $\mathbf{X}_A = \mathbf{x}_{Ak}$, which is

$$E(\theta_k) = \frac{N'_k}{N'}.$$

By using Bayes' theorem, the posterior distribution of Θ, given a dataset D, is

$$p(\Theta|D, \vartheta) = c \cdot p(D|\Theta)p(\Theta|\vartheta), \tag{3.5}$$

where c is a normalization constant. By replacing in expression (3.5) the corresponding terms specified in (3.3) and (3.4), we see explicitly that the Dirichlet prior is conjugate under multinomial sampling,

$$p(\Theta|D, \vartheta) = c \prod_{k=1}^{q} \theta_k^{N'_k + N_k - 1}, \tag{3.6}$$

where c is again a normalization constant. It follows then that the posterior expectation of θ_k given D is (DeGroot, 1970)

$$E(\theta_k|D) = \frac{N'_k + N_k}{N' + N}. \tag{3.7}$$

Since the vector of counts $\eta = (N_1, \ldots, N_q)$ embodies all the contribution of the data D to the posterior of the parameters Θ and their expectations, the counts in η are often also referred as the *sufficient statistics* for D. More formally, one says that η *is sufficient* for Θ because, conditionally on η, any other statistic κ is independent of Θ. Let's emphasize that in order to update the parameters Θ in the light of data, it suffices to update its hyperparameters ϑ by adding up the sufficient statistics.

For the purpose of model comparison, we are interested in computing the marginal probability of the data unconditional from any particular value of the parameters Θ. In order to do so, we should marginalize over Θ,

$$p(D|\vartheta) = \int_{\Theta} p(D|\Theta)p(\Theta|\vartheta)d\Theta. \tag{3.8}$$

Replacing the likelihood term $p(D|\Theta)$ in (3.8) by (3.3) and the prior term $p(\Theta|\vartheta)$ by (3.4) we obtain:

$$p(D|\vartheta) = \frac{\Gamma(N')}{\Gamma(N'_1)\cdots\Gamma(N'_q)} \int_{\theta_k} \cdots \int \theta_1^{N'_1 + N_1 - 1} \cdots \theta_q^{N'_q + N_q - 1} d\theta_1 \cdots d\theta_q. \tag{3.9}$$

The multiple integral in (3.9) is solved by taking into account that integral of the pdf. (3.4) over all proper values of theta should be one:

$$\int_{\Theta} p(\Theta|\vartheta)d\Theta = \frac{\Gamma(N')}{\Gamma(N'_1)\cdots\Gamma(N'_q)} \int_{\theta_k} \cdots \int \theta_1^{N'_1-1}\cdots\theta_q^{N'_q-1} d\theta_1\cdots d\theta_q = 1. \tag{3.10}$$

Therefore, by isolating the multiple integral in (3.10) and replacing its value in (3.9) we will obtain the closed expression for $p(D|\vartheta)$. However, for the purpose of providing full insight into the expression of the marginal $p(D|\vartheta)$ we will carry out the marginalization over θ in a stepwise manner through the dataset D. Let's consider the dataset D as the collection of records $D_1, \ldots, D_N$. From the rules of probability it follows

$$p(D|\vartheta) = \prod_{l=1}^{N} p(D_l|D_1, \ldots, D_{l-1}, \vartheta).$$

Now, we marginalize over Θ the conditional probabilities,

$$p(D|\vartheta) = \prod_{l=1}^{N} \int_{\Theta} p(D_l|D_1, \ldots, D_{l-1}, \Theta)p(\Theta|D_1, \ldots, D_{l-1}, \vartheta)d\Theta.$$

Note that for every record there is only one k for which $\delta_{lk} = 1$, thus each record D_l has a unique predictive probability θ_k, which we may express in the following way,

$$p(D|\vartheta) = \prod_{l=1}^{N} \int_{\theta_k} \cdots \int \left(\prod_{k=1}^{q} \theta_k^{\delta_{lk}}\right) p(\theta_1, \ldots, \theta_q|D_1, \ldots, D_{l-1}, \vartheta)d\theta_1\cdots d\theta_q.$$

When $\delta_{lk} = 1$ the previous multiple integral will result in the expectation of θ_k with respect to the conditional density $p(\theta_1, \ldots, \theta_q|D_1, \ldots, D_{l-1}, \vartheta)$. Again, we know we will obtain one single posterior expectation for each record,

$$p(D|\vartheta) = \prod_{l=1}^{N}\prod_{k=1}^{q} E(\theta_k|D_1, \ldots, D_{l-1}, \vartheta)^{\delta_{lk}}.$$

Let's expand these products through the indexes l and k and we will obtain:

$$\begin{aligned} p(D|\vartheta) \quad = \quad & E(\theta_1|\vartheta)^{\delta_{11}} E(\theta_2|\vartheta)^{\delta_{12}} \cdots E(\theta_q|\vartheta)^{\delta_{1q}} \cdot \\ & E(\theta_1|D_1, \vartheta)^{\delta_{21}} E(\theta_2|D_1, \vartheta)^{\delta_{22}} \cdots E(\theta_q|D_1, \vartheta)^{\delta_{2q}} \cdot \\ & E(\theta_1|D_1, D_2, \vartheta)^{\delta_{31}} E(\theta_2|D_1, D_2, \vartheta)^{\delta_{32}} \cdots E(\theta_q|D_1, D_2, \vartheta)^{\delta_{3q}} \cdot \\ & \vdots \\ & E(\theta_1|D_1, \ldots, D_{N-1}, \vartheta)^{\delta_{N1}} \cdots E(\theta_q|D_1, \ldots, D_{N-1}, \vartheta)^{\delta_{Nq}} \end{aligned}$$

Now, let's replace the posterior expectations by expression (3.7),

$$
\begin{aligned}
p(D|\vartheta) \quad = \quad & \left(\frac{N'_1}{N'}\right)^{\delta_{11}} \cdot \left(\frac{N'_2}{N'}\right)^{\delta_{12}} \cdots \left(\frac{N'_q}{N'}\right)^{\delta_{1q}} \cdot \\
& \left(\frac{N'_1+\sum_{l=1}^{2-1}\delta_{l1}}{N'+1}\right)^{\delta_{21}} \cdot \left(\frac{N'_2+\sum_{l=1}^{2-1}\delta_{l2}}{N'+1}\right)^{\delta_{22}} \cdots \left(\frac{N'_q+\sum_{l=1}^{2-1}\delta_{lq}}{N'+1}\right)^{\delta_{2q}} \cdot \\
& \left(\frac{N'_1+\sum_{l=1}^{3-1}\delta_{l1}}{N'+2}\right)^{\delta_{31}} \cdot \left(\frac{N'_2+\sum_{l=1}^{3-1}\delta_{l2}}{N'+2}\right)^{\delta_{32}} \cdots \left(\frac{N'_q+\sum_{l=1}^{3-1}\delta_{lq}}{N'+2}\right)^{\delta_{3q}} \cdot \\
& \vdots \\
& \left(\frac{N'_1+N_1-1}{N'+N-1}\right)^{\delta_{N1}} \cdot \left(\frac{N'_2+N_2-1}{N'+N-1}\right)^{\delta_{N2}} \cdots \left(\frac{N'_q+N_q-1}{N'+N-1}\right)^{\delta_{Nq}}
\end{aligned}
$$

Considering again that for each record l there is only one value k for which $\delta_{lk} = 1$ we can reorganize the products in the following way:

$$
p(D|\vartheta) = \frac{\prod_{k=1}^{q} N'_k \cdot (N'_k+1) \cdot (N'_k+2) \cdots (N'_k+N_k-1)}{N' \cdot (N'+1) \cdot (N'+2) \cdots (N'+N-1)}. \tag{3.11}
$$

Finally, by multiplying numerator and denominator in expression (3.11) by $\Gamma(N')$ and $\prod_k^q \Gamma(N'_k)$, we obtain the closed expression for the marginal probability of the data D:

$$
p(D|\vartheta) = \frac{\Gamma(N')}{\prod_{k=1}^{q} \Gamma(N'_k)} \cdot \frac{\prod_{k=1}^{q} \Gamma(N'_k+N_k)}{\Gamma(N'+N)}.
$$

Recall that at the beginning of the section we assumed without loss of generality that D represents the data generated by some subset of random variables $\mathbf{X}_A \subseteq \mathbf{X}_V$. This implies that the sufficient statistics are computed by marginalizing over $\mathbf{X}_V \backslash \mathbf{X}_A$, thus in order to be precise, we will rewrite the formula as

$$
p_A(D|\vartheta) = \frac{\Gamma(N')}{\Gamma(N'+N)} \prod_{k=1}^{q} \frac{\Gamma(N'_k+N_k)}{\Gamma(N'_k)}, \tag{3.12}
$$

where $p_A(D|\vartheta)$ indicates explicitly that the counts N_k are marginally computed from a larger contingency table for $\mathbf{X}_V$. In general, we use the notation $p_S(D)$ to indicate marginalizing D over $\mathbf{X}_V \backslash \mathbf{X}_S$.

Expression (3.12) does not contain or assume the existence of any CI restriction among the random variables in $\mathbf{X}_A$. That expression provides the marginal probability of $\mathbf{X}_A$ under the saturated model for $\mathbf{X}_A$. In the next subsections we will see how suitable assumptions about (conditional) independencies in the whole set of variables $\mathbf{X}_V$, in combination with the marginal probability of the data in (3.12), provide the score metrics for model comparison among GMMs using the following conventions.

From a Bayesian perspective we want to choose a model M according to its posterior probability given some dataset D,

$$
p(M|D) \propto p(D|M)p(M). \tag{3.13}
$$

In expression (3.13) we have used Bayes' theorem to see that this posterior is proportional to the likelihood of the data D given the model M, times the prior probability of M. A Bayesian score metric of a model M against some dataset D is the logarithm of the likelihood times the prior:

$$\mathrm{sc}(M; D) = \log\left[p(D|M)p(M)\right].$$

Therefore, what we are going to see in the next subsections, technically speaking, is the specification of the likelihood term $p(D|M)$ for DEC and DAG models. Furthermore, the Bayesian approach we are following requires a specification of each of the hyperparameters contained in $\vartheta = (N'_1, \ldots, N'_q)$. Since ϑ may be very large easily, this assignment should be carried out in a systematic way. At the same time, the set of possible assignments of ϑ are heavily constrained by the properties we need to retain from the family of probability distributions we intend to work with. This latter aspect will be treated by the specification of *prior laws* for the corresponding model. The prior probability of the model $p(M)$ in conjunction with the hyperparameters ϑ express our prior knowledge in the learning process. In this thesis we will not treat the problem of specifying $p(M)$ and we will assume a uniform distribution over the space of models. The interested reader may consult (Castelo and Siebes, 2000; Angelopoulos and Cussens, 2001) for a survey in some of the existing methods to elicitate $p(M)$.

3.2.1 Score metric for DEC models

Given an undirected graph $G = (V, E)$, a pair of subsets $A, B \subset V$ form a *decomposition* of G if $V = A \cup B$, $A \cap B$ induces a complete subgraph on G, and $A \cap B$ separates A from B in G.

Let $G = (V, E)$ be an undirected chordal graph. A probability distribution $P = p(\mathbf{X}_V)$ over a set of random variables $\mathbf{X}_V$ is called *Markov* (Dawid and Lauritzen, 1993) over G if for any decomposition of V in A and B,

$$A \perp\!\!\!\perp B | A \cap B [P],$$

where we are using A and B as a shorthand for $\mathbf{X}_A$ and $\mathbf{X}_B$. Given two distributions, $p(A)$ of A and $p(B)$ of B, we say that they are *consistent* (Dawid and Lauritzen, 1993) if they both yield the same distribution over $A \cap B$, i.e. $p_{A\cap B}(A) = p_{A\cap B}(B)$. The consistency of the two distributions $p(A)$ and $p(B)$ implies that (Dawid and Lauritzen, 1993):

1. there exists a unique distribution $P = p(A \cup B)$ such that, its marginal distributions $p_A(A \cup B) = \sum_B p(A \cup B)$ and $p_B(A \cup B) = \sum_A p(A \cup B)$ coincide with the original distributions, i.e. $p_A(A \cup B) = p(A)$ and $p_B(A \cup B) = p(B)$.

2. $A \perp\!\!\!\perp B | A \cap B [P]$.

A probability distribution $p(\mathbf{X}_V)$ is Markov over a chordal graph $G = (V, E)$ if and only if for any decomposition $A, B \subset V$ (Dawid and Lauritzen, 1993):

$$p(\mathbf{X}_V) = \frac{p_A(\mathbf{X}_V)\ p_B(\mathbf{X}_V)}{p_{A\cap B}(\mathbf{X}_V)}. \tag{3.14}$$

The chordal graph G can be represented by the set of all cliques $\mathcal{C}$ in G. The cliques in $\mathcal{C}$ can be lined up in a *perfect* order, e.g. $C_1, \ldots, C_m$ (Leimer, 1989). This perfect order on the cliques implies that the subsets $S_i = \{C_1 \cup C_2 \cup \cdots \cup C_{i-1}\} \cap C_i$ are complete in G and they form the set of *separators* of G, denoted $\mathcal{S}$.

From an inductive argument that uses expression (3.14) and this perfect order on the cliques, Dawid and Lauritzen (1993) derive the following theorem.

Theorem 3.1. *Dawid and Lauritzen (1993)*
The unique Markov distribution over a chordal graph $G = (V, E)$ having consistent distributions as its clique marginals is

$$p(\mathbf{X}_V) = \frac{\prod_{C\in\mathcal{C}} p_C(\mathbf{X}_V)}{\prod_{S\in\mathcal{S}} p_S(\mathbf{X}_V)}.$$

From the fact that a DEC Markov model $M = \mathbf{U}(G)$ is determined by a chordal graph G and that the Bayesian approach uses a vector of hyperparameters ϑ to handle the uncertainty concerning the conditional probabilities of the model, the score metric for DEC Markov models is as follows (Dawid and Lauritzen, 1993):

$$\mathrm{sc}(M; D) = \log p(M) + \log \frac{\prod_{C\in\mathcal{C}} p_C(D|\vartheta)}{\prod_{S\in\mathcal{S}} p_S(D|\vartheta)}, \tag{3.15}$$

where $p_C(D|\vartheta)$ and $p_S(D|\vartheta)$ should replaced by expression (3.12).

Prior laws

In a DEC model $\mathbf{U}(G)$, determined by a chordal graph G with clique set $\mathcal{C}$, we need to specify a vector of hyperparameters $\vartheta_C = \{N'_{C1}, \ldots, N'_{Cq}\}$ for every clique $C \in \mathcal{C}$, that determines the corresponding prior laws for $p(\theta_C|\vartheta_C)$. In order to have the clique marginals as consistent distributions in Theorem 3.1, the prior laws for $p(\theta_C|\vartheta_C)$ in every $C \in \mathcal{C}$ must also be consistent (also called *hyperconsistency*).

This constraint implies that for any two cliques C and D such that $C \cap D \neq \emptyset$, and any given lth assignment $\mathbf{x}_{\{C\cap D\}l} \in \mathcal{X}_{\{C\cap D\}}$, their corresponding hyperparameters ϑ_C and ϑ_D must satisfy

$$\sum_{\mathbf{x}_{Ck}\supset\mathbf{x}_{\{C\cap D\}l}} N'_{Ck} = \sum_{\mathbf{x}_{Dk}\supset\mathbf{x}_{\{C\cap D\}l}} N'_{Dk}. \tag{3.16}$$

A way of specifying ϑ_C for every $C \in \mathcal{C}$ correctly, as described by Madigan and York (1995), is as follows. Identify a perfect order of the cliques $\{C_1, \ldots, C_m\}$ (Leimer, 1989).

Specify initial values in ϑ_{C_1}. Then specify ϑ_{C_2} such that condition (3.16) is fulfilled. Each $\vartheta_{C_{i+1}}$ is specified such that it matches condition (3.16) with respect to ϑ_{C_i}.

A computationally cheaper way of obtaining a pairwise consistent hyper-Dirichlet prior law, is to set

$$N'_{Ck} = \frac{1}{|\mathcal{X}_C|}, \tag{3.17}$$

that is, one over the number of levels of the sample subspace of values $\mathcal{X}_C$. The hyperconsistency follows from the fact that the subspaces of values $\mathcal{X}_C$ for each clique $C \in \mathcal{C}$ are induced from a larger space $\mathcal{X}_V$ where $V = \bigcup_{C \in \mathcal{C}} C$.

As part of our prior knowledge, the hyperparameters specified by expression (3.17) imply that we do not have any preference among the levels of each of the marginal contingency tables expressed by the cliques in $\mathcal{C}$. The consequences of such a policy as prior knowledge for the parameters of the model are best understood by examining the variance of one of the parameters $\theta_i \in \theta$. The variance for θ_i indicates how much the mean of θ_i may vary in the light of new data,

$$Var(\theta_i) = \frac{N'_i(\sum_k N'_k - N'_i)}{(N')^2(N'+1)}, \tag{3.18}$$

where as before $N' = \sum_k N'_k$. This expression shows that the larger the values in ϑ are, the smaller the variance, as noted for instance in (Castillo et al., 1997). Since the assignment in (3.17) is the smallest positive hyperconsistent assignment one can set to the hyperparameters in ϑ, it follows immediately that we let the data D determine the shapes of the parameters θ as much as possible. For this reason, this type of assignment is also known as an *uninformative* assignment.

Therefore, to compute the score metric for a DEC model with this uninformative assignment we have to replace the hyperparameters N'_k in expression (3.12) by expression (3.17). Then, it is only left to replace $p_C(D|\vartheta)$ and $p_S(D|\vartheta)$ in expression (3.15) by expression (3.12).

3.2.2 Score metric for DAG models

Let $G = (V, E)$ be a DAG. Let $p(\mathbf{X}_V)$ be a joint probability distribution over $\mathbf{X}_V$. Lauritzen et al. (1990) show that by using the conditional probability distributions of every X_i given $\mathbf{X}_{pa(i)}$ (where $pa(i)$ is the parent set of vertex $i \in V$ in G), $p(\mathbf{X}_V)$ admits a *recursive factorization* as follows:

$$p(\mathbf{X}_V) = \prod_{i \in V} p(X_i|\mathbf{X}_{pa(i)}). \tag{3.19}$$

The fact that the vertices $\{i\} \cup pa(i)$ form a complete subgraph in the moralized version of G, i.e. in G^m, implies this factorization and the more interesting fact that $p(\mathbf{X}_V)$ obeys the global Markov property relative to G^m (Lauritzen et al., 1990).

An important assumption necessary to go forward for this class of GMMs is that of *global independence* (Spiegelhalter and Lauritzen, 1990). Under this assumption we can handle the conditional probability distributions for each random variable independently. This means that we can rewrite the factorization (3.19) in terms of marginal probabilities of the data, similar to what we did for DEC models:

$$p(D|M) = \prod_{i \in V} \frac{p_{\{i\} \cup pa(i)}(D|\vartheta)}{p_{pa(i)}(D|\vartheta)}. \tag{3.20}$$

Where we have used i and $pa(i)$ as a shorthand for X_i and $\mathbf{X}_{pa(i)}$, and again D stands for the dataset generated from $\mathbf{X}_V$. We have assumed the parameters of the model to be Dirichlet distributed such that a vector of hyperparameters ϑ expresses our prior knowledge regarding the parameters. Note that the terms i and $pa(i)$ come from some DAG that determines the GMM M.

The marginals of the data D in the numerator and the denominator can be replaced by the expression we got in (3.12). Since $pa(i) \subseteq \{i\} \cup pa(i)$, we see that the term $p_{pa(i)}(D|\vartheta)$ is actually the marginalization over X_i of $p_{\{i\} \cup pa(i)}(D|\vartheta)$. From this fact it follows that the sums of the hyperparameters N'_k and the counts N_k in $p_{\{i\} \cup pa(i)}(D|\vartheta)$ will sum exactly the same in $p_{pa(i)}(D|\vartheta)$, such that the term $\Gamma(N')/\Gamma(N' + N)$ will cancel in both marginals. Finally, by reorganizing indexes and products, we obtain the expression (Cooper and Herskovits, 1992; Heckerman et al., 1995):

$$p(D|M) = \prod_{i \in V} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}. \tag{3.21}$$

Where the index ijk refers to variable i, level j in the contingency table of the parents $pa(i)$, and level k of variable i. Recall $N'_{ij} = \sum_k N'_{ijk}$. The score metric for a DAG model has thus the following form:

$$\mathrm{sc}(M; D) = \log p(M) + \sum_{i \in V} \log \left[\prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \right]. \tag{3.22}$$

Prior laws

In a DAG model $\mathbf{D}(G)$ determined by a DAG $G = (V, E)$ where every vertex $i \in V$ has a set of parent vertices $pa(i)$, we need to specify a vector of hyperparameters $\vartheta_i = \{N'_{ij1}, \ldots, N'_{ijr_i}\}$ for every vertex $i \in V$, where $r_i = |\mathcal{X}_i|$ (the number of levels of X_i), and j indexes the levels of $\mathbf{X}_{pa(i)}$.

A first straightforward uninformative assignment for ϑ was given by Cooper and Herskovits (1992), where they set $N'_{ijk} = 0$. By replacing these parameters in the likelihood (3.21), we will get the particular form of the likelihood provided in (Cooper and Herskovits, 1992) and which is often referred to in the literature as the K2 metric[1]:

[1] In the same work (Cooper and Herskovits, 1992, p. 342) the authors also elaborate on the assumption of the parameters being Dirichlet distributed and give the likelihood in (3.21).

$$p(D|M) = \prod_{i \in V} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \tag{3.23}$$

However, this likelihood leads to different scores for equivalent DAG models. In our context, where we do not intend to handle causal relationships, we need a score metric that provides equal scores to equivalent DAGs, which is commonly known as an *score equivalent* metric (Heckerman et al., 1995) in the context of DAG Markov models. This requirement constrains, as it happened with hyperconsistency for DEC models, the spectrum of possible assignments for the hyperparameters in ϑ.

Prior laws for DAG models were investigated in depth in the work of Heckerman et al. (1995) and they provide the following informative assignment for the hyperparameters:

$$N'_{ijk} = N' p(X_i = x_k, \mathbf{X}_{pa(i)} = \mathbf{x}_{pa(i)j} |\, G_c), \tag{3.24}$$

where G_c refers to a complete DAG. The idea is that we assess a prior for the density of θ under the complete DAG Markov model, and from such a complete model we derive priors for any other DAG. In order to derive this expression the authors in (Heckerman et al., 1995) make an additional assumption, the assumption of *parameter modularity*. Under this assumption, given two different DAG models $\mathbf{D}(G_1)$ and $\mathbf{D}(G_2)$ over the same set of random variables $\mathbf{X}_V$, such that for some vertex $v \in V$ the parent set $pa(v)$ is the same in G_1 and G_2, the densities for the parameters $p(\theta_v|\vartheta_v)$ are the same in $\mathbf{D}(G_1)$ and $\mathbf{D}(G_2)$. This implies that the densities $p(\theta_v|\vartheta_v)$ depend only in X_v and $\mathbf{X}_{pa(v)}$.

In expression (3.24) the term N' refers to the *equivalent sample size* which is defined by Heckerman et al. (1995) as the users' confidence in the prior density $p(\theta|\, G_c)$, where $N' = 1$ implies no confidence.

This specification of the hyperparameters is called informative because we can express, through the joint density of all the random variables, i.e. the complete DAG model $\mathbf{D}(G_c)$, different priors for parameters in θ. Furthermore, in Heckerman et al. (1995) it is proven that the prior law implied by expression (3.24) makes the metric *score equivalent*. For this reason, the metric in (3.22) with hyperparameters specified as in (3.24) is known as the Bayesian Dirichlet equivalent, or BDe metric.

The constraint of score equivalence plays such an important role that it has been proven in (Geiger and Heckerman, 1995) that under this constraint the only possible choice for the prior density of the parameters is a Dirichlet distribution.

Finally, Buntine (1991) proposed an assignment for the hyperparameters such that it makes the metric in (3.22) score equivalent too. In contrast with the previous one in (3.24), this assignment is uninformative, i.e. the parameters in θ are uniformly distributed, and is as follows:

$$N_{ijk} = \frac{1}{q_i \cdot r_i}. \tag{3.25}$$

As noted in Heckerman et al. (1995), this assignment can be seen as a special case of the assignment in (3.24) where the levels of the joint density for the complete DAG

model $\mathbf{D}(G_c)$ are equally probable, and the equivalent sample size N' is set to 1. The score metric in (3.22) with this prior law is commonly known as the BDeu metric. As for the prior law we have seen for DEC models in (3.17), this uninformative assignment provides the largest variance for the parameters in θ, thus it lets the data determine the parameters θ as much as possible.

In Heckerman et al. (1995) the authors carry out an experimental study where they compare the BDe metric with different sample sizes against the BDeu metric. Their conclusion is that unless we are able to establish prior information that is close to the true model (BDe metric with informative priors) we better assume complete ignorance in our prior information (BDeu metric). In the experiments in sections 3.5 and 3.6 we use the BDeu metric.

3.3 Neighborhoods and Traversal Operators

As we have seen, the class of DAG Markov models allows the derivation of closed formulas that provide the efficient estimation of the goodness of fit of a model against a given dataset. For other classes of GMMs this is not the case, e.g. the class of UG Markov models, and iterative procedures must be used. This means that much more time is needed in order to score a single model.

The advantage thus, of using the DAG model, is that we are able to score and compare many models in a reasonable time, and therefore devise some *model search* (model selection) procedure that works in a systematic way in order to find the set of models that best meet our desired criterion.

Furthermore, the score metrics in the class of DAG Markov models are not only computationally cheap but they can be split up in independent terms such that we can score any given model by performing only local computations. In Heckerman et al. (1995) such score metrics are defined as *decomposable score metrics*, since the global measure they compute can be written as a product of local measures, where each local measure is a function of one vertex and its parent set only. The notion of computing the score incrementally is related to the concept of *neighborhood*. We say that two models are neighbors if they differ to some bounded extent. In the case of GMMs the differences among models lie in the set of edges of the corresponding graphs. As we shall see below, the extent to which GMMs usually differ is bounded by one single adjacency difference.

It is also possible to devise a neighborhood of models that differ in two or more adjacencies, but this choice will increase the size of the neighborhood exponentially (in the number of adjacencies that differ). Some research in this direction can be found in (Xiang et al., 1996).

From the concept of neighborhood, follows the concept of *traversal operator*. A traversal operator is a function that given an input model returns a neighborhood of models. By repeatedly applying the traversal operator we can *traverse* the search space of models, hence the name.

Another important concept in this context, is that of *legal move* between GMMs. Let $\mathbf{M}(G)$ be a GMM determined by a graph G such that $\mathbf{M}(G)$ belongs to certain class $\mathcal{M}$ of GMMs. A *legal move* for $\mathbf{M}(G)$ is a transformation of G into another graph G' such that the GMM determined by G' remains in the same class, i.e. $\mathbf{M}(G') \in \mathcal{M}$. For instance, as we shall see later, a legal move for DAG Markov models is an addition of a directed arc

that does not introduce a directed cycle.

Legal moves require a characterization of the circumstances under which one can add or remove a directed or undirected edge or set of edges. When these characterizations were not available, the procedure to apply a traversal operator was to transform the input graph and then check, *a posteriori*, whether the graph remained in the desired class. This methodology is more inefficient than one that uses legal moves and luckily, legal moves have already been characterized for the class of DAG Markov models.

As we shall see below, the characterizations of legal moves use specific (additional) representations of the graph that determine the model, in order to foresee whether a move is legal or not. For DEC models this additional representation consists of junction trees, and for DAG models it consists of ancestor matrices. These representations need to be updated at each moment the search strategy selects (*moves to*) a new subset[2] of models to continue the search. The computational cost of this latter step should be bounded such that the use of legal moves still pays back in comparison with the traditional *a posteriori* tests of the validity of neighbors.

Therefore, to provide more concise specifications afterwards, we further divide the *traversing* step into *proposing* and *moving* (Giudici and Castelo, 2001b) as follows:

1. **Proposing**. Apply the traversal operator to the current subset of models using the characterizations of legal moves to efficiently create a neighborhood of models.

2. **Moving**. Update the current subset of models to those selected by the search strategy to continue searching. Update their additional representations in order to foresee legal moves afterwards.

3.3.1 The search space of DEC Markov models

In chapter 2, DEC Markov models were defined as those determined by chordal graphs. Thus, given a DEC model $\mathbf{U}(G)$, the adjacencies in G either have no edge or have an undirected edge, such that neighboring models are determined by a graph with one arc more or one arc less. Yet, not all additions or removals of undirected edges are *legal*. A legal move in the class of DEC Markov models must keep the graph that determines the model, chordal.

Proposing

Let $\mathbf{U}(G)$ be a DEC Markov model determined by a chordal graph $G = (V, E)$, where V is its vertex set and E its edge set. Recall from chapter 2 that a *path* between two vertices $a, b \in V$ is a sequence of vertices $a = v_0, \ldots, b = v_k$ such that $(v_i, v_{i+1}) \in E$. A *connected component* in G is an induced subgraph $G_S = (V_S, E_S)$ in which there are paths between all pairs of vertices in V_S. The legal moves for the class of DEC Markov models are defined as follows:

- Addition (Giudici and Green, 1999).
  Let $v_i, v_j \in V$ be two non-adjacent vertices. The addition of an edge between v_i and v_j is legal if and only if either:

[2] Very often a singleton.

  - v_i and v_j belong to different connected components.
  - v_i and v_j belong to the same connected component, and there exists a path in the associated junction forest, between a clique C_i, that contains v_i and a clique C_j, that contains v_j, such that $C_i \cap C_j$ is a separator[3] in that path.

- Removal (Frydenberg and Lauritzen, 1989).
  Let $v_i, v_j \in V$ be two adjacent vertices. The removal of the edge between v_i and v_j is legal if and only if this edge belongs to a single clique.

Moving

To *move* in the context of DEC models means to update the associated junction forest according to the new organization of the cliques after the graph has been updated. For this purpose, we have used the maximum cardinality search algorithm from Tarjan and Yannakakis (1984) in the particular form given by Cowell et al. (1999). We may see the algorithm in Figure 3.1 and its complexity is of order $\mathcal{O}(|V| + |E|)$. An alternative to rebuild the junction forest entirely, is to update only those parts of the junction forest that change by additions or removals of edges performed in the chordal graph. Giudici and Green (1999) have characterized which parts of the junction forest change after an addition or removal of an edge in the corresponding chordal graph. Deshpande et al. (2001) have provided an algorithm that using such characterization updates the junction forest and its complexity is of order $\mathcal{O}(|V|^2)$.

The algorithm in Figure 3.1 takes as an argument a chordal graph g and returns a tree t. Following object oriented notation, this algorithm assumes that g and t implement the following methods:

- $g.bd(v)$ Given a vertex v, this method returns the set of vertices adjacent to v in g.
- $g.complete(s)$ Given a set of vertices s, this method returns *true* if s forms a complete subgraph in g.
- $t.addclique(s)$ Given a set of vertices s, this method adds to the tree t a node with s.

In Figure 3.2 we may see an example of a chordal graph and its corresponding junction tree which may be built using the algorithm in Figure 3.1.

A detailed description of the junction tree representation of a chordal graph is beyond the scope of this section, but the interested reader may find precise explanations in (Jensen and Jensen, 1994; Cowell et al., 1999).

Neighborhoods

In the class of DEC Markov models we consider only one single concept of neighborhood, the one formed by all chordal graphs with one edge more and one edge less, such that the resulting undirected graph is chordal. We will refer to this neighborhood as the *1M1L neighborhood*, denoted by $\mathcal{N}_{1\text{M}1\text{L}}(G)$ for any given chordal graph G for which the mentioned neighborhood is created.

[3]Note that the term *separator* refers here to the intersection of two adjacent nodes in the corresponding junction tree and not to the general concept of *separation* in graphs.

```
algorithm build_junction_forest(dag g=(V,E)) returns tree
01   bool ischordal = true
02   tree t = ∅
03   int i = 0, c[|V|], n[|V|]
04   set Π[|V|], L = ∅
05   for v ∈ V do c[v] = 0 endfor
06   while L ≠ V and ischordal do
07       set U = V\L
08       n[i] = arg max_{v∈U} c[v]
09       Π[i] = g.bd(n[i]) ∩ L
10       if ¬g.complete(Π[i]) then
11           ischordal =false
12       else
13           if i > 0 or |V| = 1 then
14               if |Π[i]| < |Π[i−1]| + 1 or i = |V| − 1 then
15                   t.addclique(Π[i−1] ∪ n[i−1])
16               endif
17           endif
18           for w ∈ g.bd(n[i]) ∩ U do
19               c[w] = c[w] + 1
20           endfor
21           L = L ∪ {n[i]}
22           i = i + 1
23       endif
24   endwhile
25   if ¬ischordal then
26       return ∅
27   else
28       return t
29   endif
endalgorithm
```

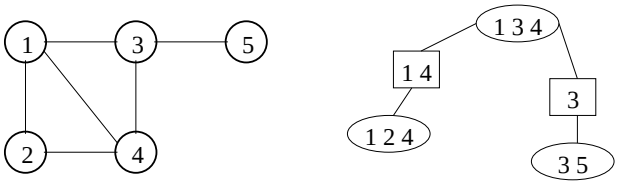
Figure 3.1: Algorithm to create the junction forest from a chordal graph.

3.3.2 The search space of DAG Markov models

Recall from chapter 2 that DAG Markov models were determined by acyclic digraphs, which are directed graphs without directed cycles. Therefore, a legal move in the class of DAG Markov models must keep the directed graph that determines the model acyclic.

Proposing

There are three types of possible moves on a single adjacency: addition, removal and reversal. In order to foresee efficiently legal moves in the class of DAG models, we make use of an *ancestor matrix*. Let $\mathbf{M}(G)$ be a DAG Markov model determined by the DAG $G = (V, E)$, where V is its vertex set and E its edges set. The cardinality of V is n. Let I be the $n \times n$ *incidence matrix* associated to E such that $I(i, j)$ is set to 1 if $v_j \to v_i \in E$, and 0 otherwise. Recall that to say that $v_j \to v_i \in E$ is equivalent to the convention $(v_j, v_i) \in E$



chordal graph junction tree

Figure 3.2: Example of a chordal graph and its corresponding junction tree representation.

and $(v_i, v_j) \notin E$.

An *ancestor matrix* A associated to the edge set E is a $n \times n$ matrix where an entry $A(i, j)$ is set to 1 if E contains a directed path from the vertex v_j to the vertex v_i (or in other words, v_j is ancestor of v_i), and 0 otherwise. The legal moves are characterized in the following way (Giudici and Castelo, 2001b):

- Addition.
  When we consider the addition of an arc $v_j \rightarrow v_i$, this addition is legal if the entry $A(j, i)$ is 0. If the entry $A(j, i)$ was 1, this would mean that v_i is ancestor of v_j and therefore there would exist a directed path from v_i to v_j which becomes a directed cycle when we add $v_j \rightarrow v_i$.

- Removal.
  The removal of an arc cannot introduce any directed cycle, thus it is always legal.

- Reversal.
  The reversal of an arc is a two-step move, in which first we remove the arc, and then we add it in the opposite direction. The first step poses no problem since removals are always legal, but the second step can introduce a directed cycle since it is an addition.

  Let $v_i \rightarrow v_j$ be the arc we want to reverse. If this reversal is not legal, the addition of the arc $v_i \leftarrow v_j$, after the removal, will introduce a directed path from v_i to v_j. We cannot detect that by using the ancestor matrix A in the same way we did in the case of addition because for the v_j row, this matrix describes the ancestors including those it inherits from vertex v_i which now is not part of its *parents* anymore.

  If this directed path from v_i to v_j exists, it means that v_i was ancestor of v_j by some directed path that goes through the parent set of vertices of v_j. Therefore it suffices to check the ancestorship of v_i through every vertex v_k that is parent of v_j, with $k \neq i$.

To decide whether a move is legal using an ancestor matrix is of a cost $\mathcal{O}(1)$ for addition, and $\mathcal{O}(n)$ for reversal. This efficiency is achieved because the computational costs necessary to search for directed paths are turned into the costs necessary to update the ancestor matrix. However, updating the data structures that describe the graph is

part of what we have called here *moving*. In general, the subset of models to which a learning algorithm will *move* will be always substantially smaller than the subset of models initially *proposed* for comparison. In this case the overhead for banning illegal moves is lowered to the minimum in proposing a move.

The optimality of this acyclicity testing overhead for move proposal follows from the fact that, for additions cost cannot be lower than $\mathcal{O}(1)$, i.e. constant. For reversals, a cost of $\mathcal{O}(n)$, i.e. linear in the number of vertices, is the lowest possible because firstly, the current ancestor information of the former sink vertex cannot be used. Secondly, if the reversal introduces a cycle, the former source vertex must have, before the reversal, a directed path to the former sink vertex, through any of the other parents of this former sink, thus ancestorship of the former source should be checked for each of those parents.

Moving

To *move* in the context of DAG models means to update the incidence and ancestor matrices. The updating of the incidence matrix is straightforward. As in the description of *proposing*, let I be the incidence matrix and A the ancestor matrix. When an arc $v_i \rightarrow v_j$ is added, then $I(j,i) = 1$, when the same arc is removed then $I(j,i) = 0$ and finally, when the arc is reversed from $v_i \rightarrow v_j$ to $v_i \leftarrow v_j$, two entries in I need to be updated, namely $I(j,i) = 0$ and $I(i,j) = 1$. The updating of the ancestor matrix A is carried out as follows (Giudici and Castelo, 2001b):

- Addition.
  Let $v_i \rightarrow v_j$ be the added arc. Firstly, we have to set v_i as ancestor in v_j. Secondly, we have to add all the the ancestors of v_i as ancestors of v_j. And, finally, the ancestors of v_j must be added to the ancestors of the descendants of v_j. The descendants of v_j are all those vertices that have v_j as ancestor (i.e. $A(k,j) = 1, k = 1, \ldots, n$). Since we keep an ancestor matrix the latter step will be performed at most $n-1$ times, therefore the cost of the whole operation is of an order linear in the number of vertices, i.e. $\mathcal{O}(n)$.

- Removal.
  Let $v_i \rightarrow v_j$ be the removed arc. Ancestors cannot be propagated as in the previous case, but they have to be rebuilt for the sink vertex v_j and all the descendants of v_j in topological order. In the first place, the ancestors of v_j are set to the parents of v_j, i.e. the updated row of v_j in the incidence matrix I. In the second place the ancestors of every parent vertex of v_j are added to the ancestors of v_j. Now, by inspecting the incidence matrix I we go through all the descendants of v_j in topological order, and for each of them we rebuild the corresponding row in the ancestor matrix in the same way we did with v_j but with respect to the corresponding parent set. Therefore the cost is, at most, of order $\mathcal{O}(n^2)$.

- Reversal.
  Let $v_i \rightarrow v_j$ be the arc to be reversed to $v_i \leftarrow v_j$. In the first place we update a removal $v_i \rightarrow v_j$, and in the second place we update an addition $v_i \leftarrow v_j$.

In the usual *a posteriori* criterion to determine acyclicity of a DAG one searches through the paths of the DAG for directed cycles. As we see now, this search through

the paths is translated here into the updating of an ancestor matrix. The advantage of this latter approach is that it allows us to foresee legal moves (proposing) in an optimal amount of time.

Markov equivalence

In chapter 2 we saw that two DAG Markov models $\mathbf{D}(G)$ and $\mathbf{D}(G')$ determined by two different DAGs G and G' may represent the same restrictions of conditional independence. Recall that these two equivalent DAGs will have some compelled edges and some reversible ones.

The fact that an edge is reversible does not imply that it can be reversed immediately. It just means that at some point it will be possible to reverse it. Those reversible edges that can be indeed reversed at the same time in a given DAG are called *covered* and they were characterized in (Chickering, 1995) in the following way.

Definition 3.1. *Covered Edge (Chickering, 1995)*
Let $G = (V, E)$ be a DAG. An edge $a \rightarrow b \in E$ is covered *in G if $pa(b) = \{a\} \cup pa(a)$.*

Therefore, from the previous notion of equivalence and covered edge, it follows that the reversal of a covered edge will not lead to a different set of conditional independence restrictions. Conversely, the reversal of a non-covered edge will introduce or destroy an immorality and therefore it will lead to a different set of conditional independence restrictions. These two facts were more formally described in (Kočka, 2001) using the following lemma.

Lemma 3.1. *Let $\mathbf{D}(G)$ and $\mathbf{D}(G')$ be two DAG Markov models. The following three conditions are equivalent:*

1. $\mathbf{D}(G) = \mathbf{D}(G')$
2. *G and G' have the same skeleton and contain the same immoralities.*
3. *There exists a sequence $L_1, \ldots, L_n$ of DAGs such that $L_1 = G$ and $L_n = G'$ and L_{i+1} is obtained from L_i by reversing a covered arc in L_i for $i = 1, \ldots, n-1$.*

The equivalence $(1) \Leftrightarrow (2)$ was proven in (Verma and Pearl, 1990; Andersson et al., 1997a), and in the more general framework of CG Markov models in (Andersson et al., 1997b, Theorem 3.1)[4]. The equivalence $(1) \Leftrightarrow (3)$ was proven in (Chickering, 1995; Heckerman et al., 1995; Andersson et al., 1997a).

Neighborhoods

The definition of legal move allows to describe different concepts of neighborhoods in the following way (Kočka and Castelo, 2001):

- **NR** (No Reversals) All DAGs with one arc less and one arc more that does not introduce a directed cycle.

[4]Frydenberg (1990a) provided a proof before but under the additional condition CI5 (see chapter 2).

- **AR** (All Reversals) The NR neighborhood plus all DAGs with one arc reversed that does not introduce a directed cycle.
- **CR** (Covered Reversals) The NR neighborhood plus all DAGs with one covered arc reversed.
- **NCR** (Non-Covered Reversal) The NR neighborhood plus all DAGs with one non-covered arc reversed.

For any given DAG G upon which we built its corresponding neighborhood, the previously described neighborhoods will be noted as $\mathcal{N}_{\mathrm{NR}}(G)$, $\mathcal{N}_{\mathrm{AR}}(G)$, $\mathcal{N}_{\mathrm{CR}}(G)$ and $\mathcal{N}_{\mathrm{NCR}}(G)$, respectively. The NR neighborhood is used in MCMC search by the MC^3 algorithm of Madigan and York (1995). The NR neighborhood may lead easily to local maxima in heuristic search, or to an extremely small probability of reaching the most probable model given the data in MCMC search. This problem may be alleviated by using an AR neighborhood, which is quite common in many other learning algorithms. The CR and NCR neighborhoods are variations of the AR neighborhood that are not intended to enhance the AR neighborhood but we use them here for comparison purposes, as we shall see later.

3.4 Graphical Markov model inclusion

So far, we have seen that structural learning of GMMs involves scoring models and traversing the search space. In order to traverse the search space one devises efficient ways of transforming the graph that determines the GMM to create a neighborhood of candidate models. The scope of models that form a neighborhood depends on which transformations we apply to the given graph, and so far, the transformations do not comply with a specific policy, but just to enable reaching any graph of the search space from any other one.

In this section, we show that there is such a policy which, if followed, it leads to a better performance of the structural learning process. We study the organization of the search space from the perspective of a particular order among the GMMs, the *inclusion order*. Afterwards, we formalize a traversing policy for the search space of GMMs in the form of new concepts of neighborhood. From these new concepts of neighborhood, we introduce in the forthcoming sections 3.5 and 3.6 two new learning algorithms and show the improvement they afford to the structural learning process on synthetic data.

By *graphical Markov model inclusion* we denote a particular relation of partial order among GMMs. A relation of partial order is irreflexive, asymmetric and transitive, and some pairs of elements may not be related, otherwise it would be a *total order*. The intuition behind the graphical Markov model inclusion partial order, or *inclusion order* for short, is that one GMM $\mathbf{M}(G)$ precedes another GMM $\mathbf{M}(G')$ if and only if all the CI restrictions encoded in G are also encoded in G'.

Recall from chapter 2 that a GMM $\mathbf{M}(G)$ determined by some graph G is the set of probability distributions that are Markov over G, i.e. the set of probability distributions that satisfy the CI restrictions encoded by the graph G. Without loss of generality, let's assume we are working with DEC Markov models. A DEC model $\mathbf{U}(G)$ is determined by an undirected chordal graph G.

The complete chordal graph G_c that encodes no CI restriction at all, determines a DEC model $\mathbf{U}(G_c)$ that consists of all possible discrete probability distributions over the corresponding set of random variables, due to the fact that any such distribution is Markov over the complete chordal graph G_c.

On the opposite side, we find the empty chordal graph $G_\emptyset$, with no edges, under which all random variables are marginally independent, such that it encodes all possible CI restrictions among these random variables from the closure of the semi-graphoid axioms. The chordal graph $G_\emptyset$ determines a GMM $\mathbf{U}(G_\emptyset)$ that consists of "only" those discrete probability distributions under which all the random variables are marginally independent. Clearly, the set of probability distributions that are Markov over G_c includes those that are Markov over $G_\emptyset$, therefore

$$\mathbf{U}(G_\emptyset) \subseteq \mathbf{U}(G_c). \tag{3.26}$$

However, the CI restrictions encoded by G_c (none!) are included into those encoded by $G_\emptyset$ (all!), and this latter notion is the one that determines the inclusion order. The notation used in (3.26) might be somewhat counterintuitive with the idea that $\mathbf{U}(G_c)$ precedes $\mathbf{U}(G_\emptyset)$ under the inclusion order. Therefore, we will explicitly express the set of the CI restrictions encoded by a graph G that determines the GMM $\mathbf{M}(G)$ as:

$$\mathbf{M}^I(G) = \{(A, C, B) \,:\, A, B \neq \emptyset \wedge A \perp\!\!\!\perp B | C[G]\}.$$

Note that $\mathbf{M}^I(G)$ is a general definition that applies to every class of GMMs. Now, in order to note the inclusion relationship between the fully restricted DEC model $\mathbf{U}_\emptyset(G_\emptyset)$ and the unrestricted DEC model $\mathbf{U}_c(G_c)$ we may simply write it as:

$$\mathbf{U}^I(G_c) \subseteq \mathbf{U}^I(G_\emptyset).$$

In general, we can decide the inclusion order among DEC models with a rather straightforward graphical criterion, which we may formalize in the following theorem.

Theorem 3.2. *Let $\mathbf{U}(G_1), \mathbf{U}(G_2)$ be two DEC Markov models. We say that the model $\mathbf{U}(G_1)$ is included in the model $\mathbf{U}(G_2)$, and we write it as $\mathbf{U}^I(G_1) \subseteq \mathbf{U}^I(G_2)$, if and only if G_2 is a subgraph of G_1.*

Proof. Necessity. Let's assume that $\mathbf{U}^I(G_1) \subseteq \mathbf{U}^I(G_2)$. Then, every CI restriction read off the chordal graph G_1 can be read off the chordal graph G_2. In this situation, the undirected global Markov property for DEC models implies that every pair of subsets of vertices A, B separated by a third subset S_1 in G_1, must be separated in G_2 by a subset S_2 such that $S_2 \subseteq S_1$. Otherwise the CI restriction in G would not hold in G_2. It follows immediately that G_2 has to be a subgraph of G_1.

Sufficiency. Let's assume that G_2 is a subgraph of G_1. Then, every two subsets of vertices A, B separated by a subset S_2 in G_2, will be separated in G_1 by a subset S_1 such that $S_2 \subseteq S_1$. From this fact and the undirected global Markov property, it follows immediately that every CI restriction in G_1 holds in G_2. □

The notion of inclusion order appeared already in the context of DEC and log-linear models in the works of Havránek (1984) and Edwards and Havránek (1985), although we have adopt here a slightly different formulation and notation. Let's emphasize that in the particular cases we have seen of Markov equivalence (Theorem 2.1) and Markov inclusion (Theorem 3.2), GMMs allow us to make quite strong claims among rather complex interaction models by purely graphical criteria.

In the case of DAG models, there is no cheap graphical criterion to decide the inclusion order, but an operational one, as we shall see now. The first attempt to provide necessary and sufficient conditions to characterize the inclusion order among DAG Markov models was done in (Verma and Pearl, 1988), and in (Kočka, 2001) it is shown that these conditions are necessary but not sufficient. Later, (Meek, 1997) conjectured the following operational criterion to decide DAG Markov model inclusion.

Conjecture 3.1. *Meek's conjecture (Meek, 1997)*
Let $\mathbf{D}(G)$ and $\mathbf{D}(G')$ be two DAG Markov models determined by two DAGs G and G'. The DAG model $\mathbf{D}(G)$ is included in $\mathbf{D}(G')$, $\mathbf{D}^I(G) \subseteq \mathbf{D}^I(G')$, if and only if there exists a sequence of DAGs $L_1, \ldots, L_n$ such that $G = L_1$, $G' = L_n$ and the DAG L_{i+1} is obtained from L_i by applying either the operation of covered arc reversal or the operation of arc removal for $i = 1, \ldots, n$.

Kočka et al. (2001) have proved Meek's conjecture for the particular case in which G and G' differ in at most one adjacency, providing the following operational criterion.

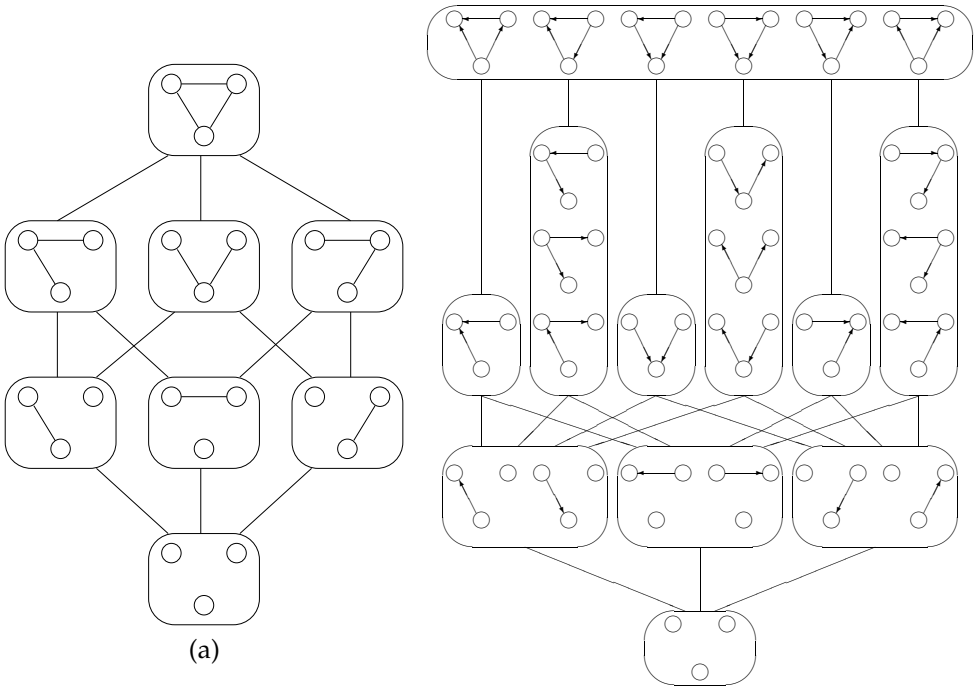
Lemma 3.2. *(Kočka et al., 2001)*
Let $\mathbf{D}(G)$ and $\mathbf{D}(G')$ be two DAG Markov models determined by two DAGs G and G' such that their skeletons differ in at most one adjacency. The DAG Markov model $\mathbf{D}(G)$ is strictly included in $\mathbf{D}(G')$, i.e. $\mathbf{D}^I(G) \subset \mathbf{D}^I(G')$, iff there exists a sequence of DAGs $L_1, \ldots, L_n$ such that $G = L_1$, $G' = L_n$ and the graph L_{i+1} is obtained from L_i by applying the operation of covered arc reversal for $i = 1, \ldots, j-1$, the operation of arc removal for $i = j$ and the operation of covered arc reversal for $i = j+1, \ldots, n-1$ where $j \in \{1, \ldots, n-1\}$.

Very recently, Chickering (2002) has been able to prove Meek's conjecture in its full scope. This fact gives more relevance to the facts that we uncover later about the learning problem.

It is easy to realize that the collection of sets of CI restrictions $\mathbf{M}^I(G) \in \mathcal{M}$ where $\mathcal{M}$ is any given class of GMMs, forms a poset that we can represent by means of a Hasse diagram. In Figure 3.3 we may see this representation for the classes of DEC and DAG Markov models over three variables. Note that because DAG models are organized in classes of equivalence, the nodes in Figure 3.3b may contain more than one graph.

3.4.1 Implications in learning

With the Bayesian score metrics we surveyed at the beginning of this chapter, we aim at recovering the largest set of CI restrictions that our data satisfies and our GMM can represent. Ideally, these CI restrictions should help us to draw conclusions about the mechanism that generated the data, in order to gain a better insight into this data. Therefore, when we say that this or that algorithm did not learn the right model(s) (got stuck in a local maximum), we acknowledge the fact that the learned model(s) either reflect a subset of the CI restrictions of the data which is smaller than what our GMM can represent,



(a) (b)

Figure 3.3: Hasse diagrams for the DEC (a) and DAG (b) Markov models over three variables.

or they reflect CI restrictions that are not actually satisfied by the data generating mechanism. In any of these two scenarios, the learning algorithm performed one or more moves that led to the current *unsatisfactory* model(s). This may be caused by, basically, four reasons (and combinations of them):

1. The data is too sparse.
2. The score metric, incorporating our prior knowledge, is not appropriate.
3. The traversal operator is not appropriate.
4. The search strategy is not appropriate.

The inclusion order is one of the several ways of structuring the search space. But because the inclusion order determines the poset of models of CI restrictions, it may help understanding why the traversal operator is not appropriate. Since *moving*, as part of traversing the search space, is a deterministic operation, blaming the traversal operator implies blaming how the traversal operator *proposes* models. The way models are proposed depends on the concept of neighborhood that is employed.

Kočka and Castelo (2001) introduce the following concept in order to formalize later certain property that neighborhoods must retain.

Definition 3.2. *Inclusion Boundary (Kočka, 2001)*
Let $\mathbf{M}(H), \mathbf{M}(K), \mathbf{M}(L)$ *be three GMMs determined by the graphs* H, K, L. *Let* $\mathbf{M}^I(H) \prec \mathbf{M}^I(L)$ *denote that* $\mathbf{M}^I(H) \subset \mathbf{M}^I(L)$ *and for no* $\mathbf{M}^I(K)$, $\mathbf{M}^I(H) \subset \mathbf{M}^I(K) \subset \mathbf{M}^I(L)$. *The* inclusion boundary *of the GMM* $\mathbf{M}(G)$, *denoted by* $\mathcal{IB}(G)$, *is*

$$\mathcal{IB}(G) = \{\mathbf{M}(H) : \mathbf{M}^I(H) \prec \mathbf{M}^I(G)\} \cup \{\mathbf{M}(L) : \mathbf{M}^I(G) \prec \mathbf{M}^I(L)\}$$

Intuitively, the inclusion boundary of a given GMM $\mathbf{M}(G)$ consists of those GMMs $\mathbf{M}(G_i)$ that induce a set of CI restrictions $\mathbf{M}^I(G_i)$ which *immediately* follow or precede $\mathbf{M}^I(G)$ under the inclusion order. In the Hasse diagrams we see in Figure 3.3, the inclusion boundary for a given node is formed by those nodes adjacent to it.

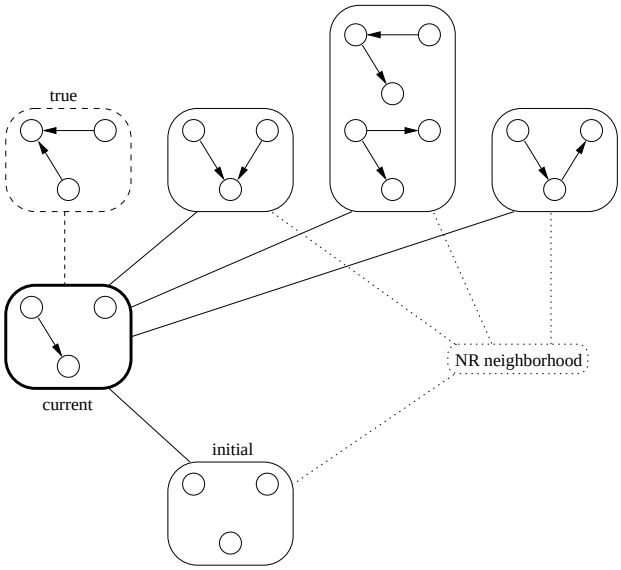


Figure 3.4: Example of getting stuck in a local maximum because the concept of neighborhood employed (NR) does not contain the inclusion boundary.

The following definition of *inclusion boundary condition* establishes a necessary condition that a traversal operator must satisfy in order to avoid local maxima.

Definition 3.3. *Inclusion Boundary Condition (Kočka, 2001)*
A learning algorithm for GMMs satisfies the Inclusion Boundary Condition *if for every GMM determined by a graph* G, *the traversal operator can create a neighborhood* $\mathcal{N}(G)$ *such that* $\mathcal{N}(G) \supseteq \mathcal{IB}(G)$.

Note that some computational performance may be gained if the traversal operator creates a neighborhood $\mathcal{N}(G) = \mathcal{IB}(G)$, since it already satisfies the inclusion boundary condition.

In Figure 3.4 we find a situation, in the context of DAG models, in which the model highlighted with a thick line is our current model. The model highlighted with a dashed line is the one that the data reflects (the *true* model). The rest of the models are the NR neighborhood of the current one. All models around the current one form its inclusion boundary. As we may appreciate, because the current neighborhood does not entirely include the inclusion boundary, it is not possible to reach the *true* model from the current one in a single step. In this situation, it may become very difficult for the learning algorithm to reach the *true* model. Note that we can select easily the current model starting from the empty one, provided that a score equivalent metric would score equally the addition of a single isolated arc in either direction.

In the case of DEC models, we introduced only one concept of neighborhood in subsection 3.3.1, the 1M1L neighborhood, that consists of all undirected graphs with one edge more and one edge less that are chordal. It is rather straightforward to see that such neighborhood coincides with the inclusion boundary for such kind of GMMs.

Theorem 3.3. *Let* $\mathbf{U}(G)$ *be a DEC Markov model. Let* $\mathcal{N}_{1\text{M1L}}(G)$ *be the set of DEC Markov models determined by the 1M1L neighborhood for* G*. Let* $\mathcal{IB}(G)$ *be the inclusion boundary for* $\mathbf{U}(G)$*. It follows that*

$$\mathcal{N}_{1\text{M1L}}(G) = \mathcal{IB}(G).$$

Proof. From the definition of 1M1L neighborhood it follows that all DEC models determined by the immediate subgraphs and supergraphs of G are in $\mathcal{N}_{1\text{M1L}}(G)$. This fact and the characterization of Markov inclusion for DEC models in Theorem 3.2 prove the theorem. □

The fact that the 1M1L neighborhood for DEC models coincide with its inclusion boundary frees the traversal operator of any responsibility of a wrong outcome of the learning process. The inclusion order for DEC models, and its characterization by Theorem 3.2, was used in the works of Havránek (1984) and Edwards and Havránek (1985). The authors used a stepwise model selection procedure that would remove edges starting from the unrestricted model until no removal would improve the fit. By knowing the inclusion order among DEC models, their learning procedure rejects all models that follow any given rejected one in the inclusion order.

For DAG Markov models, the situation is more complex since they are organized in equivalence classes, and the inclusion boundary is defined with respect to these equivalence classes, as we saw for three variables in Figure 3.3b. Therefore, it is clear that the concepts of neighborhoods introduced for this kind of GMM so far (NR, AR, CR and NCR), do not cover the inclusion boundary. In Figure 3.5 we may find this problem more clearly illustrated. In the center we have a circle that represents an equivalence class of DAG models with 16 members determined by graphs G_0 to G_{15}. Surrounding this circle we have other equivalence classes, where those that are shaded represent its inclusion boundary. Given any of the equivalence classes represented in Figure 3.5, its immediate neighbors are reachable by a transformation in a single adjacency of the graph.

Consider the equivalence class in the inclusion boundary that is darker shaded. As it is clear from the figure, only G_9, G_{10} and G_{11} can reach this neighboring equivalence class by a transformation in a single adjacency. Therefore, if our learning algorithm does

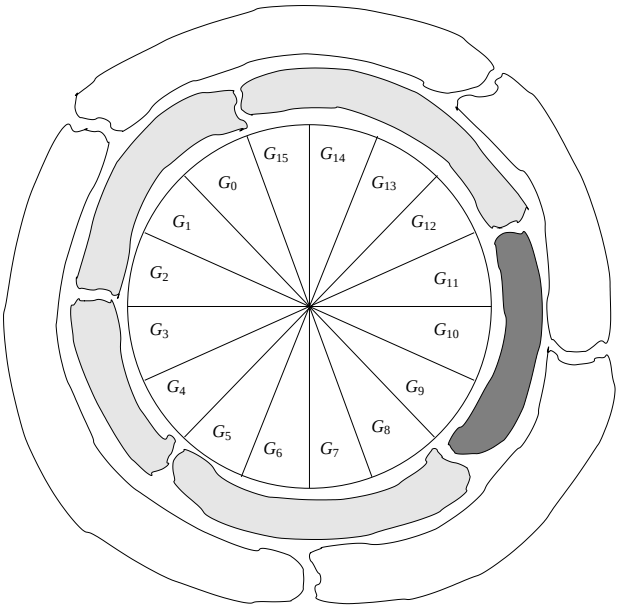


Figure 3.5: Every member $G_0, \ldots, G_{15}$ of an equivalence class cannot reach the entire inclusion boundary (shaded classes) by a transformation of a single adjacency.

not consider any of G_9, G_{10} or G_{11}, it will not be able to reach that neighbor class in a single move.

In order to define a neighborhood that coincides with the inclusion boundary we will take into account the operational criterion defined in the *Meek's conjecture* (Conjecture 3.1) and in particular the partial characterization provided by Lemma 3.2.

Let $\mathbf{D}(G)$ be any given DAG Markov model that belongs to some equivalence class $\mathcal{C} = \{\mathbf{D}(G_1), \ldots, \mathbf{D}(G_n)\}$, i.e. $\mathbf{D}(G_1) = \mathbf{D}(G_2) = \cdots = \mathbf{D}(G_n)$. Consider the following two neighborhoods for $\mathbf{D}(G)$:

- **ENR** (Equivalence class No Reversals) All DAGs with one arc more and one arc less than every $\mathbf{D}(G_i) \in \mathcal{C}$.
- **ENCR** (Equivalence class Non-Covered Reversals) All DAGs with one arc more, one arc less and one non-covered arc reversed than every $\mathbf{D}(G_i) \in \mathcal{C}$.

Now, we will investigate the relationships among the ENR and ENCR neighborhoods, and the other neighborhoods defined earlier.

Lemma 3.3. *Let $\mathbf{D}(G)$ and $\mathbf{D}(G')$ be two DAG Markov models. If G and G' have the same number of edges then either the two DAG Markov models are equivalent, i.e. $\mathbf{D}(G) = \mathbf{D}(G')$, or the two DAG Markov models are not in an inclusion relation, i.e. $\mathbf{D}(G) \not\subset \mathbf{D}(G')$ and $\mathbf{D}(G) \not\supset \mathbf{D}(G')$.*

Proof. We will distinguish three cases. First when the two DAGs, G and G', have different skeletons, second when G and G' have the same skeletons but different immoralities and third when G and G' have the same skeletons and the same immoralities.

In the first case, the two skeletons of G and G' are different but have the same number of edges. This implies that there are two distinct vertices u and v such that are adjacent in G and are non-adjacent in G'. Either $v \in nd(u)$ or $u \in nd(v)$ holds in G' as otherwise there would be a cycle. By the DPMP (see Definition 2.7) there exists a CI restriction between u and v in G'. This CI restriction cannot hold in G according to the d-separation criterion because the u and v are adjacent in G. Thus $\mathbf{D}(G) \not\supset \mathbf{D}(G')$. The same argument applies for $\mathbf{D}(G) \not\subset \mathbf{D}(G')$.

In the second case, G and G' have the same skeletons but different immoralities. Let $u \to w \leftarrow v$ be an immorality formed by three distinct vertices u, v and w which, without loss of generality, is in G but not in G'. This implies that the subgraph induced in G' by u, v and w will be either $u \leftarrow w \leftarrow v$ or $u \to w \to v$ or $u \leftarrow w \to v$ where u and v are non-adjacent as in G.

Either $v \in nd(u)$ or $u \in nd(v)$ holds in G' as otherwise there would be a cycle. By means of the DPMP (see Definition 2.7), there is a CI restriction $u \perp\!\!\!\perp v|C$ in G' for some C where $w \in C$. If $w \notin C$, $u \perp\!\!\!\perp v|C$ would not hold in G' according to the d-separation criterion. The same CI restriction $u \perp\!\!\!\perp v|C$, where $w \in C$, cannot hold in G because $w \in C$ creates an active path between u and v, so they are not d-separated. This leads to $\mathbf{D}(G) \not\supset \mathbf{D}(G')$. Analogously, for some subset C' where $w \notin C'$, the CI restriction $u \perp\!\!\!\perp v|C$ holds in G while it does not hold in G', therefore $\mathbf{D}(G) \not\subset \mathbf{D}(G')$.

The third case follows from Lemma 3.1, which leads to $\mathbf{D}(G) = \mathbf{D}(G')$. □

Lemma 3.4. *Let $\mathbf{D}(G)$ and $\mathbf{D}(G')$ be two DAG Markov models. If $\mathbf{D}(G) \subset \mathbf{D}(G')$ then G' has less edges than G.*

Proof. We will prove this lemma by contradiction. Assume $\mathbf{D}(G) \subset \mathbf{D}(G')$ and the two possibilities: (a) the two DAGs G and G' have the same number of edges and (b) G' has at least one more edge than G.

In (a) the contradiction follows from Lemma 3.3 which forces $\mathbf{D}(G)$ and $\mathbf{D}(G')$ to be either equivalent or not in an inclusion relation.

In (b) the contradiction follows from the fact that there is a pair of vertices u, v which are non-adjacent in G and adjacent in G'. By the DPMP (see Definition 2.7) there is a CI restriction $u \perp\!\!\!\perp v|nd(u)\backslash v$ in G which does not hold in G', i.e. $\mathbf{D}(G) \not\subset \mathbf{D}(G')$. □

The following result discusses some of the relationships among the different concepts of neighborhoods as well as with respect to the inclusion boundary.

Theorem 3.4. *Let $\mathbf{D}(G)$ denote a DAG Markov model. Let $\mathcal{N}_{\mathrm{NR}}(G)$, $\mathcal{N}_{\mathrm{CR}}(G)$, $\mathcal{N}_{\mathrm{NCR}}(G)$, $\mathcal{N}_{\mathrm{AR}}(G)$, $\mathcal{N}_{\mathrm{ENR}}(G)$ and $\mathcal{N}_{\mathrm{ENCR}}$ be the sets of DAG Markov models that form, respectively, the NR, CR, NCR, AR, ENR and ENCR neighborhoods for $\mathbf{D}(G)$. The following statements hold:*

1. *For all G: $\mathcal{N}_{\mathrm{NR}}(G) \subseteq \mathcal{N}_{\mathrm{ENR}}(G) \subseteq \mathcal{IB}(G)$; $\mathcal{N}_{\mathrm{NR}}(G) \subseteq \mathcal{N}_{\mathrm{NCR}}(G) \subseteq \mathcal{N}_{\mathrm{ENCR}}(G)$; $\mathcal{N}_{\mathrm{NR}}(G) \subseteq \mathcal{N}_{\mathrm{CR}}(G) \subseteq \mathcal{N}_{\mathrm{AR}}(G)$; $\mathcal{N}_{\mathrm{NR}}(G) \subseteq \mathcal{N}_{\mathrm{NCR}}(G) \subseteq \mathcal{N}_{\mathrm{AR}}(G)$; $\mathcal{N}_{\mathrm{ENR}}(G) \subseteq \mathcal{N}_{\mathrm{ENCR}}(G)$.*
2. *For all G: $\{\mathcal{N}_{\mathrm{NCR}}(G)\backslash\mathcal{N}_{\mathrm{NR}}(G)\} \cap \mathcal{IB}(G) = \{\mathcal{N}_{\mathrm{AR}}(G)\backslash\mathcal{N}_{\mathrm{CR}}(G)\} \cap \mathcal{IB}(G) = \{\mathcal{N}_{\mathrm{ENCR}}(G)\backslash\mathcal{N}_{\mathrm{ENR}}(G)\} \cap \mathcal{IB}(G) = \emptyset$.*
3. *For all G: $\mathcal{N}_{\mathrm{AR}}(G) \cap \mathcal{IB}(G) = \mathcal{N}_{\mathrm{CR}}(G)$.*
4. *There exists G such that: $\mathcal{N}_{\mathrm{AR}}(G) \not\supseteq \mathcal{IB}(G)$.*

Proof. Statement 1. The part $\mathcal{N}_{\text{NR}}(G) \subseteq \mathcal{N}_{\text{ENR}}(G)$ follows directly from the fact that ENR performs all operations that NR does. The same argument applies to the $\mathcal{N}_{\text{NR}}(G) \subseteq \mathcal{N}_{\text{NCR}}(G)$, $\mathcal{N}_{\text{NCR}}(G) \subseteq \mathcal{N}_{\text{ENCR}}(G)$, $\mathcal{N}_{\text{NR}}(G) \subseteq \mathcal{N}_{\text{CR}}(G)$, $\mathcal{N}_{\text{CR}}(G) \subseteq \mathcal{N}_{\text{AR}}(G)$, $\mathcal{N}_{\text{NCR}}(G) \subseteq \mathcal{N}_{\text{AR}}(G)$ and $\mathcal{N}_{\text{ENR}}(G) \subseteq \mathcal{N}_{\text{ENCR}}(G)$.

The relationship $\mathcal{N}_{\text{ENR}} \subseteq \mathcal{IB}(G)$ follows from Lemmas 3.1, 3.2 and 3.4. The first lemma guarantees us that the ENR neighborhood is created by transforming every member of an equivalence class. The second lemma proves that the DAG Markov model from which the ENR neighborhood is created, precedes all DAG Markov models in the ENR neighborhood under the inclusion order. The third lemma shows that, under the inclusion order, there are no DAG Markov models *in between* the one from which the ENR neighborhood is created, and those from the ENR neighborhood, i.e. $\mathcal{N}_{\text{ENR}} \subseteq \mathcal{IB}(G)$.

Statement 2. The DAG Markov models in the difference sets $\{\mathcal{N}_{\text{NCR}}(G) \backslash \mathcal{N}_{\text{NR}}(G)\}$, $\{\mathcal{N}_{\text{AR}}(G) \backslash \mathcal{N}_{\text{CR}}(G)\}$ and $\{\mathcal{N}_{\text{ENCR}}(G) \backslash \mathcal{N}_{\text{ENR}}(G)\}$ are created by the reversal of a non-covered arc in G. This statement says that, for any given DAG Markov model $\mathbf{D}(G)$, if we reverse a non-covered arc of G obtaining a new DAG G', then $\mathbf{D}(G') \notin \mathcal{IB}(G)$. We prove this as follows. If an arc is not covered in G, its reversal either introduces or destroys an immorality in G' (see Definition 3.1) that yields a non-equivalent model (see Lemma 3.1), i.e. $\mathbf{D}^I(G) \neq \mathbf{D}^I(G')$. Since the number of edges remains the same, by Lemma 3.3, $\mathbf{D}^I(G) \not\subset \mathbf{D}^I(G')$, and therefore $\mathbf{D}^I(G) \not\subseteq \mathbf{D}^I(G')$.

Statement 3. It follows from statement 2 and the fact that $\mathcal{N}_{\text{AR}}(G) = \mathcal{N}_{\text{CR}}(G) \cup \mathcal{N}_{\text{NCR}}(G)$.

Statement 4. Figure 3.5 illustrates this statement. □

The intuition behind the previous theorem is that the ENR neighborhood covers a larger part of the inclusion boundary than any of the previously introduced neighborhoods. In particular, the ENR will coincide with the inclusion boundary under the next circumstance.

Theorem 3.5. *Let $\mathbf{D}(G)$ be a DAG Markov model. If Meek's conjecture holds, then it follows that*

$$\mathcal{N}_{\text{ENR}}(G) = \mathcal{IB}(G).$$

Proof. From the characterization of the inclusion order provided by Meek's conjecture it follows that the inclusion boundary are all DAGs with one arc more and one arc less than every member of the equivalence class of a given DAG, which coincides with the definition of ENR neighborhood. □

As it follows from its definition, the ENCR neighborhood contains the ENR neighborhood plus other models that are not part of the inclusion boundary. Assuming that Meek's conjecture holds, the ENR neighborhood retains already the desired property of containing the inclusion boundary. Nevertheless, we shall see later on the experiments, a variation of the ENCR neighborhood seems to be actually very useful, hence we formalize now this neighborhood.

A straightforward observation is that both the ENR and ENCR neighborhoods are not computationally efficient to handle. More concretely, the effort to enumerate the members of an equivalence class is prohibitive since there is no *cheap* graphical characterization of those members.

However, in (Kočka and Castelo, 2001) the authors conjecture that because, as we saw in chapter 2, the ratio of DAGs per equivalence class seems to be bounded by some constant, it might suffice to *simulate* somehow the ENR neighborhood. In particular, Kočka and Castelo (2001) introduce the *repeated covered arc reversal* algorithm, or RCAR algorithm, that allows to reach any member of the equivalence class with certain probability. We may see the RCAR algorithm in Figure 3.6.

```
algorithm g.rcar(int r) is
01    int rr = rnd(0,r)
02    for i = 0 to rr do
03        vector ce = g.covered_edges()
04        int j = rnd(0,ce.size()-1)
05        edge e = ce[j]
06        g.reverse_edge(e)
07    endfor
endalgorithm
```

Figure 3.6: RCAR algorithm implemented as a method for an object g that embodies a DAG and implements a method that returns a vector of the covered edges and an another method that reverses a given edge.

The algorithm in Figure 3.6 takes a constant r as parameter and iterates some random number of times between 0 (no iteration) and r. At each iteration, it picks at random a covered arc and reverses it. Lemma 3.1 guarantees us that the RCAR algorithm reaches any member of the equivalence class with a positive probability for a *sufficiently large* maximum number r of iterations. The bounded ratio of DAGs per equivalence class suggests that a small number between 4 and 10 should be *sufficiently large*.

Note that when the number of undirected edges in the corresponding essential graph is lower or equal to the number of iterations of RCAR, then RCAR is able to reach any DAG Markov model in its equivalence class. Gillispie and Perlman (2001) show that the distribution of the sizes of the equivalence classes represented by essential graphs, follows a very particular pattern. In particular, they make the following observation (Gillispie and Perlman, 2001):

> The pattern of the distribution shows that certain sizes appear more frequently than others. In particular, larger compound numbers occur more often than larger prime numbers. This is probably due to separate sets of undirected edges in the essential graph acting independently to produce class sizes that are products of the sizes of their independent components.

In a nutshell, essential graphs, whose graph is disconnected, with many connected components represent equivalence classes with a large number of members. However, the size of these connected components is inversely proportional to the number of them.

This fact allows RCAR to reach a substantial amount of the members of those equivalence classes.

Using the RCAR algorithm, Kočka and Castelo (2001) define the following two new concepts of neighborhood for DAG Markov models:

- **RCARNR** (RCAR+NR) Perform the RCAR algorithm and then create a NR neighborhood, denoted by $\mathcal{N}_{\text{RCARNR}}(G)$.
- **RCARR** (RCAR+NCR) Perform the RCAR algorithm and then create a NCR neighborhood, denoted by $\mathcal{N}_{\text{RCARR}}(G)$.

The RCARNR neighborhood may be seen as a simulation, or an approximation, of the ENR neighborhood, and analogously between the RCARR and ENCR neighborhoods. We can establish the following property for the RCARNR and RCARR neighborhoods.

Theorem 3.6. *Let* $\mathbf{D}(G)$ *be DAG Markov model. For a* sufficiently large *maximum number of iterations r of the RCAR algorithm,* $\mathbf{D}(G') \in \mathcal{N}_{\text{ENR}}(G) \Rightarrow \mathbf{D}(G') \in \mathcal{N}_{\text{RCARNR}}(G)$ *and* $\mathbf{D}(G') \in \mathcal{N}_{\text{ENCR}}(G) \Rightarrow \mathbf{D}(G') \in \mathcal{N}_{\text{RCARR}}(G)$*, with probability* $p > 0$*.*

Proof. It follows directly from Lemma 3.1 and the definitions of ENR, ENCR, RCARNR and RCARR neighborhoods. □

3.5 Heuristic Search

The problem of learning a DAG model, or a set of DAG models, that maximize the Bayesian Dirichlet metric (3.22) is proven to be NP-complete (Chickering, 1996a). For that reason, heuristic methods are suitable in order learn models that are close to the optimal solution. There is a large variety of heuristic algorithms to learn DAG Markov models, more popularly known as algorithms to *learn Bayesian Networks*. We may find surveys on these learning algorithms in (Bouckaert, 1995; Buntine, 1996b; Sangüesa and Cortés, 1997). Most of them, are basically a modification of the well-known K2 algorithm of Cooper and Herskovits (1992), which is shown in Figure 3.7.

The K2 algorithm implements a loop through the variables of the dataset from which it learns the model. At each iteration it adds the edge that maximizes the scoring function $g(i, \pi_i)$, where i refers to a given vertex and π_i to its parent set in the DAG, which is as follows:

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod N_{ijk}!.$$

This scoring function corresponds to the Bayesian Dirichlet metric with uninformative hyperparameters $N'_{ijk} = 0$ we already saw in expression (3.23), which assigns different scores to equivalent DAG models. As we may see in line 9 of the K2 algorithm, the number of parent vertices is constrained by some constant u. Also the possible additions of arcs are constrained by a causal ordering of the variables that bans adding an arc $a \rightarrow b$ if b *precedes* a in the ordering. This is implemented in the K2 algorithm through

```
algorithm K2 is
01    input: a set of n nodes, an ordering on the nodes,
02    an upper bound u on the number of parents of any node,
03    a database D containing N cases
04    output: for each node, a printout of the parents of the node
05    for i:=1 to n do
06        π_i := ∅
07        P_old := g(i, π_i)
08        OkToProceed :=true
09        while OkToProceed and |π_i| < u do
10            let z ∈ Pred(x_i) − π_i be the node that maximizes g(i, π_i ∪ {z})
11            P_new := g(i, π_i ∪ {z})
12            if P_new > P_old then
13                P_old := P_new
14                π_i := π_i ∪ {z}
15            else
16                OkToProceed :=false
17            endif
18        endwhile
19        write('node ',x_i,' parents ',π_i)
20    endfor
endalgorithm
```

Figure 3.7: The K2 algorithm from Cooper and Herskovits (1992)

the function $Pred(x_i)$, used in line 10, which returns those variables that precede x_i in the ordering. Here x_i refers to one of the random variables.

The usual extension of the K2 algorithm is to replace the loop through the variables by a loop that iterates until the scoring function does not improve. The scoring function is evaluated throughout a NR or an AR neighborhood at each iteration. This type of algorithm is commonly known as a *hill-climber* or *greedy search*.

In more general terms, a hill-climber creates a neighborhood of candidate models, scores them using some scoring function, and iterates by picking up the model that provides the highest increase in the scoring function. If no model provides any increase then the hill-climber stops and returns the current model.

The NR or AR neighborhoods for DAG models have been traditionally restricted by a maximum number of parents and a causal ordering because otherwise the hill-climber would not find a model of a reasonable *quality* or the algorithm would not scale to a larger number of variables. Therefore, some of the existing algorithms assume that the causal ordering is known, or they search for a *good* causal ordering that can be used later (Bouckaert, 1992; Singh and Valtorta, 1993; Larrañaga et al., 1996a; Friedman and Koller, 2000).

However, the causal ordering reduces the already small part of the inclusion boundary that was reachable using a NR or AR neighborhood. Therefore, errors in the ordering may easily lead to very bad local maxima, as shown in (Chickering et al., 1995). The same reasoning holds for the number of parents.

Heuristic algorithms that do not use any form of causal ordering may be found in Spirtes and Meek (1995); Chickering (1996b, 2001). These algorithms search directly in

the space of equivalence classes of DAG Markov models. They outperform all previous algorithms but at the computational cost of switching continuously between the space of essential graphs and the space of DAGs, in order to score the models. However, in this thesis we argue that the better results obtained by the use of the search space of equivalence classes are not merely a consequence of using the one to one representation (essential graphs) only, but by the side-effect that essential graphs respect the inclusion order better. Therefore, a sensible approach is to try to devise a way to use the space of DAGs, which is computationally cheaper, and account for the inclusion order.

Following this approach, we introduce a new heuristic algorithm (Kočka and Castelo, 2001) that partially accounts for the inclusion order and we show empirically that it is able to perform as good as the ones working on the space of essential graphs.

```
algorithm hcmc(int r, bool ncr) returns dag
01  dag g = emptydag
02  bool local_maximum = false
03  int trials = 0
04  while (¬local_maximum) do
05    g.rcar(r)
06    set nh = g.neighborhood(ncr)
07    dag g' = g.score_and_pick_best(nh)
08    local_maximum = (g'.score() < g.score())
09    if (¬local_maximum) then
10      g = g'
11      trials = 0
12    else if (trials < MAXTRIALS) then
13      g.rcar(r)
14      local_maximum = false
15      trials = trials + 1
16    endif
17  endwhile
18  return g
endalgorithm
```

Figure 3.8: Hill-Climber Monte Carlo algorithm

The algorithm we propose is given in Figure 3.8. It consists of a usual hill-climber that iterates through lines 4 to 17. It contains two modifications. One, on line 5, is to perform the RCAR algorithm from Figure 3.6 on the current DAG. The other is to perform again the RCAR algorithm when a local maximum is reached, as in line 13. This last step is performed a limited number of times (MAXTRIALS). If within this limited number of times, it has not been possible to escape from that local maximum, then the RCAR algorithm is not called again and the hill-climber will stop iterating.

The first of the two modifications, on line 5, is followed by the creation of an NR or NCR neighborhood, depending on the truth value of the parameter *ncr*. In this way, a

RCARNR or RCARR neighborhood will be employed. Afterwards, on line 7, the method $g.score\_and\_pick\_best(nh)$ scores all members of the neighborhood and returns the one that provides the highest score, which is assigned to g'.

The second of the two modifications, on lines 12 to 16, resembles in a way the *iterative hill-climber* introduced in Chickering et al. (1995), which consists of perturbing randomly the current model once a local maximum is reached. This is done in line 13 as well but the perturbation is constrained to a move within the equivalence class of the current model. Due to the random nature of the new steps introduced in the hill-climber, we call this algorithm the *Hill-Climber Monte Carlo*, or HCMC for short.

3.5.1 Experimental results on the Alarm dataset

In order to show the effectiveness of the HCMC algorithm we have applied it to the Alarm dataset (Beinlich et al., 1989), which is a synthetic dataset that has become a standard benchmark for the assessment of learning algorithms for DAG models on discrete data. The Alarm dataset was sampled from the Bayesian Network in Figure 3.9, which was designed for a monitoring system in the context of intensive care unit ventilator management.

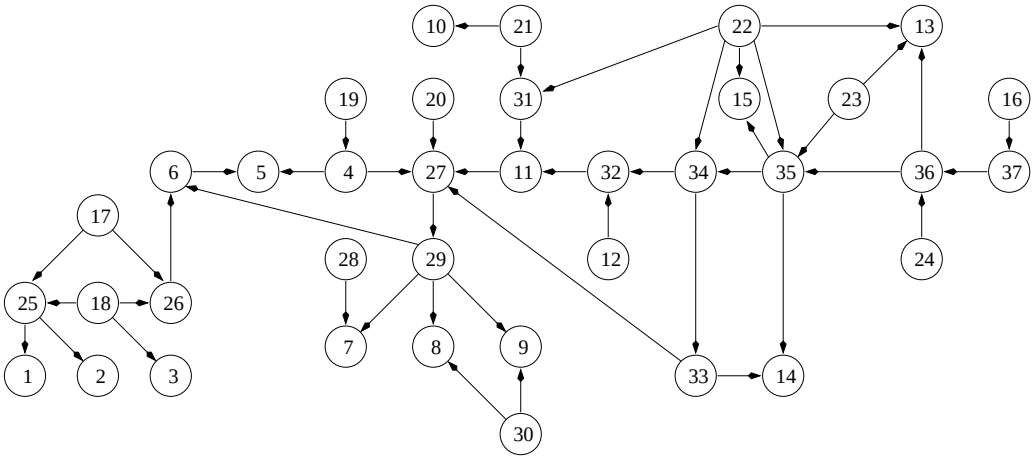


Figure 3.9: The Alarm Network Beinlich et al. (1989): 37 vertices and 46 edges.

The Alarm dataset, originally employed by Herskovits (1991), contains 20000 records. From this dataset the first 10000 records where used by Cooper and Herskovits (1992) to assess the K2 learning algorithm. We will use here this latter dataset of 10000 records. From this 10000 records we have sampled six different datasets of two different sizes: three of 1000 records and three of 5000 records. The experiments reported on 10000 records regard only the single dataset of the first 10000 records. As reported by Cooper and Herskovits (1992), this dataset of 10000 records does not support the arc between vertices 12 and 32 (see Figure 3.9).

The assessment has been done as follows. On each of the seven datasets, the HCMC was run ten times for four different cardinalities of RCAR (2, 4, 7 and 10) and three

different neighborhoods (AR, RCARR and RCARNR). The traditional hill-climber that uses an AR neighborhood will be referred here as RCAR 0. In all the runs the HCMC has employed the BDeu score metric (expression (3.22) with prior law (3.25)). We have also considered an uniform prior $p(M)$ over the space of GMMs.

The reason for running the HCMC several times is obvious since this new hill-climber performs random moves that may lead to different results in different runs. The maximum number of trials for escaping local maxima (MAXTRIALS) is set to 50. The results have been averaged over ten runs and 95% confidence intervals for the means of score and structural difference, have been included. The results are given in Tables 3.1 and 3.2.

Table 3.1: Results (1st part) of the HCMC learning algorithm over the Alarm dataset. Performance is averaged over RCARR and RCARNR.

smpl	rcar	performance		score		struct diff	
		steps	sec/st	RCARNR	RCARR	RCARNR	RCARR
1ka	0	55	0.27	-11480.47	-11480.47	29	29
	2	58	0.40	-11491.52±15.12	-11470.46±15.79	18.90±3.06	16.50±2.00
	4	55	0.44	-11484.69±19.29	-11473.41±14.18	18.00±3.28	16.30±2.18
	7	54	0.43	-11469.06±07.88	-11470.75±14.67	15.50±1.55	15.60±1.88
	10	53	0.43	-11470.43±15.94	-11464.03±11.00	14.80±2.15	15.20±1.78
1kb	0	60	0.28	-11115.13	-11115.13	28	28
	2	58	0.40	-11113.50±19.07	-11105.10±20.62	18.10±2.77	14.70±3.87
	4	56	0.42	-11121.49±24.64	-11090.15±08.51	17.90±4.41	11.10±2.35
	7	53	0.42	-11095.19±12.38	-11083.13±05.07	13.40±2.55	10.00±1.47
	10	53	0.43	-11095.87±11.25	-11094.17±18.72	12.40±2.05	11.50±2.11
1kc	0	62	0.61	-11530.80	-11530.80	37	37
	2	60	0.41	-11451.58±14.23	-11453.70±15.75	18.20±2.23	15.90±3.10
	4	59	0.43	-11438.31±08.01	-11436.65±07.46	14.90±2.95	13.40±1.94
	7	56	0.67	-11431.02±06.23	-11427.84±03.62	11.80±1.61	10.80±0.81
	10	53	0.86	-11440.88±10.78	-11428.89±08.95	13.70±2.13	11.00±1.17
5ka	0	69	1.54	-55249.43	-55249.43	46	46
	2	66	2.50	-55072.99±67.61	-54993.41±08.70	11.60±6.70	7.20±2.23
	4	57	2.09	-55051.93±40.93	-54992.40±10.92	7.90±2.12	5.20±1.38
	7	56	2.14	-55024.53±48.12	-54989.70±10.12	7.10±2.17	4.90±1.37
	10	56	2.08	-55025.19±43.49	-54985.99±06.68	6.10±1.95	5.10±2.49
5kb	0	57	0.92	-54732.19	-54732	33	33
	2	57	2.02	-54679.46±34.06	-54641.27±60.60	12.50±6.25	6.10±4.08
	4	56	1.35	-54610.82±15.24	-54607.60±14.34	5.20±3.93	3.80±1.38
	7	53	1.29	-54611.85±25.63	-54602.77±10.93	4.50±1.52	3.70±1.31
	10	52	1.28	-54602.98±10.85	-54606.47±12.48	4.00±1.26	4.10±1.32
5kc	0	59	0.88	-54454.16	-54454.16	36	36
	2	63	1.19	-54340.02±16.48	-54335.27±32.25	10.20±3.97	8.00±2.18
	4	59	1.20	-54335.49±19.99	-54326.25±11.15	8.60±1.91	8.40±2.05
	7	55	1.25	-54331.19±12.09	-54315.06±07.33	8.50±1.55	7.70±1.35
	10	55	1.28	-54363.17±52.63	-54329.40±11.13	10.00±2.18	8.50±1.85
10k	0	56	1.86	-108697.78	-108697.78	21	21
	2	56	2.23	-108495.65±68.33	-108463.65±46.17	4.90±2.20	5.40±4.10
	4	54	2.28	-108549.53±63.63	-108437.83±35.72	6.80±2.25	**1.60±0.90**
	7	50	2.29	-108477.50±52.06	-108485.55±58.14	5.50±3.22	**2.80±1.11**
	10	50	2.41	-108468.56±53.07	-108477.98±51.65	4.20±1.34	**3.30±1.17**

The information in Table 3.1 regards two aspects, the performance of the algorithm and the accuracy of the learned models with respect to the true one (Figure 3.9). The performance is provided in terms of number of steps, i.e. local transformations of a

DAG, that the HCMC performs before stopping and the average time per step. This is meant to be a timeless way of assessing performance since we would only need to scale the average time per iteration as the hardware performance improves. In this case, we have run the experiments on a Pentium-III processor machine with 512Mbytes of RAM running Linux (kernel version 2.2.14). The software implementing the algorithms has been developed as a module on top of the main-memory database system Monet (Boncz and Kersten, 1995, `http://www.cwi.nl/~monet`).

The accuracy of the learned models is measured in terms of the score metric, where the higher is the better and the structural difference[5] between the essential graph of the learned DAG and the essential graph of the true model of Figure 3.9. In summary, these are the measures taken:

- *steps*: number of steps ($g = g'$ in the algorithm) of the HCMC.
- *sec/st*: speed of the HCMC in seconds per step.
- *score*: confidence interval of the mean of the score.
- *struct diff*: confidence interval of the mean of the structural difference.

As we may appreciate, there is a substantial difference in using RCAR within the hill-climber. The structural differences, throughout all the seven samples, for the standard hill-climber (RCAR 0) fall far away outside the confidence intervals for any of the different cardinalities of RCAR, which are nicely centered at much lower values than the values for RCAR 0.

Regarding the score, on samples of 1000 records, RCAR 7 and 10 show a significantly higher score than RCAR 0, since the latter does not fall into their confidence intervals. At 5000 and 10000 records, the score for RCAR 0 falls outside the intervals for any of the cardinalities of RCAR.

The highest accuracy of the learned models is achieved when a RCARR neighborhood is used. The most striking evidence lies in the case of 10000 records and RCARR. There, an average of only 1.6 structural differences is achieved in about 54 steps. As its confidence interval already suggests, we note the fact that on *eight* out of ten of those runs, there was only one structural difference, corresponding to the missing arc not supported by the data. In some of those eight times, the result was reached in 49 steps and the same result was reached for RCAR 7 and 10 even in 48 steps, which is extremely close to the optimal path of the right result (46 additions).

In order to gain further insight into the HCMC algorithm and discuss its performance, we have taken further measures given in Table 3.2:

- *models considered*: accumulated number of models that have improved the score at each step of the learning algorithm.
- *escapes/trials*: number of escapes of local maxima performed by doing RCAR and average number of trials per escape.

As we did with the scores, confidence intervals at a level of 95% are provided for the number of models considered. Although all of them are quite wide, they show that

[5]Number of adjacencies that differ.

Table 3.2: Results (2nd part) of the HCMC learning algorithm over the Alarm dataset. Performance is averaged over RCARR and RCARNR.

smpl	rcar	models considered		escapes/trials		struct diff	
		RCARNR	RCARR	RCARR	RCARNR	RCARNR	RCARR
1ka	0	6505	6505	0	0	29	29
	2	6037±137.1	6061±131.4	2.40/7.72	2.30/6.82	18.90±3.06	16.50±2.00
	4	5856±117.2	5936±145.9	1.60/6.55	2.00/3.80	18.00±3.28	16.30±2.18
	7	5810±71.4	5862±113.5	1.60/4.47	1.70/2.92	15.50±1.55	15.60±1.88
	10	5723±63.6	5816±111.6	1.20/3.00	1.70/3.45	14.80±2.15	15.20±1.78
1kb	0	6460	6460	0	0	28	28
	2	5940±79.7	5956±131.7	1.70/5.07	2.20/11.36	18.10±2.77	14.70±3.87
	4	5933±97.3	5861±120.1	2.00/10.53	1.60/6.17	17.90±4.41	11.10±2.35
	7	5800±89.7	5810±109.9	0.20/4.00	0.70/6.00	13.40±2.55	10.00±1.47
	10	5787±59.0	5794±67.0	0.40/1.33	0.60/16.33	12.40±2.05	11.50±2.11
1kc	0	6760	6760	0	0	37	37
	2	5943±101.6	6123±141.5	5.40/8.68	2.50/2.02	18.20±2.23	15.90±3.10
	4	5903±127.5	5928±126.8	3.70/4.66	3.50/6.10	14.90±2.95	13.40±1.94
	7	5904±94.7	5782±80.0	1.80/3.02	1.40/3.10	11.80±1.61	10.80±0.81
	10	5823±53.0	5711±43.6	1.70/3.52	1.90/1.96	13.70±2.13	11.00±1.17
5ka	0	8307	8307	0	0	46	46
	2	6956±153.5	7346±236.5	4.70/9.08	6.10/6.80	11.60±6.70	7.20±2.23
	4	6849±82.7	6921±188.8	1.30/7.94	1.80/7.99	7.90±2.12	5.20±1.38
	7	6784±97.8	6733±160.6	1.60/10.39	1.50/5.87	7.10±2.17	4.90±1.37
	10	6760±197.5	6678±88.3	0.90/7.21	1.00/3.97	6.10±1.95	5.10±2.49
5kb	0	7418	7418	0	0	33	33
	2	6881±171.4	6704±217.4	4.20/7.11	2.10/15.12	12.50±6.25	6.10±4.08
	4	6552±158.9	6565±159.8	2.70/6.18	2.50/7.68	5.20±3.93	3.80±1.38
	7	6414±116.3	6464±160.6	1.60/7.64	1.50/7.28	4.50±1.52	3.70±1.31
	10	6305±114.7	6417±129.5	0.80/7.71	1.30/5.96	4.00±1.26	4.10±1.32
5kc	0	7560	7560	0	0	36	36
	2	7165±149.5	7081±149.5	4.70/6.45	3.70/7.65	10.20±3.97	8.00±2.18
	4	6931±159.9	6905±178.2	1.70/5.10	2.40/3.42	8.60±1.91	8.40±2.05
	7	6873±122.4	6803±114.8	1.30/7.94	0.70/7.22	8.50±1.55	7.70±1.35
	10	6846±44.8	6783±111.8	1.20/3.60	0.60/2.33	10.00±2.18	8.50±1.85
10k	0	7774	7774	0	0	21	21
	2	6915±184.6	6908±185.0	2.80/14.25	2.20/8.93	4.90±2.20	5.40±4.10
	4	6865±199.1	6874±163.5	1.30/5.57	2.60/12.57	6.80±2.25	**1.60±0.90**
	7	6695±111.4	6790±147.5	0.80/4.83	1.30/7.91	5.50±3.22	**2.80±1.11**
	10	6682±140.0	6784±78.1	1.00/6.56	1.20/7.22	4.20±1.34	**3.30±1.17**

the HCMC algorithm considers significantly less models to achieve a better result. This means that the HCMC algorithm makes better choices during the search process which, provided its greedy nature, implies that the way HCMC traverses the search space is definitely better.

The number of escapes and trials, provide an idea of how effective the RCAR algorithm is in avoiding local maxima. We may see that a higher number of RCAR implies a lower number of times that the HCMC escapes from a local maxima achieving a similar result. We may appreciate that with the RCARR neighborhood the number of times HCMC is able to escape from local maxima is slightly higher. In preliminary experiments not reported here, we noted that this difference is more significant if the maximum number of trials (MAXTRIALS) is smaller. This fact may be one of the reasons why RCARR seems to perform slightly better than RCARNR.

Finally, let's highlight the computational trade-off that this approach affords. We want to know what is the overhead of using the RCAR operation as in the HCMC algorithm in front of the usual hill-climber. We can see that by comparing the seconds taken per step in RCAR 0 with respect to any other cardinality of RCAR. This information is in Table 3.1, and we may see that the HCMC algorithm is, at most, *two times* slower than the usual hill-climber.

In the three papers[6] that develop heuristic algorithms that work directly in the search space of essential graphs and provide experimental results, one of them (Chickering, 1996b) reports that the algorithm is about 20 times slower than the usual hill-climber in the space of DAGs for a 10000 record dataset sampled from the Alarm network. In (Spirtes and Meek, 1995) only absolute times on a specific platform are provided for a dataset of the same size sampled also from the Alarm network, thus comparison is not possible. However, both algorithms rely on the transformation of DAG to essential graph, and essential graph to DAG, in order to score the models. Therefore we conjecture that the trade-off of the algorithm in (Spirtes and Meek, 1995) will be similar to the one in (Chickering, 1996b). This fact allows us to conclude that our algorithm is *an order of magnitude* faster than these two works, and yields the same result.

In the third paper (Chickering, 2001), the author provides a set of traversal operators in the space of essential graphs that always yield local changes in the score, such that they are as efficient as the usual traversal operators in the space of DAGs. No experiments with the Alarm dataset are reported and therefore, direct comparison with our approach is not possible, as in (Spirtes and Meek, 1995; Chickering, 1996b). Nevertheless, the approach in (Chickering, 2001) does not account for the inclusion order and may easily suffer of bad local maxima as in the situation illustrated in Figure 3.4. Therefore we may assert that our approach is also fully competitive with this learning algorithm.

3.6 The Markov Chain Monte Carlo Method

In section 3.2 we reviewed Bayesian score metrics which are used to rank models. In particular, expression (3.13) shows us that computing the likelihood times the prior of the model, suffices to rank models from a Bayesian perspective. Applying heuristic procedures, as the one we saw in the previous section (Figure 3.8), we obtain a model (or a set of them) that maximizes (up to local maxima) the Bayesian score metric. However, we still may not be completely satisfied for the following two reasons:

1. We can see from the ranking which models are better, but we cannot figure out from this ranking to which extent they are supported by the data.

2. We need to draw conclusions, or compute some quantity of interest, conditioning on the best models. Nevertheless, because conclusions or quantities may differ substantially among models, we need to weigh somehow the different alternatives and a ranking of the models does not provide such information.

These two problems arise from the same source: the need to account for the *uncertainty* of the models (Draper, 1995). From the Bayesian perspective, that requires the

[6] This is all the competition I'm aware of.

computation of the normalizing constant such that we obtain the full posterior $p(M|D)$. Recall Bayes' theorem:

$$p(M|D) = \frac{p(D|M)p(M)}{p(D)},$$

where

$$p(D) = \sum_{M \in \mathcal{M}} p(D|M)p(M). \tag{3.27}$$

Once we account for the uncertainty of the models, it is possible to weigh competing conclusions and quantities of interest, by averaging over all the models in the following way:

$$p(\Delta|D) = \sum_{M \in \mathcal{M}} p(\Delta|M, D)p(M|D), \tag{3.28}$$

where Δ refers to the quantity of interest and $p(\Delta|D)$ is its posterior distribution given the data D as the notation suggests. We know from chapter 2 that the size of any class $\mathcal{M}$ of GMMs prohibits enumerating all the models. In other words, it is not feasible to compute the sums in (3.27) and (3.28).

The Markov chain Monte Carlo (MCMC hereafter) method solves this problem by sampling directly from the posterior distributions $p(M|D)$ and $p(\Delta|D)$, thus performing the summations implicitly. The MCMC method has its origins in a sampling method introduced by Metropolis et al. (1953) within the context of statistical physics. However, it was Hastings (1970), who generalized this sampling method for statistical problems, by using the theory of Markov chains, introducing the well known *Metropolis-Hastings* algorithm.

The scope of application of the MCMC method is very broad and in this section we will focus on its use for structural learning in GMMs. The interested reader may find insightful explanations, different applications as well as useful extensions of the method in (Smith and Roberts, 1993; Chib and Greenberg, 1995; Carlin and Chib, 1995; Green, 1995; Brooks, 1998; Chib and Jeliazkov, 2001).

The Metropolis-Hastings algorithm was adapted for structural learning of GMMs by Madigan and York (1995), who called it the *Markov Chain Monte Carlo Model Composition*, or MC^3 algorithm for short.

Let $\mathcal{M}$ denote the class of GMMs under consideration. For each GMM $M \equiv \mathbf{M}(G) \in \mathcal{M}$, let $\mathcal{N}(G)$ be the set of neighbor models of M. Let q be a transition matrix such that for some other $M' \equiv \mathbf{M}(G') \in \mathcal{M}$, $q(M \to M') = 0$ if $M' \notin \mathcal{N}(G)$ and $q(M \to M') > 0$ if $M' \in \mathcal{N}(G)$. Using the transition matrix q we build a Markov chain $M(t, q)$, $t = 1, 2, \ldots, n$, with state space $\mathcal{M}$.

Let the chain $M(t, q)$ be in state M and let's draw a candidate model M' from $q(M \to M')$. The proposed model M' is accepted with probability

$$\alpha(M', M) = min\left\{1, \frac{|\mathcal{N}(G)|p(M'|D)}{|\mathcal{N}(G')|p(M|D)}\right\}, \tag{3.29}$$

where $|\mathcal{N}(G)|$ refers to the cardinality of the set of neighbors of the model M. If M' is not accepted, $M(t,q)$ remains in state M. The underlying idea is that a Markov chain $M(t,q)$ built in this way, has $p(M|D)$ as its equilibrium distribution. This means that, after the chain has run *long enough*, the draws can be regarded as a sample from the target density $p(M|D)$, and one says that the chain has *converged*.

The *convergence* of the Markov chain to the target density $p(M|D)$ is guaranteed under two regularity conditions:

1. *Irreducibility*. There is a positive probability that the chain moves from any model $M \in \mathcal{M}$ to any other model $M' \in \mathcal{M}$ in a finite number of steps.

2. *Aperiodicity*. There is a positive probability of remaining in the current model.

In depth discussion of these conditions is beyond the scope of this section, but the reader may consult (Smith and Roberts, 1993). In our particular case, we should take care on how a new candidate graph is proposed, in order to retain those regularity conditions. We shall see later the concrete cases of DEC and DAG Markov models.

Given output $M(t,q) = \{M_{t=1}, M_{t=2}, \ldots, M_{t=n}\}$ of the Markov chain, the regularity conditions allow us to derive the following asymptotic results (Chung, 1967; Hastings, 1970; Smith and Roberts, 1993; Madigan and York, 1995):

$$M_{t=n} \overset{n\to\infty}{\longrightarrow} M \sim p(M|D) \tag{3.30}$$

$$\frac{1}{n}\sum_{t=1}^{n} g(M(t,q)) \overset{n\to\infty}{\longrightarrow} E(g(M)) \tag{3.31}$$

Which imply that, when the Markov chain $M(t,q)$ converges:

- the draws from the Markov chain mimic a random sample from $p(M|D)$. Therefore, in order to get an estimate of $p(M|D)$, it suffices to account for the frequency of visits of each model M and divide it by the number iterations n,

and

- the average of the realizations of any function of the model, $g(M)$, is an estimator of the expectation of $g(M)$. Therefore, by setting $g(M) = p(\Delta|M,D)$ we will approximate the sum in (3.28) (Madigan and York, 1995),

respectively. In Figure 3.10 we see the pseudocode of the MC^3 algorithm, which requires the specification of the following five parameters:

- *init*: some arbitrary GMM from which the Markov chain starts.
- n: number of iterations that the Markov chain will perform.
- g: function that takes a GMM as input and gives an integer number as output, which indexes some quantity of interest of the model.
- q: function that takes a GMM as input and gives a GMM as output, drawn from an irreducible transition matrix.

```
algorithm MC3(gmm init, int n, int(gmm) g, gmm(gmm) q,
              int burn_in) returns {flt[],flt[]}
01  gmm M = init
02  int i = 0
03  flt p[]
04  flt d[]
05  while (i < n) do
06    gmm M' = q(M)
07    flt r = U(0,1)
08    if (r ≤ α(M,M')) then M = M' endif
09    if (i > burn_in) then
10      p[M] = p[M] + 1
11      d[g(M)] = d[g(M)] + 1
12    endif
13    i = i + 1
14  endwhile
15  for M = p.fst() to p.lst() do p[M] = p[M]/n enddo
16  for i = 1 to d.size() do d[i] = d[i]/n enddo
17  return {p,d}
endalgorithm
```

Figure 3.10: The MC^3 algorithm.

- *burn_in*: number of iterations we want to discard before the algorithm starts accounting for the frequency of visits to the models.

The algorithm uses two vectors p and d to store the estimated posteriors $p(M|D)$ and $p(\Delta|D)$, which are the result that the algorithm returns.

The Markov chain iterates through lines 5 to 14. In line 6, it draws a candidate model M', which is accepted in line 8 with probability specified by the acceptance ratio in (3.29). If the current iteration is larger than the burn-in period (line 9), then the current state of the Markov chain is stored in lines 10 and 11.

Finally, in lines 15 and 16, the averages of the stored quantities p and d are computed and they are returned in line 17.

In subsection 3.3 we introduced the notions of neighborhood and of traversing the search space. In particular, we formalized this latter notion into two steps, namely, proposing and moving.

These two steps correspond in the MC^3 algorithm to lines 6 and 8, respectively, in Figure 3.10. That is, to draw a candidate model, and to accept it with certain probability. However, while in section 3.3, proposing requires the creation of an entire neighborhood of models, here proposing is picking up a single model from that neighborhood randomly.

In the following two subsections, we tackle particular aspects of proposing and moving for DEC and DAG Markov models, as well as their implications into the irreducibility of the Markov chain. Afterwards, we provide insight into the question: for how long should we run the Markov chain?. To conclude this section, we show some experimental results regarding convergence, using the improvements proposed in subsection 3.6.2 for

the MC^3 algorithm on DAG Markov models.

3.6.1 MC^3 on DEC Markov models

In subsection 3.3.1 we specified the conditions for a legal move in the class of DEC Markov models. These legal moves consisted of adding and removing one undirected edge such that the new graph remains chordal. By performing these legal moves on every adjacency, we obtained the 1M1L neighborhood, noted $\mathcal{N}_{1\text{M}1\text{L}}(G)$.

We want to pick at random a model in $\mathcal{N}_{1\text{M}1\text{L}}(G)$, such that the Markov chain is irreducible. Define $q(M)$ as follows:

1. Let $M \equiv \mathbf{U}(G)$ be the DEC Markov model given as parameter, and therefore G is a chordal graph.
2. Let $G = (V, E)$ where V is the vertex set and E the edge set. Pick two different vertices $u, v \in V$ at random.
3. If u–$v \notin E$ then check whether $G' = (V, E \cup \{u\text{–}v\})$ is chordal. If the addition is legal (G' is chordal) then return $\mathbf{U}(G')$, otherwise go to 2.
4. If u–$v \in E$ then check whether $G' = (V, E \backslash \{u\text{–}v\})$ is chordal. If the removal is legal (G' is chordal) then return $\mathbf{U}(G')$, otherwise go to 2.

.

Frydenberg and Lauritzen (1989, Lemma 5) proved that adding and removing legal edges at random in this way is sufficient to achieve irreducibility. Once the proposed move is accepted (line 8 in Figure 3.10), we proceed moving as specified in subsection 3.3.1.

As specified also in line 8 of the MC^3 algorithm in Figure 3.10, the acceptance of a move depends on the acceptance ratio (3.29). This ratio is in fact the product of the following two ratios:

- the ratio of the cardinalities of the neighborhoods, $|\mathcal{N}(G)|/|\mathcal{N}(G')|$, which is known as the candidate-generating ratio (Chib and Greenberg, 1995).
- the ratio of two posterior probabilities that belong to the posterior distribution from which we intend to sample, $p(M'|D)/p(M|D)$.

Although the chordality condition of the graphs here constrains the set of edges one may remove or add for each graph, it is reasonable to assume that $|\mathcal{N}(G)| = |\mathcal{N}(G')|$ given that G and G' differ in one single adjacency. In that case one says that we are assuming a *symmetric* candidate-generating density (Chib and Greenberg, 1995).

In this way, the acceptance ratio (3.29) involves only $p(M'|D)/p(M|D)$. Because the normalizing constant appears in both the numerator and denominator, the calculation of (3.29) does not require this constant. The acceptance ratio corresponds then to the Bayes factor

$$\alpha(M', M) = \frac{p(D|M')p(M')}{p(D|M)p(M)}, \tag{3.32}$$

where each of the factors corresponds to the product of the likelihood of the model times the model prior, which we already saw in section 3.2, expression (3.15). Because the two graphs that determine the models M and M' differ in one single adjacency, the ratio (3.32) requires only a few local computations.

Let u–v be the edge in G' ($M' \equiv \mathbf{M}'(G')$) that is not in G ($M \equiv \mathbf{M}(G)$). Let C be the clique in G' that contains the edge u–v. Let $C_u = C \backslash \{v\}$, $C_v = C \backslash \{u\}$ and $C_0 = C \backslash \{u, v\}$. The Bayes factor in (3.32) of M' against M can be expressed as (Dawid and Lauritzen, 1993):

$$\alpha(M', M) = \frac{p_C(D|\vartheta) p_{C_0}(D|\vartheta) p(M')}{p_{C_u}(D|\vartheta) p_{C_v}(D|\vartheta) p(M)}, \tag{3.33}$$

where each of the four terms corresponds to the marginal of the data as in (3.12). If, conversely, the edge u–v is in M and not in M', then the acceptance ratio is simply the inverse of expression (3.33).

3.6.2 MC^3 on DAG Markov Models

In subsection 3.3.2 we specified the conditions for legal additions, removals and arc reversals. Moves in this class are legal as long as acyclicity of the directed graph is retained. As argued by Madigan and York (1995), legal addition and removal of arcs, at random, suffices to make the Markov chain irreducible. Since, we also include arc reversal, proposition is specified as follows (q function called in line 6 of the MC^3 algorithm):

1. Let $M \equiv \mathbf{D}(G)$ be the DAG Markov model given as parameter, and therefore G is an acyclic digraph.

2. Let $G = (V, E)$ where V is the vertex set and E the edge set. Pick an ordered pair of vertices $(i, j) \in V \times V$ at random.

3. If $j \to i \notin E$ then check whether $G' = (V, E \cup \{j \to i\})$ is acyclic. If the addition is legal (G' is acyclic) then return $\mathbf{D}(G')$, otherwise go to 2.

4. If $j \to i \in E$ then pick one of these two possibilities at random:

   (a) $G = (V, E \backslash \{j \to i\})$. Return $\mathbf{D}(G')$.

   (b) if $G' = (V, \{E \backslash \{j \to i\}\} \cup \{j \leftarrow i\})$ remains acyclic then return $\mathbf{D}(G')$, otherwise go to 2.

Analogous to the case of DEC Markov models, we assume here a symmetric candidate-generating density, i.e. $|\mathcal{N}(G) = |\mathcal{N}(G')|$. Again this is reasonable because G and G' differ in one single adjacency. Therefore, as we saw in expression (3.32), only likelihoods and model priors are involved. We also take advantage here of the single adjacency difference to compute the Bayes factor in the following way.

From the likelihood expression in (3.21), let $p_i(D|M)$ be the ith term of the outmost product.

$$p_i(D|M) = \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}. \tag{3.34}$$

Let M' be the result of proposing either an addition or a removal of an arc $j \rightarrow i$ to/of M. The acceptance ratio involves only those sets of parent vertices that differ, that is

$$\alpha(M', M) = \frac{p_i(D|M')p(M')}{p_i(D|M)p(M)}, \tag{3.35}$$

where $p_i(D|M')$ and $p_i(D|M)$ correspond to the expression in (3.34). If M' is the result of proposing the reversal of the arc $j \rightarrow i$ into $j \leftarrow i$, then the acceptance ratio involves four different sets of parent vertices, as follows

$$\alpha(M', M) = \frac{p_i(D|M')p_j(D|M')p(M')}{p_i(D|M)p_j(D|M')p(M')}. \tag{3.36}$$

An enhanced MC^3 algorithm for DAG Markov models

In section 3.4 we introduced the concept of graphical Markov model inclusion and discussed its relevance to structural learning of GMMs from data. In the same section we described two new neighborhood concepts, namely RCARNR and RCARR, based on the RCAR algorithm (see Figure 3.6).

The benefit of using these two neighborhood concepts was shown in section 3.5 within the context of heuristic search, by extending the usual hill-climber. In the same vein, we are going to incorporate the RCAR operation in the MC^3 algorithm.

The modification is very simple. The MC^3 algorithm (see Figure 3.10) for DAG Markov models, as introduced by Madigan and York (1995), only considered additions and removals of arcs. Using the terms introduced in section 3.3, the MC^3 algorithm picks a model randomly from an NR neighborhood. We propose here that the MC^3 algorithm picks a model randomly from either an RCARNR or an RCARR neighborhoods (Kočka and Castelo, 2001). Thus modifying the function $q(M)$ in the following way:

1. Let $M \equiv \mathbf{D}(G)$ be the DAG Markov model given as parameter, and therefore G is an acyclic digraph. Let *ncr* be a boolean variable that is *true* if $q(M)$ should pick a model from an RCARR neighborhood and *false* if $q(M)$ should pick a model from an RCARNR neighborhood. Let r be the maximum number of iterations that the RCAR algorithm should perform.
2. Let $G = (V, E)$ where V is the vertex set and E the edge set.
3. Perform the RCAR operation over G, i.e. call $G.rcar(r)$ (see algorithm in Figure 3.6).
4. Pick an ordered pair of vertices $(i, j) \in V \times V$ at random.
5. If $j \rightarrow i \notin E$ then check whether $G' = (V, E \cup \{j \rightarrow i\})$ is acyclic. If the addition is legal (G' is acyclic) then return $\mathbf{D}(G')$, otherwise go to 2.

6. If $j \rightarrow i \in E$ and $ncr =$*false* then $G' = (V, E \backslash \{j \rightarrow i\})$. Return $\mathbf{D}(G')$.

7. If $j \rightarrow i \in E$ and $ncr =$*true* then pick one of these two possibilities at random:

   (a) $G' = (V, E \backslash \{j \rightarrow i\})$. Return $\mathbf{D}(G')$.

   (b) if $j \rightarrow i$ is not covered in E and $G' = (V, \{E \backslash \{j \rightarrow i\}\} \cup \{j \leftarrow i\})$ is acyclic then return $\mathbf{D}(G')$, otherwise go to 2.

As we see, in order to obtain the RCARNR neighborhood, a pair of ordered vertices should be picked at random and an arc added or removed, after the RCAR algorithm has been applied to the current model M. Hereafter, we will refer to this modified MC^3 algorithm as the *enhanced* or *e*MC^3.

Additionally, to obtain the RCARR neighborhood within the MC^3 algorithm, we should also consider the possibility of reversing the arc, when it exists between the two randomly chosen vertices. However, this arc should be reversed only if it is non-covered. Thus, performing a non-covered reversal (see section 3.3). We shall see experimentally the benefits of incorporating these two new neighborhoods in subsection 3.6.4.

3.6.3 Convergence Diagnostics

At the beginning of this section, we pointed out the necessary condition of the Markov chain to *converge* to the equilibrium distribution, in order to regard the learned models as samples from that distribution. We briefly reviewed the regularity conditions under which this convergence is guaranteed. However, these conditions do not determine how long the Markov chain should run in order to converge. This is for MCMC methods in general, a difficult empirical question, for which specific notions and techniques need to be devised in each different implementation.

The literature regarding convergence diagnostics for GMMs is rather small. We will discuss four convergence diagnostics introduced by Giudici and Green (1999) and Giudici and Castelo (2001b). In the next subsection, their use will be illustrated through experiments with synthetic data.

As pointed out by Smith and Roberts (1993, pg. 9), a sensible approach to assess convergence consists of monitoring ergodic averages of selected scalar quantities for stationarity. In the case of GMMs, this means to monitor the mixing of the simulation over the graphs, through a summary measure of each of them. For instance, the average number of edges (Giudici and Green, 1999).

This average can be easily implemented by accumulating in some integer variable, the sum of the number of edges present in the current model M, in each iteration of the Markov chain. This sum would be carried out at the level of line 13 in the MC^3 algorithm in Figure 3.10. Then, dividing this quantity by the current iteration i, we obtain the running average number of edges.

The rationale behind monitoring the average number of edges is that those GMMs with higher posterior will be sampled more often, and therefore the average number of edges of these GMMs should be approached by the running average number of edges. If we are assessing convergence at some moment during the Markov chain run, and this average shows some slope (it is not straight), it is most likely that the Markov chain has not converged.

The next convergence diagnostic for GMMs is to monitor the approximated marginal likelihood of the data $p(D)$ (Giudici and Castelo, 2001b). In Bayes' theorem, we can swap terms and obtain the following expression for this marginal:

$$p(D) = \frac{p(D|M)p(M)}{p(M|D)}.$$

Note that this equality holds for any given GMM M. Obviously, when the posterior $p(M|D)$ is approximated by MCMC, only an approximate marginal likelihood $\hat{p}(D)$ can be obtained. Such an approximation will be better for models in an area of high probability in the posterior distribution. As Kass and Raftery (1995) point out, small likelihoods may have large effects on the final approximation and make the resulting estimator $\hat{p}(D)$ very unstable. This suggests that we compute the approximate marginal likelihood as an average of the approximations from the models with highest posteriors only.

Let $\hat{p}(M|D)$ be the current estimated posterior for model M give data D. Let $p(D|M)$ be the current likelihood of the model M. The marginal likelihood $\hat{p}(D)$ can be estimated as:

$$\hat{p}(D) = \frac{1}{|\mathcal{B}|} \sum_{\mathcal{B}} \frac{p(D|M)p(M)}{\hat{p}(M|D)} \quad M \in \mathcal{B}, \tag{3.37}$$

where $\mathcal{B}$ is a set formed by the models M with highest posterior probability in each iteration of the MC^3 algorithm. In our experiments below, we have chosen $\mathcal{B}$ as the best 5 models. Again, this running average can be implemented at the level of line 13 in Figure 3.10.

Another interesting average to monitor is the ratio between the number of accepted and rejected proposed models (Giudici and Castelo, 2001b). If this ratio does not stabilize around some constant value, it is likely that the Markov chain is within an area of low posterior probabilities, and is jumping *uphill* (accepting very often) in order to reach an area of higher posterior distribution. Similarly, when this area of high posterior probabilities is reached, it will take some iterations to stabilize the average of this ratio. The calculation of this ratio should be implemented in the same place as the previous two diagnostics.

Finally the last convergence diagnostic corresponds to the posterior distribution of the total number of edges present (Giudici and Castelo, 2001b). More formally, let n be the number of vertices of the graphs that determine the GMMs. Let W be the random variable that takes as value, the number of edges of the graph at each iteration of the Markov chain. The integer random variable W will take values in the range $[0, 1, 2, \ldots, n(n-1)/2)]$, which are all possible cardinalities of an edge set of either an undirected chordal graph or an acyclic digraph on n vertices.

The quantity to monitor is $p(W|D)$, which is computed as in the general case of any quantity of interest Δ (see beginning of this section). We have noticed in our experiments that this posterior distribution has a normal shape, and it is centered close to the cardinality of the model for which the Markov chain gives the highest posterior. If the center of the normal shape shifts through longer runs, it means that the Markov chain has not converged.

As a final remark regarding convergence we will discuss the issue of the starting point of the Markov chain (the *init* parameter by the MC^3 algorithm in Figure 3.10). It is of general agreement within the MCMC literature that the run of the Markov chain is often sensitive to the starting point. Therefore it makes sense to try several runs from different starting points chosen at random.

However, within the context of random generation of acyclic digraphs, Melançon et al. (2000) point out that starting the process on the empty graph gives an effective way of achieving a good mixing rate of the chain. They explain such effect as follows:

> Observe that the maximal distance between any two acyclic digraphs is bounded by $n(n-1)$, since an obvious (but far from optimal) path connecting them goes through the empty graph (by first deleting all edges from the first graph and then adding the edges of the second graph).

We consider that in our context, where the distribution of the graphs (determining the GMMs) is not uniform, it still makes sense to follow that advice because one may reason analogously in terms of graphical Markov model inclusion.

In addition, we will consider as good starting points those GMMs of a high likelihood, as they are close to the mode of the distribution and therefore it may accelerate convergence of the chain. One may obtain GMMs with a high likelihood by using heuristic procedures as those explained in section 3.5. Note that, however, the heuristic procedures do not guarantee to find the model that maximizes the score. Hence, starting the Markov chain from such points may also give a bias in that the Markov chain can miss an important area of the posterior.

3.6.4 Experimental results on the Alarm dataset

In this subsection we illustrate the benefit of extending the MC^3 algorithm with the RCARNR and RCARR neighborhoods (using the *e*MC^3 algorithm), by using the convergence diagnostics previously introduced. At the same time we will see these diagnostics working. The set up of the experiments will be the same as in the corresponding subsection for heuristic search, section 3.5.

A further aspect we consider here is the effect on the outcome of starting the *e*MC^3 algorithm from a different point than the empty DAG model (determined by an acyclic digraph with no edges). We have started most of the experiments from the empty DAG model, but for some we take the output of the HCMC algorithm as starting point. Recall from section 3.5 that this output was the true Alarm network with one arc missing which was not supported by the data. We will refer to this Alarm network as the *almost true* Alarm network and it will be noted with an asterisk in the legends.

As for heuristic search, we will be using here the BDeu metric (see section 3.2) in its form as a Bayes factor explained before (see expression (3.36)). Because this metric gives equal scores to equivalent DAG Markov models, the posterior distribution of DAG models as well as other derived quantities, are transformed into quantities or distributions indexed by essential graphs. For that purpose, we have used the algorithm of Chickering (1995) that obtains the corresponding essential graph of a given DAG.

We have run the *e*MC^3 algorithm for 10^5 iterations over each of the seven samples of the Alarm dataset, and for the 10000 records sample we ran the chain starting from the

Table 3.3: Mobility of the Markov chain and Kullback-Leibler distance per essential graph.

	size	AR	CR	NCR	RCARR	RCARNR
number essential graphs	1k	1764	1622	1632	1898	2017
	5k	830	780	660	991	955
	10k	561	470	526	727	654
	10k*	553	485	612	577	626
K-L per e.g.	1k	4.36525	4.46228	4.37798	4.33639	4.38295
	5k	3.78184	3.78113	3.95602	3.66680	3.75692
	10k	3.73537	4.05203	3.97652	3.53715	3.75776
	10k*	2.70025	2.70035	2.70018	2.70029	2.70046

almost true Alarm network. We do not provide all the results for every sample, since for some combinations the conclusions are the same.

A first aspect we are going to look at, is the *mobility* of the Markov chain. The mobility is a relevant aspect because the higher the mobility, the lower the chance that the approximated posterior distribution does not reflect an important area of the search space.

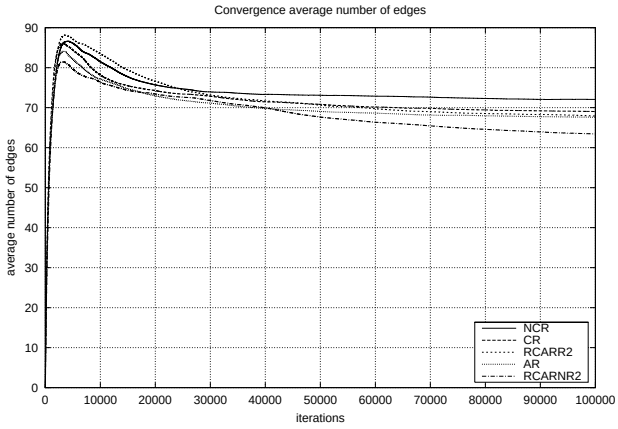
In Table 3.3 we can see the averages across the samples and across the RCAR cardinalities of 2,4 and 10 of the different essential graphs visited during the process. We see the average Kullback-Leibler distance (Kullback and Leibler, 1951) per essential graph as well. It is clear that RCAR yields a higher mobility of the Markov chain since more essential graphs were visited when RCAR was used.

This higher mobility results in a better choice of the DAG Markov models during the process, as can be seen from the lower Kullback-Leibler ratios for the cases where RCAR was used. The asterisk in Table 3.3 denotes the case where we used the output of the HCMC algorithm as starting point. In this latter situation, the Kullback-Leibler ratio does not show a gain while using RCAR. This is because the Markov chain starts from a good point and all the neighboring models to where the chain jumps still provide a good Kullback-Leibler distance.

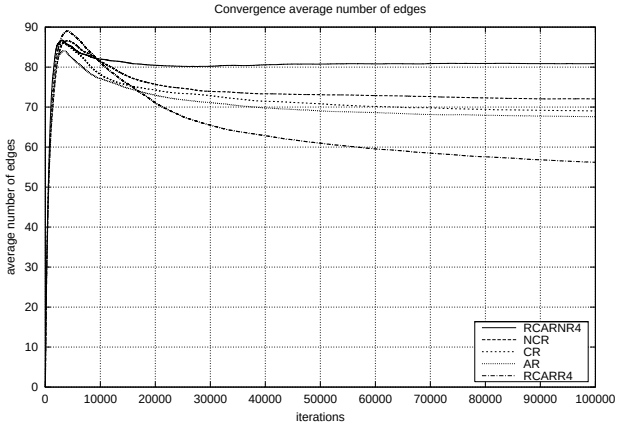
The main point we want to show in fact, is that the RCAR operation improves the convergence of the MC^3 algorithm. First we will look at the behavior of the running average number of edges. We have monitored this diagnostic on the MC^3 algorithm starting from the empty DAG model and using the standard neighborhoods AR, CR and NCR, and the newly introduced RCARNR and RCARR (eMC^3 algorithm). In the case of these two latter ones, we have run the experiments with three different parameter values for the RCAR algorithm (see Figure 3.6), namely 2,4 and 10.

In contrast with the experiment about the mobility of the chain, we will not show here the results for all samples, but only for the 10^4 records sample, as it already shows all the points we want to make.

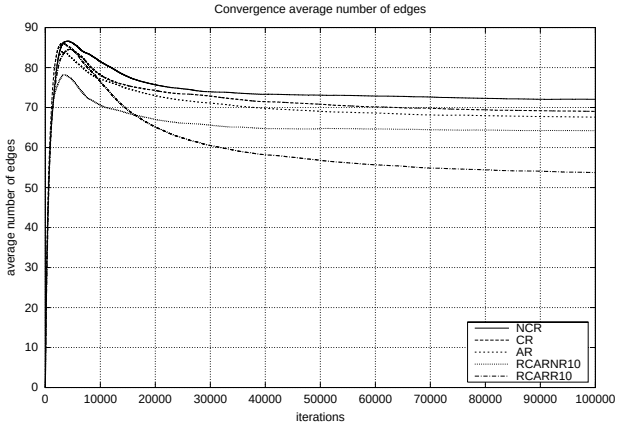
We have plotted the AR, CR and NCR neighborhoods against RCARNR and RCARR, for each of the three cardinalities, 2, 4 and 10, in Figure 3.11. Recall that the true Alarm network has 46 edges (see Figure 3.9). Therefore we should expect that the average number of edges converges towards that number. We have run the Markov chain for 10^5 iterations and we may see in all these three plots that all lines still have some slope downwards at the last iteration. According to this convergence diagnostic, it implies that



(a)



(b)



(c)

Figure 3.11: Convergence average number of edges for the 10k dataset. Legends are ordered with lines.

the chain has not converged. However, it is already possible to observe which approach has a faster convergence.

In all three plots, the fastest convergence is obtained by using either an RCARR or an RCARNR neighborhood. We may appreciate that the larger the cardinality of RCAR is, the larger the difference between the use of RCAR and any of the other *non-RCAR* neighborhoods. In the particular case of cardinality 4, although RCARR4 clearly outperforms the rest, RCARNR4 shows the slowest convergence. This may be due to a sequence of jumps that led the Markov chain into an area of local maxima from which it is very improbable to escape.

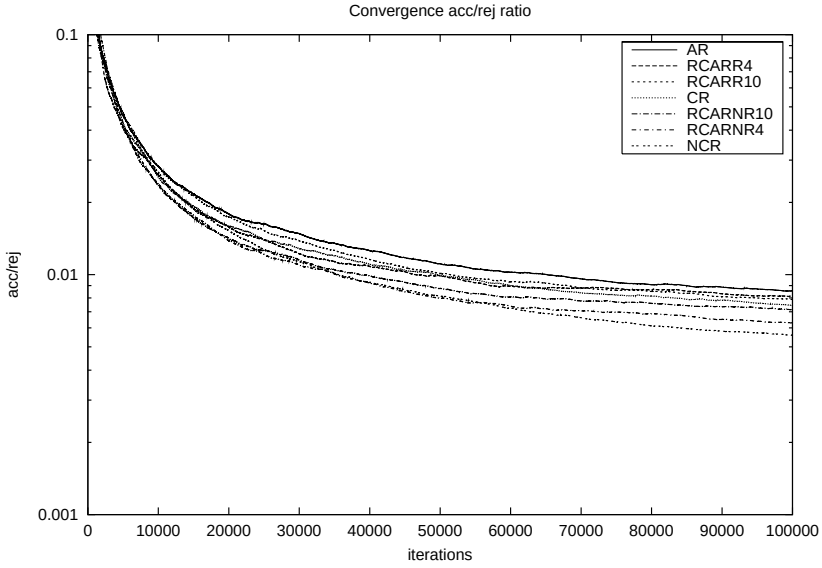
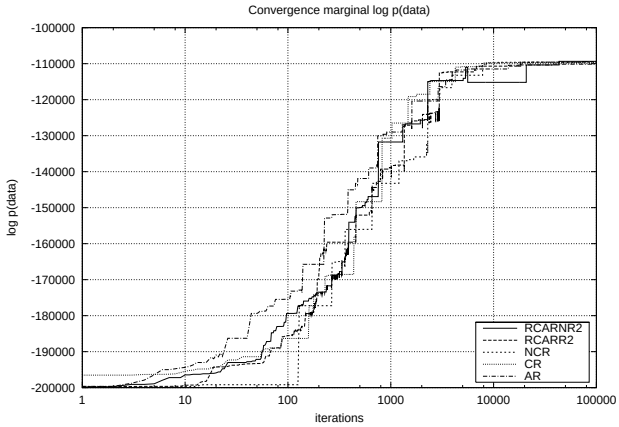


Figure 3.12: Convergence accepts/rejects ratio for the 10k dataset. Legends are ordered with lines.

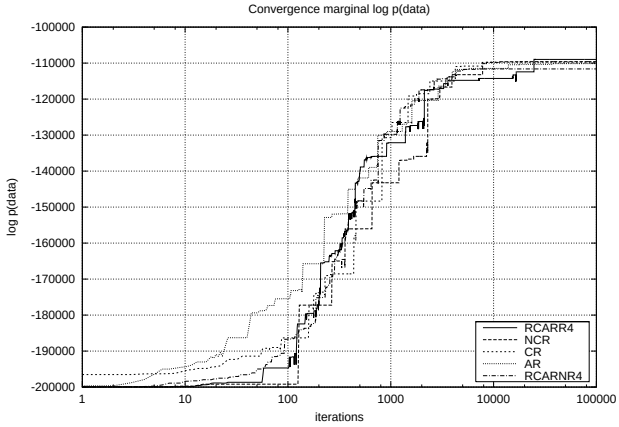
We can see this effect by examining the convergence of the ratio of accepts over rejects, in Figure 3.12, where RCARNR4 has one of the lowest ratios, only larger than NCR. We did not include RCARR2 and RCARNR2 but they are larger than RCARNR4. Note from this convergence diagnostic that the lines still have some slope, so we can also conclude from this diagnostic that the chain did not converge.

The next convergence diagnostic, the marginal of the data $p(D)$ (in log form), is in Figure 3.13. In Figure 3.14 we will find the same plots with the marginal range zoomed for the higher values. Here a larger $p(D)$ indicates that the Markov chain moves in an area of a higher posterior and therefore yields faster convergence. Again RCARR and RCARNR outperform CR, NCR and AR neighborhoods. We observe that for the RCAR 4, the slow convergence shown in plot 3.11b is in agreement with a low $\log p(D)$.

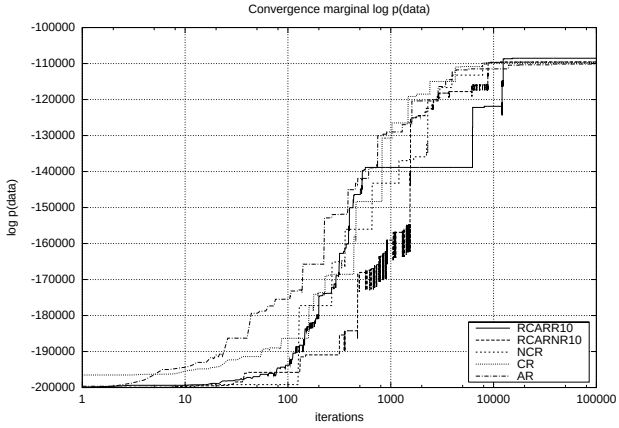
Another interesting fact is in the slope of the curve for the AR neighborhood when we look at the entire length of the Markov chain (left plots). We may appreciate that using the AR neighborhood, $p(D)$ increases faster than any of the others. This means that the AR neighborhood performs larger steps in the search space, but then later seems to get stuck in worse local maxima than using RCARR or RCARNR neighborhoods.



(a)

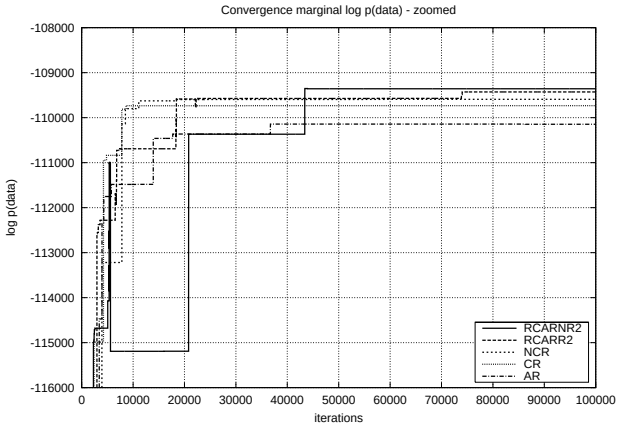


(b)

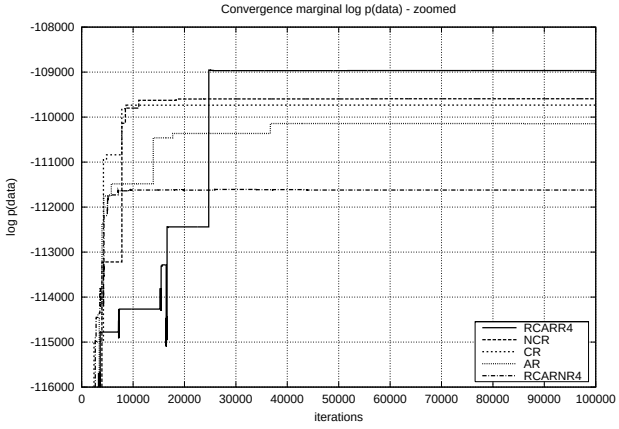


(c)

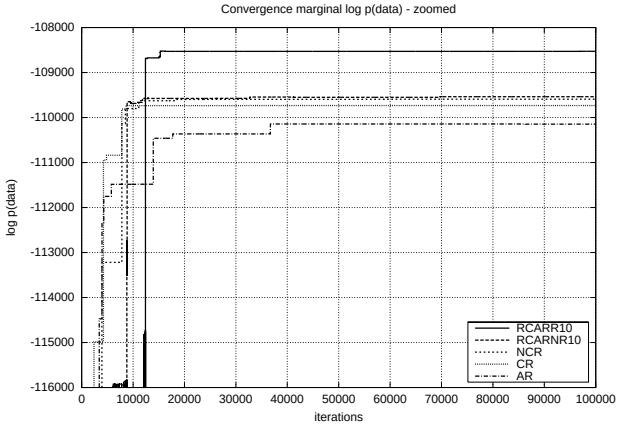
Figure 3.13: Convergence of the marginal of the data (entire marginal range). Comparison between AR, CR, NCR, RCARNR and RCARR. Legend is ordered with lines.



(a)

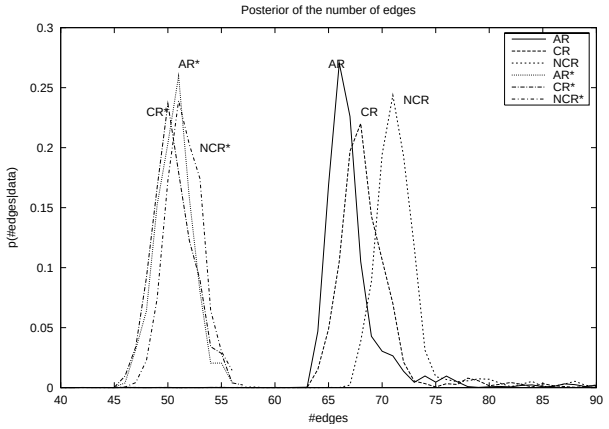


(b)

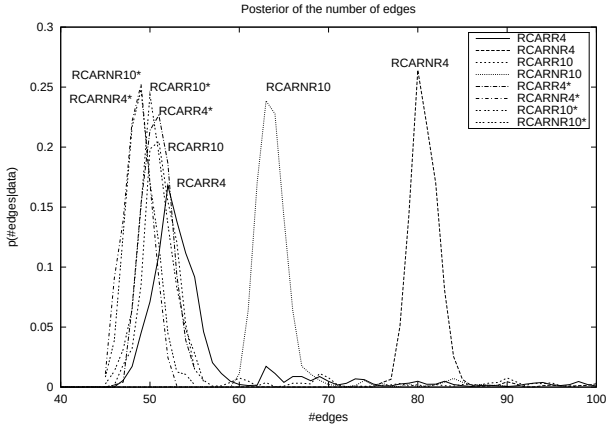


(c)

Figure 3.14: Convergence of the marginal of the data (zoomed marginal range). Comparison between AR, CR, NCR, RCARNR and RCARR. Legend is ordered with lines.



(a)



(b)

Figure 3.15: Comparison convergence ability.

In Figure 3.15 we have the posterior distribution of the different numbers of edges. In this case, we have started the Markov chain from both the empty DAG model and the almost true Alarm network, which has been noted with an asterisk next to the name of the corresponding neighborhood. In plot 3.15a we can see the runs for the AR, CR and NCR neighborhoods. Clearly, those that started at the almost true Alarm network show a faster rate of convergence than those that did not.

In plot 3.15b we see the runs for cardinalities 4 and 10 of the RCARNR and RCARR neighborhoods. In contrast with the previous case, here two of them, namely RCARR4 and RCARR10, are able to provide distributions quite similar to those of the ones that started in the almost true alarm network. With this diagnostic, we end the analysis of the convergence of the MC^3 algorithm and we can conclude that the RCAR operation improves substantially the rate of convergence of the MC^3 algorithm.

A further interesting outcome of our experimentation arises from looking at the num-

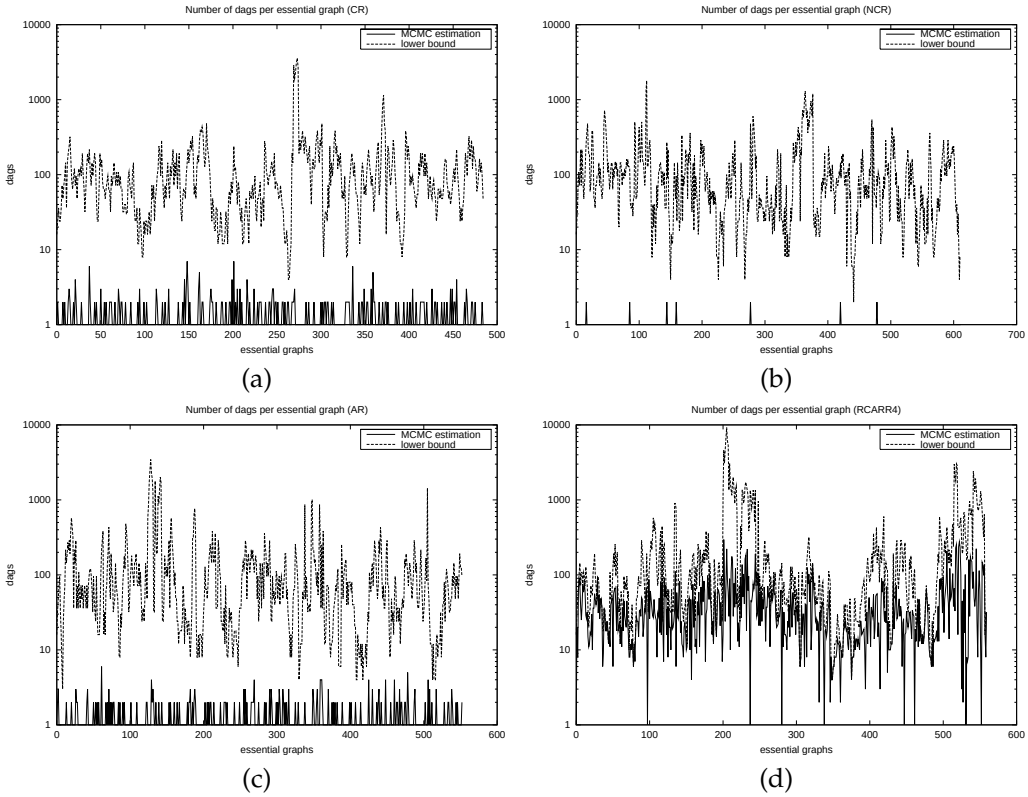


Figure 3.16: DAGs per essential graph.

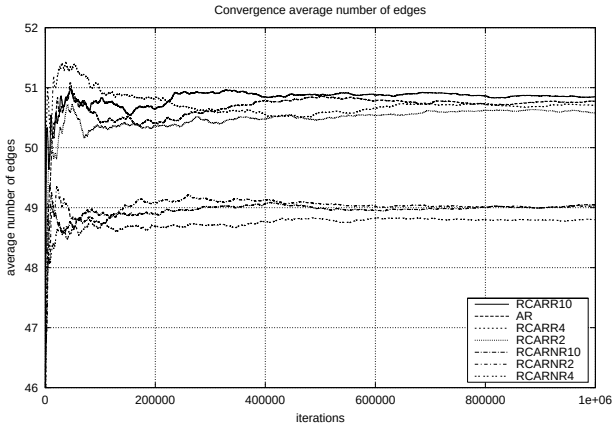
ber of members of each equivalence class of DAG Markov models, visited by the Markov chain during the 10^5 iterations. For comparison, we have computed a lower bound on the size of each equivalence class, as follows.

Let $\mathbf{M}(G)$ be a DAG Markov model. Let $\mathbf{E}(G^*)$ be its corresponding EG Markov model, i.e. $\mathbf{M}(G) = \mathbf{E}(G^*)$, where G^* is the essential graph representation of G. Let G^* have m connected components, where each of them has ρ_i reversible edges. The lower bound on the number of members of the equivalence class represented by G^* is

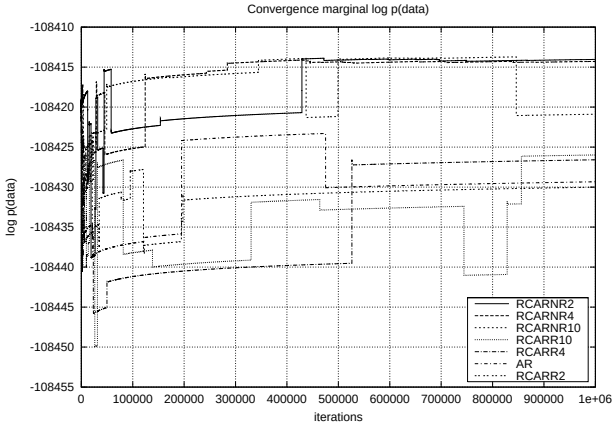
$$\prod_{i=1}^{m}(\rho_i + 1).$$

In Figure 3.16 we see the plots of these numbers for the runs that started on the almost true Alarm network. In plots (a) and (b) we see CR and NCR, where in this latter case there are no more than two members visited due to the nature of the non-covered reversal operation.

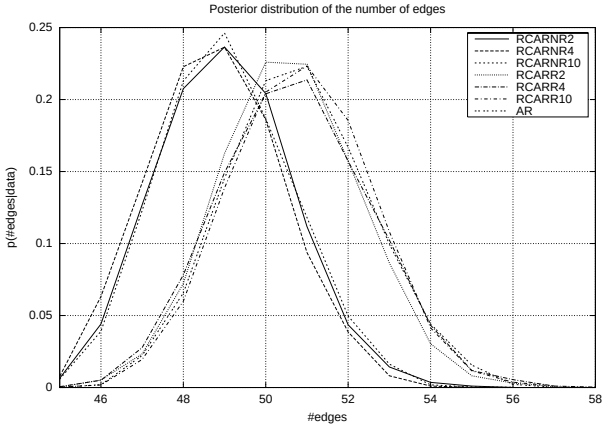
In plots (c) and (d) we see that AR visits a similar amount of members per class as the CR neighborhood and RCARR4 can estimate the number of members of each equivalence class much better. This is the empirical evidence for Theorem 3.4, which explains



(a)



(b)



(c)

Figure 3.17: Convergence diagnostics for a Markov chain of length 10^6 iterations, starting from the almost true Alarm network. Legends are ordered with lines.

why the RCAR operation enhances both heuristic search and the MCMC method.

The length of Markov chain we have used (10^5 iterations) is long enough as to clearly distinguish among different rates of convergence. However, all the diagnostics show lack of convergence of the chain. We have run a longer chain for 10^6 iterations starting from the almost true Alarm network. The convergence diagnostics are in Figure 3.17.

In this case we do not include the CR and NCR neighborhoods for comparison. In plot 3.17a we find the average number of edges during the run. In plot 3.17b we find the convergence of the marginal of the data. In plot 3.17c we find the posterior distribution of the cardinalities of the corresponding sets of edges. In all these three diagnostics we can distinguish two groups that converge at a different rate. One formed by AR, RCARR10, RCARR4 and RCARR2, and another by RCARNR10, RCARNR2 and RCARNR4. The second group shows better convergence.

In a way, this is surprising because, as we had seen so far, RCARR was outperforming RCARNR. Now, it is clear that starting from the almost true Alarm network and running that long, RCARNR converges faster. Nevertheless, this behavior should not be surprising to us, if we assume that Meek's conjecture (see Conjecture 3.1) holds. As we saw in Theorem 3.4, the ENR neighborhood provides the largest portion of the inclusion boundary among all concepts of neighborhood presented here. Recall that the RCARNR is an approximation of the ENR neighborhood. Thus, in theory we had no reason to believe that any of the other neighborhoods would improve the RCARNR neighborhood. Only we could see empirically that when starting from the empty Alarm network, the RCARR neighborhood converges faster.

We may conclude that when the Markov chain is close to the model with largest likelihood, the non-covered reversal operation in the RCARR neighborhood leads the chain to local maxima that slows down the convergence. The accepts over rejects ratios, which we don't show here, are lower for RCARR neighborhoods, meaning precisely this effect, that there is a very low probability of escaping from where the chain is. This might be happening because the non-covered reversal always makes the chain jump to a model out of the inclusion boundary (see Theorem 3.4).

We conjecture that probably a better strategy would combine, during the run of the Markov chain, both the RCARR and RCARNR neighborhoods.

Finally, we will take a look at the computational overhead produced by the use of the RCAR operation within the MC^3 algorithm. In Figure 3.18 we have plotted the average number of iterations (lines 5 through 14 in Figure 3.10) per second. This plot corresponds to the runs of length 10^5 iterations. In addition to the legend, we have labeled the lines with their corresponding neighborhood and also included the total average ratio of accepts and rejects. A lower ratio speeds up the chain as it is making, on average, less moves.

For clarity we only report comparison between AR, RCARR4 and RCARR10, as the differences are similar for other combinations. Examining separately the runs that start from the empty DAG model and those that start from the almost true Alarm network, we see that, in both cases, using the RCAR operation is between two and three times slower than not using it.

This cost is similar to the one we observed for heuristic search, and we consider it to be a very good trade off. Madigan et al. (1996) extended the MC^3 algorithm to work directly in the space of equivalence classes of DAG models. They do not show the

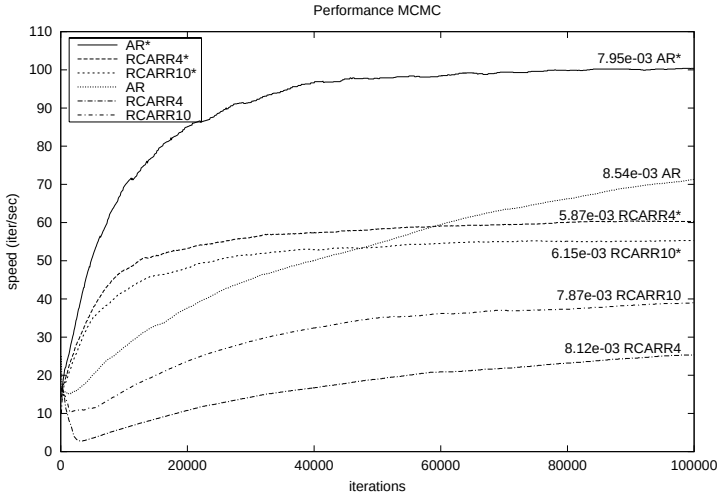


Figure 3.18: Performance.

computational overhead with respect to the space of DAG models. However, the fact that their Markov chain modified two adjacencies at a time, to guarantee irreducibility, suggests that their method is computationally more expensive than ours.

3.7 Concluding Remarks

In this chapter we have studied the problem of structural learning of DEC and DAG Markov models. We have surveyed the corresponding Bayesian score metrics and two of the most important model-based approaches to structural learning: heuristic search and the MCMC method.

The two main problems that a learning algorithm has to face are the sparseness of the data, and the local maxima in the search space for the combination of traversal operator and scoring function used in the learning process. While restricting ourselves to work with a higher amount of data alleviates the first problem, it may aggravate the second one. The way in which a learning algorithm traverses the search space is fundamental for the success of the algorithm in front of the previously mentioned problems. In order to investigate ways to improve current traversal operators we have introduced the notion of graphical Markov model inclusion and carried out a thorough study of the inclusion order in the class of DAG Markov models.

In this study we showed the relevance of the inclusion order to structural learning of GMMs. In the particular case of DAG models, this study combined theoretical results (Kočka, 2001) with a specific policy (Kočka and Castelo, 2001, the RCAR algorithm) that partially accounts for the inclusion order, to provide two new learning algorithms, the HCMC and the eMC^3.

The experiments with synthetic data show that the new algorithms improve the current approaches without compromising the computational performance.

Chapter 4

Data Mining

4.1 Introduction

In the first chapter we have mentioned that the database area has played an important role in the growth of data mining. This has been clearly the case in one of the most popular data mining algorithms: *association analysis*.

There is a trend in business to provide a *personalized* service to customers, by means of using barcodes, and many other mechanisms, that identify our transactions. This trend materializes in the storage of large amounts of information in the so-called *transactional databases*.

Transactional databases (see for instance (Han and Kamber, 2001)) store records formed by a unique transaction identifier and a list of items members of that transaction. A traditional example of a transaction is our shopping basket in a supermarket, when we use a customer loyalty card at the moment we pay such that, our identity and the list of products we buy, are registered.

Association analysis (Han and Kamber, 2001) is the discovery of *association rules*. An association rule, as we shall see in depth later, is a set of attribute-value conditions that occur frequently enough that these conditions become an interesting observation about what is stored in the database and, thus, what happens in the store.

This is an instance of one of the main differences between the models used in data mining, and those used in statistics. While in the former tend to describe local patterns also, the latter try to construct a global model only.

The induction of association rules poses several difficult problems regarding the computational complexity of extracting such rules from large databases. Many algorithms have been developed for such purpose and in this chapter we will see how GMMs provide a new perspective for this problem. This new perspective opens new ways of tackling the problem of retrieving association rules from databases. The approach is interesting in the sense that combines the use of a global model to guide the process of finding an interesting local pattern.

In the next section we will describe in detail what association rules are, and existing algorithms for their retrieval, and in section 3 we will introduce a new algorithm for such purpose (Castelo et al., 2001). Finally, we will summarize the important points of this chapter.

4.2 Association Rules

Let $\mathcal{U} = \{I_1, I_2, \ldots, I_m\}$ be a finite set of available items. A transaction is a subset $\mathcal{T}_i \subseteq \mathcal{U}$ uniquely identified by some index i, which may correspond, for instance, to a shopping basket purchased in a supermarket. Given two disjoint non-empty subsets of items $\mathcal{A}, \mathcal{B} \subseteq \mathcal{U}$, an association rule $\mathcal{A} \rightarrow \mathcal{B}$ indicates cross-sell effects on the items in $\mathcal{A}$ and $\mathcal{B}$ across the transactions $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N$.

For each item consider a binary random variable Z_i which, in a given transaction, will take the value 1 if the i_{th} item belongs to the transaction, and 0 otherwise. In this way we can represent a particular set of items by a particular assignment $\mathbf{Z} = \mathbf{z}$, where as in previous chapters, $\mathbf{Z}$ represents a set of variables and $\mathbf{z}$ is some instantiation of the product space $\times \mathcal{Z}_i$, $\mathcal{Z}_i = \{0, 1\}$. Now, we can write an association rule as

$$\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y},$$

where $\mathbf{X}, \mathbf{Y} \subset \mathbf{Z}$. For simplicity, we will neglect the instantiated values $\mathbf{x}$ and $\mathbf{y}$ from the previous notation when their presence is not relevant in the discussion, thus just writing $\mathbf{X} \rightarrow \mathbf{Y}$.

In general terms, the problem of finding *interesting* association rules corresponds to the problem of finding assignments $\mathbf{Z} = \mathbf{z}$, that provide a high probability density in the joint probability distribution $p(\mathbf{Z})$. As pointed out in Hastie et al. (2001), this may be viewed as a problem of "mode finding" or "bump hunting" which reveals that the problem, as formulated, is intractable.

In order to provide a tractable solution to the problem of finding interesting association rules, one does not try to seek these particular assignments $\mathbf{z}_1, \ldots, \mathbf{z}_\mathbf{m}$ that concentrate the density of $p(\mathbf{Z})$. Instead one tries to find *regions* in the product space $\times \mathcal{Z}_i$ that provide a high probability relative to the amount of transactions that fall within those regions.

The *proportion* of transactions that fall within a particular region of the product space $\times \mathcal{Z}_i$, $\mathbf{Z} = \mathbf{z}$, is called the *support* of the items present throughout these transactions, and noted $\sup(\mathbf{Z} = \mathbf{z})$. A set of items, or *itemset*, $\mathbf{X} = \mathbf{x}$ is said to be *frequent* if its support exceeds some predefined support t, i.e. $\sup(\mathbf{X} = \mathbf{x}) \geq t$. Analogously, for a given association rule $\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}$, one says that the *support of this rule* is the proportion of transactions that contain all items in $\mathbf{X}$ and $\mathbf{Y}$, noted $\sup(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})$. Using the concept of support, the *confidence* of a rule is defined as:

$$\mathrm{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{\sup(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})}{\sup(\mathbf{X} = \mathbf{x})}. \tag{4.1}$$

Association rules were introduced by Agrawal et al. (1993) who provided the *Apriori* algorithm to learn them from data. Given some fixed support and confidence thresholds, t and c respectively, $0 < t, c < 1$, the Apriori algorithm works by creating a collection of frequent itemsets which are used later as building blocks to find interesting association rules whose confidence exceeds the corresponding threshold c.

More concretely, it begins by creating all single frequent itemsets. Then, the algorithm creates all frequent itemsets of size two using the single frequent itemsets previously created. The algorithm works in this way systematically, using previously encountered itemsets and discarding those newly created that do not meet the fixed threshold on

support t. This strategy follows from the realization that if an itemset is not frequent, then no superset of this itemset can be frequent.

Once we have a collection $\mathcal{F}$ of frequent itemsets, then the association rules are generated as follows:

1. For every frequent itemset $F \in \mathcal{F}$, create all non-empty subsets of F.
2. For every non-empty subset S of F, let $S = (\mathbf{X} = \mathbf{x})$ and $F \backslash S = (\mathbf{Y} = \mathbf{y})$. If $\text{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) > c$, then output the rule $\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}$.

Since the antecedent and the consequent of the rule are derived from a single frequent itemset, it follows immediately that the rule also satisfies the minimum support threshold t. A further measure typically used in this context to rank and filter interesting rules is an estimate of the association between the antecedent and the consequent, which is known as the *lift* and computed as follows:

$$\text{lft}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{\text{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})}{\text{sup}(\mathbf{Y} = \mathbf{y})}. \tag{4.2}$$

The lift is a positive quantity that has value 1 if there is no association between the antecedent and the consequent, and we will be interested in those association rules with a high lift.

Examples of the application of the Apriori algorithm, more detailed explanations and enhancements for improving performance may be found in (Agrawal et al., 1995; Han and Kamber, 2001). A different approach for rule extraction, may be found in (Goodman et al., 1992).

4.3 Association Rules based on Conditional Independencies

The support pruning by Apriori poses a serious problem when applying association rules in practice. Rules with high support are in general already known, while using a (very) low support threshold causes two problems. Firstly, it results in very many rules, most of which are uninteresting. The second, as pointed out by Hastie et al. (2001, pg. 444), is that the number of frequent itemsets and their sizes can grow exponentially.

For the first problem, many intestingness measures, such as the lift, have been defined, that can be used to filter or order the resulting association rules. The computational problem, however, has proven to be more difficult. In fact, the support threshold was introduced for computational reasons because a high support threshold substantially reduces the number of possible frequent itemsets.

From a purely data analysis point of view, pruning on support is therefore an artificial device unrelated to what the data may reflect. A sensible approach is to devise algorithms that do not rely on support pruning to find interesting association rules. Two recent works in this direction are (Silverstein et al., 1998; Cohen et al., 2001). In this section we will introduce another approach that does not rely on support pruning.

We propose to use the relationships of conditional independence among the random variables representing the items, to make the discovery of association rules computationally feasible. As we already discussed in previous chapters, conditional independence

plays an important role in the mechanism that generates the data, therefore it is an appealing way to restrict the set of association rules considered.

Recall the definition of conditional independence (see Definition 2.1), under which two non-empty sets of variables $\mathbf{X}$ and $\mathbf{Y}$ are said to be conditionally independent given a third set $\mathbf{Z}$ if for all configurations $\mathbf{x}, \mathbf{y}, \mathbf{z}$ satisfying $p(\mathbf{Z} = \mathbf{z}) > 0$, it holds that

$$p(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) = p(\mathbf{X} = \mathbf{x} | \mathbf{Z} = \mathbf{z}). \tag{4.3}$$

This relationship was noted as $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$, and the notion behind is that if we know $\mathbf{Z}$, any further information about $\mathbf{Y}$ cannot enhance the current state of information about $\mathbf{X}$, i.e. given $\mathbf{Z}$, $\mathbf{Y}$ becomes irrelevant to $\mathbf{X}$.

Since the support for a given itemset $\mathbf{X} = \mathbf{x}$ is defined as the proportion of transactions that contain the items in $\mathbf{X} = \mathbf{x}$, it can be interpreted as an estimate of the probability of $\mathbf{X} = \mathbf{x}$, $p(\mathbf{X} = \mathbf{x}) = \sup(\mathbf{X} = \mathbf{x})$. The support of a rule is then also an estimate of the joint probability of the antecedent and consequent, $p(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = \sup(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})$.

Consequently, the confidence (4.1) of a rule is an estimate of the conditional probability $p(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) = \mathrm{cnf}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y})$. Finally, the lift (4.2) of a rule is therefore an estimate of the following ratio,

$$\mathrm{lft}(\mathbf{X} = \mathbf{x} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{p(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x})}{p(\mathbf{Y} = \mathbf{y})}. \tag{4.4}$$

.

If we know that $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$, then by the factorization in (4.3), we know that the previous ratio for the rule $\{\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}\} \rightarrow \mathbf{Y} = \mathbf{y}$ can be rewritten as follows

$$\mathrm{lft}(\{\mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z}\} \rightarrow \mathbf{Y} = \mathbf{y}) = \frac{p(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z})}{p(\mathbf{Y} = \mathbf{y})} = \frac{p(\mathbf{Y} = \mathbf{y} \mid \mathbf{Z} = \mathbf{z})}{p(\mathbf{Y} = \mathbf{y})}.$$

In a nutshell, the lift does not rise by adding knowledge about $\mathbf{X}$, $\mathbf{X}$ is irrelevant to $\mathbf{Y}$ given $\mathbf{Z}$. Or, if we have an association rule for $\mathbf{Y}$ with $\mathbf{X}$ and $\mathbf{Z}$ on the lefthand side, we might as well filter $\mathbf{X}$ out.

This is interesting if $\mathbf{Z}$ *shields* $\mathbf{Y}$ from all other variables, i.e. if $\mathbf{Y} \perp\!\!\!\perp \mathbf{U} \backslash (\mathbf{Y} \cup \mathbf{Z}) \mid \mathbf{Z}$ where $\mathbf{U}$ denotes the entire set of variables. Because then we only have to consider association rules whose lefthand side is within $\mathbf{Z}$. All of this is even more interesting if $\mathbf{Z}$ is *minimal*, i.e. if we remove an element from $\mathbf{Z}$ it no longer shields $\mathbf{Y}$ from the rest. Such a minimal set is called a *Markov blanket* of $\mathbf{Y}$ (Pearl, 1988).

To keep our discussions simple, we will assume that the consequent of the association rule is a singleton, as $\mathbf{X} = \mathbf{x} \rightarrow Y = y$. The ideas and procedures explained here are easy to adapt afterwards to association rules with a consequent of an arbitrary cardinality.

We want to build an oracle that can answer queries over the validity of CI statements of the form $Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$ in a given dataset. Or more precisely, whether $\mathbf{Z}$ is the Markov blanket of Y for a given dataset. The problem that immediately arises is that we do not have the probability distribution but a dataset sampled from it. Thus, there is an implicit uncertainty in the validity of such a CI restriction.

In order to account for this uncertainty we adopt a Bayesian approach and use as oracle a posterior distribution over the set of possible Markov blankets of a given consequent Y for a rule, i.e. CI restrictions of the form $Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$.

To carry out this idea we will make use of DEC Markov models to handle CI restrictions and the MCMC method to compute the posteriors. For a given vertex v in a DEC Markov model, its Markov blanket corresponds to its boundary $bd(v)$ (see chapter 2). So, we can compute the posterior of a CI statement $I \equiv Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$ given the data as follows:

$$p(I|D) = \sum_{M \in \mathcal{M}} p(I|M, D) p(M|D),$$

which corresponds to the computation of a quantity of interest Δ as indicated in expression (3.28). In the present context, $\mathcal{M}$ corresponds to the class of DEC Markov models, which we use here as a device to handle CI restrictions.

The general procedure to learn association rules is as follows. First, we build an oracle which consists of pairs (I, p) where I is a CI restriction of the form $Y \perp\!\!\!\perp \mathbf{U} \backslash (Y \cup \mathbf{Z}) \mid \mathbf{Z}$ and p is its posterior probability given the data. These pairs will be obtained from the MCMC output.

Second, for every variable, we consider some maximum number of Markov blankets from the corresponding posterior distribution. For each of the Markov blankets, we generate association rules with a subset of the Markov blanket as lefthand side. There are further details to take into account which we will explain now. The complete algorithm, called Mambo (Maximum A posteriori Markov Blankets using an Oracle), is in Figure 4.1. The Mambo algorithm takes the following four parameters:

- *nmb*: maximum number of Markov blankets.
- $\mathbf{Y}$: set $\{Y_1, \ldots, Y_n\}$ of random variables from the dataset.
- *dc*: minimum improvement in confidence for a superrule.
- *dl*: minimum improvement in lift for a superrule.

It has a main loop that goes through lines 1 to 11. At each iteration it will create all possible association rules with the variable Y_i on the righthand side. The association rules created in each iteration are temporarily stored in *pres*, which is initialized in line 3. In line 4 the function oracle$(\cdot)$ is called which returns a set of, at most *nmb*, Markov blankets for the variable Y_i.

The loop that iterates through lines 5 to 9 considers each of these *nmb* Markov blankets, noted $\mathbf{Z}$. The inner loop iterates through all possible subsets of the Markov blanket $\mathbf{Z}$, i.e. the power set $\mathcal{P}(\mathbf{Z})$ without the empty set. Each of those subsets is noted as $\mathbf{X}$ and is used to generate association rules of the form $\mathbf{X} = \mathbf{x} \rightarrow Y_i = y_i$ for all possible values $\mathbf{x} \in \mathcal{X}$ and $y_i \in \mathcal{Y}_i$. In this context $\mathcal{X}$ is the product space $\times \mathcal{X}_i$ where $\mathcal{X}_i$ are the levels of variable X_i and $\mathcal{Y}_i$ are the levels of variable Y_i. All these association rules are stored in the set *pres*.

The set $\{\mathbf{Z}_1, \ldots, \mathbf{Z}_{nmb}\}$ of *nmb* Markov blankets for a particular righthand side variable Y_i, may contain two Markov blankets $\mathbf{Z}_i = \{X_1, \ldots, X_m\}$ and $\mathbf{Z}_j =$

```
algorithm mambo(int nmb, set Y, flt dc, flt dl) returns set
01  set res = ∅
02  for Y_i ∈ Y do
03    set pres = ∅
04    set O = oracle(Y_i, nmb)
05    for Z ∈ O do
06      for X ∈ P(Z)\{∅} do
07        pres = pres ∪ {(X = x → Y_i = y_i) | x ∈ X, y_i ∈ Y_i}
08      enddo
09    enddo
10    res = res ∪ filter(pres, dc, dl)
11  enddo
12  return res
endalgorithm
```

Figure 4.1: The Mambo algorithm.

$\{X_1, \ldots, X_m, W_1, \ldots, W_k\}$, thus $\mathbf{Z}_i \subset \mathbf{Z}_j$. This may lead to generate the following two association rules:

$$\begin{array}{lcl} R_1 & : & \mathbf{X} = \mathbf{x} \rightarrow Y = y \\ R_2 & : & \{\mathbf{X} = \mathbf{x}, \mathbf{W} = \mathbf{w}\} \rightarrow Y = y \end{array}$$

In this circumstance we will say that R_2 is a superrule of R_1. Given an association rule, not all of its superrules are necessarily of interest. In particular, if they have similar confidence and lift. Then, we will prefer the smaller rule since it is more general. For this reason, in line 10 of the Mambo algorithm, we call the function filter$(\cdot)$ which removes from the initial set *pres* all those superrules that do not improve confidence and lift in the given quantities *dc* and *dl*. The resulting filtered set is added to the final set of association rules *res*, which after all iterations of the main loop is returned. The filtering algorithm is specified in Figure 4.2 and takes three parameters:

- S: set of association rules to filter.
- dc: minimum improvement in confidence for a superrule.
- dl: minimum improvement in lift for a superrule.

The algorithm first copies the input set of association rules into the set F on which the filtering (removal) operations will take place. It has a main loop from line 2 till line 31 that goes through every association rule in F. It starts by storing the confidence and lift of the current association rule $\mathbf{X} = \mathbf{x} \rightarrow Y = y$ in c^* and l^* respectively, which are used later on.

From lines 5 till 15 it checks whether the association rule $\mathbf{X} = \mathbf{x} \rightarrow Y = y$ is a superrule in F that does not improve confidence and lift in dc and dl. In such case the boolean variable *sr* takes the truth value and then in line 17 this rule is removed from the

```
algorithm filter(set S, flt dc, flt dl) returns set
01  set F = S
02  for (X = x → Y = y) ∈ F do
03    flt c* = cnf(X = x → Y = y)
04    flt l* = lft(X = x → Y = y)
05    bool sr = false
06    set Q = {(Z = z → W = w) ∈ F | W = Y ∧ w = y}
07    for (Z = z → W = w) ∈ Q and ¬sr do
08      if Z ⊆ X and z ⊆ x then
09        flt c = cnf(Z = z → W = w)
10        flt l = lft(Z = z → W = w)
11        if c* − c ≤ dc and l* − l ≤ dl then
12          sr = true
13        endif
14      endif
15    endfor
16    if sr then
17      F = F \ {X = x → Y = y}
18    else
19      set R = ∅
20      for (Z = z → W = w) ∈ S do
21        if Z ⊇ X and z ⊇ x then
22          flt c = cnf(Z = z → W = w)
23          flt l = lft(Z = z → W = w)
24          if c − c* ≤ dc and l − l* ≤ dl then
25            R = R ∪ {Z = z → W = w}
26          endif
27        endif
28      endfor
29      F = F \ R
30    endif
31  endfor
32  return F
endalgorithm
```

Figure 4.2: The filtering function for the Mambo algorithm.

set F. Otherwise, the rule will remain in F but all other rules in F that are superrules of this one, and do not improve confidence and lift in dc and dl, should be removed from F. This is implemented in lines 19 through 29 where the set R stores such rules and it is finally removed from F in line 29.

4.3.1 Experimental Results

We have run the Apriori and Mambo algorithms over a dataset in order to show the differences between our approach and a minimum support driven approach. The comparison is not intended to show that one is "better" than the other, but rather to illustrate where the differences lie with our conditional independence driven approach.

We used the insurance company benchmark of the COIL 2000 challenge, available from the UCI KDD archive (`http://kdd.ics.uci.edu`). This dataset contains information about the customers of an insurance company, and consists of 86 variables concerning product ownership and socio-demographic characteristics derived from postal area

codes. For this experiment we ignored the socio-demographic variables (1-43). The remaining variables consist of the target variable of the original challenge (whether or not someone owns a caravan policy) and variables concerning the ownership (number of policies) and contribution (money amount) for 21 other insurance products.

In order to make the data suitable for the analysis with binary association rules, we considered only the product ownership variables coding with a value of 1 if the product is owned and 0 otherwise. This leaves us with a dataset consisting of 22 binary variables for a total of 5822 customers. Table 4.1 provides a description of the variables and the corresponding relative frequencies of product ownership.

Table 4.1: Variables in the insurance dataset.

Var.	Description	Freq.
1	private third party insurance	0.402
2	third party insurance (firms)	0.014
3	third party insurance (agriculture)	0.021
4	car policy	0.511
5	delivery van policy	0.008
6	motorcycle/scooter policy	0.038
7	lorry policy	0.002
8	trailer policy	0.011
9	tractor policy	0.025
10	agricultural machines policy	0.004
11	moped policy	0.068
12	life insurance policy	0.050
13	private accident insurance policy	0.005
14	family accidents insurance policy	0.007
15	disability insurance policy	0.004
16	fire policy	0.542
17	surfboard policy	0.001
18	boat policy	0.006
19	bicycle policy	0.025
20	property insurance policy	0.008
21	social security insurance policy	0.014
22	mobile home insurance policy	0.060

In the analysis with the Apriori algorithm we used a minimum support of $t = 0.01$ that corresponds to 59 records. All rules with 1 item on the righthand side were generated from the frequent itemsets found. This yielded a total of 102 association rules.

The oracle was created by running the MC^3 algorithm on the class of DEC Markov models defined on the 22 variables, through 10^6 iterations which seemed to be sufficient to achieve convergence.

We have run the Mambo algorithm with a number of variations in the parameter values in order to illustrate the effect of these different parameters. In the most basic run we selected for each variable only the Markov blanket with highest posterior probability, i.e.

$nmb = 1$ (see Figure 4.1). Furthermore we only consider positive rules for the moment, i.e. rules of type $\mathbf{X} = \mathbf{1} \rightarrow Y = 1$. In general however, values in the antecedent and consequent of the rule may be either 0 or 1. For an initial comparison with the Apriori results we further selected only those rules with support at least 0.01. It is important to note that this was done afterwards, since the minimal support is not used in Mambo to constrain the search.

In order to consider the effect of filtering the superrules that do not improve confidence and lift, we performed the initial analysis without filtering them. This yielded a total of 72 rules, 30 less than Apriori (since we selected rules with support at least 0.01, all 72 rules were also found by Apriori). To illustrate why some rules generated by Apriori were not considered by Mambo we look to an example.

For variable 9 we found that the Markov blanket with highest posterior probability (0.4) is $\{3, 10\}$. One of the rules found with Apriori is $\{3, 16\} \rightarrow 9$ (we use this notation as a shorthand for $\{3 = 1, 16 = 1\} \rightarrow 9 = 1$) with confidence 0.64, lift 26 and support 0.013. This rule was not found with Mambo since the antecedent contains variable 16 which does not belong to the best Markov blanket. The rule $3 \rightarrow 9$ with confidence 0.62, lift 25 and support 0.013 was found by Mambo because its antecedent is a subset of the best Markov blanket.

If we filter superrules that do not improve confidence by 0.05 and lift by 0.1, the number of rules is reduced from 72 to 60. For example, one of the 72 rules found is $\{4, 3\} \rightarrow 16$ with a lift of 1.83. There exists a subrule $3 \rightarrow 16$ however, with a lift 1.80. Therefore the superrule is filtered out from the ruleset.

Now, as mentioned, with Mambo we can find interesting rules regardless of their support. For example, the best rule for product 4 found with Apriori is $\{1, 6, 22\} \rightarrow 4$ with a confidence of 0.88 and lift 1.7. With Mambo (without any support restrictions) we find the rule $\{21, 1\} \rightarrow 4$, with confidence 0.96, lift 1.9, and support 0.008 (46 records). This rule has higher confidence and lift than the best rule found with Apriori, but does not meet the minimum support requirement of 0.01.

Table 4.2: The number of rules found by Mambo for different values of *nmb*, with removal of superrules. Only rules with support at least 0.005 have been counted. The last row also counts rules with zero values in the antecedent or consequent.

	nmb		
ruletype	1	2	5
positive	78	78	85
all	518	549	750

It is interesting to note that setting the maximum number of Markov blankets *nmb* to 5 instead of 1, does not lead to a dramatic increase of the number of rules (see Table 4.2). For example, if we consider only positive rules and remove superrules that do not meet the minimum improvement requirements previously specified, we obtain a total of 78 rules with *nmb* set to 1 (selecting only the rules that have support at least 0.005). If we use the same settings, but select the 5 most probable Markov blankets, we obtain a total of 85 rules. The 7 additional rules found are of high quality in the sense

that they all have a lift larger than 1. One of the additional rules found with $nmb = 5$ is $\{16, 21\} \rightarrow 1$, with confidence 0.72, lift 1.8 and support 0.007. This rule was not found with $nmb = 1$ since the Markov blanket for 1 with highest posterior probability (0.78) is $\{2, 3, 4, 11, 12, 16\}$, i.e. it does not contain 21. However, the Markov blanket with third highest posterior probability (0.036) is the same set with 21 added (see Figure 4.3). Furthermore, $\{16, 21\} \rightarrow 1$ is not pruned by a subrule since it improves the lift as required on the subrules $16 \rightarrow 1$ (lift 1.6) and $21 \rightarrow 1$ (lift 1.5).

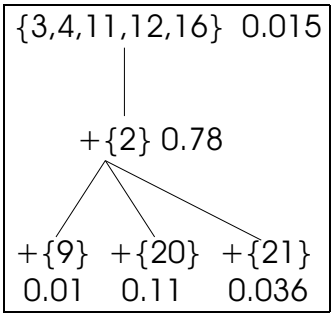


Figure 4.3: The 5 best Markov blankets for variable 1. Only additional elements to the set on top are shown. Posterior probability is shown to the right of or below a set.

Looking at Figure 4.3 we can understand why not very many rules are added when *nmb* is increased from 1 to 5. For example, the set at the top is a subset of the best blanket, and can therefore not generate any additional rules. The sets at the bottom have one element additional to the best set and could generate additional rules. Many of these rules are removed however because they are only slightly better, or even worse, than one of their subrules. For example, $\{4, 16\} \rightarrow 1$ (lift: 1.9) is generated from the best Markov blanket. Later the rule $\{4, 16, 21\} \rightarrow 1$ (lift: 1.9) is considered through the third best Markov blanket but immediately filtered because it has the same lift as the subrule just mentioned.

4.4 Concluding Remarks

Selecting *interesting* association rules is a very difficult problem as in large databases the amount of them, meeting frequency thresholds, can be huge. As Hand et al. (2001) point out, the use of significance testing methods to distinguish interesting rules is problematic.

We have presented an algorithm for the discovery of association rules that is driven by conditional independence relations between the variables rather than minimum support restrictions. We consider this to be an interesting alternative because for some applications minimum support is a rather artificial constraint primarily required for computational reasons.

Making use of conditional independencies is an intuitively appealing way to restrict the set of association rules considered, because CI restrictions are part of the mechanism that generates the data. Furthermore, the information concerning conditional independencies between variables may be of interest in its own right.

We have analyzed an insurance dataset with Mambo and illustrated the differences with a minimum support driven approach. It is ultimately an empirical question whether rules based on CI restrictions or rules based on minimum support restrictions are to be preferred for a particular application. If the rules are used exclusively for descriptive purposes, it makes sense to consult domain experts for the assessment of rule quality. If the rules are used for predictive purposes as well one can compare the predictive accuracy of the rules obtained with the different approaches.

Chapter 5

Applications

5.1 Introduction

In this chapter we will show three applications on real-world data of the Bayesian approach to learning DEC models. In particular we will be using the MC^3 algorithm introduced in chapter 3. We have not used DAG models as it is difficult to assess what variables occur on different footing for these datasets. Therefore, we take a more conservative approach and assume that the data generating mechanism sets all variables on equal footing.

The first application (Giudici and Castelo, 2001a) is on data that comes from the world wide web. The second (Giudici and Castelo, 2001b) on data from a medical domain. The third (Giudici and Castelo, 2001b), and last one, is on data from a supermarket. We will see that the three datasets have an increasing degree of sparseness which implies an increasing degree of difficulty in the analysis. The Bayesian approach used here, however, can be used in all three scenarios in the same way and, as we shall see, it provides a great insight into the analyzed data.

5.2 Web Mining

The term *Web Mining* refers to the task of analyzing data that describes the accesses through the internet to a website. The explosion of the electronic commerce through the internet has made this particular type of data an invaluable source of corporate information about customers. For a given company with presence on the internet, its competitors may be just one click away from the pages of its website. Therefore, it becomes crucial to understand how users visit the pages that form the website in order to improve the so-called *customer relationship management*, or CRM.

We consider a model-based approach to Web Mining, in order to carry out an explorative study that will help understand the way people interact with a website. With the exception of webcrawlers there is usually a person involved so we may assume there is a purpose in the visit to the website and it makes sense to believe there is a correlation between the hits registered. An analysis of pairwise correlations, as done in a typical association analysis, provides some information about how people access the website.

Still more information may be gathered by examining higher order correlations and conditional independencies.

The dataset we have used corresponds to all visited pages on the web site `http://www.microsoft.com` by a total population of 5000 anonymous visitors, randomly chosen. This dataset is available through the UCI KDD archive (`http://kdd.ics.uci.edu`). Each visitor is identified with a number going from 10000 to 15000 and no personal information is given. For each visitor the number of visits to each page of the site in a week of February 1998 is recorded. Site pages are identified with a number, which corresponds to an address (e.g. 1057 corresponds to "MS Power Point News"). This data does not contain information about the click order of the visitor.

For illustrative purposes, we report here a piece of the log-file. The codes for for visitors are denoted by C, and for vroots (pages) denoted by V.

```
C, "10908", 10908
V, 1108
V, 1017
C, "10909", 10909
V, 1113
V, 1009
V, 1034
C, "10910", 10910
V, 1026
V, 1017
```

This log-file in particular is practically *ready* to analyze, but in general, log-files consist of an much more entangled amalgamation of codes and identifiers. Their transformation into a clear format that can be processed by some mining algorithm is in itself a very difficult task.

Note that a client typically visits only a few different pages; in fact the average number of visited pages per client is four. This leads to a rather sparse contingency table, when the data is arranged according to all categorical variables corresponding to the pages (vroots). Indeed, to alleviate the problem, we have eliminated all clients who had visited only one page from the dataset.

This data was first analyzed by Breese et al. (1998) with the purpose of establishing a comparison among different predictive algorithms for collaborative filtering. As stated in that paper, collaborative filtering uses previously stored information about user preferences, to predict possible future user behavior.

It is not our goal here to mine data from the web in order to predict web usage, but rather to gather insight into the way that users access the website. This will be achieved by means of two types of exploratory graphical modeling strategies, based respectively on a the frequentist and the Bayesian statistical approach.

The use of GMMs for Web Mining has also been illustrated by Heckerman et al. (2000), but in the framework of *dependency networks* and greedy search.

If pages are arranged by absolute frequency of visits, their distribution is heavily asymmetric. In Figure 5.1, the most visited page is number 1008, "Free Downloads". To understand associations between different pages determined by the visits of the clients, we build up a contingency table that has, as leading descriptors, as many random variables as considered pages. However, given the very high number of pages in the site,

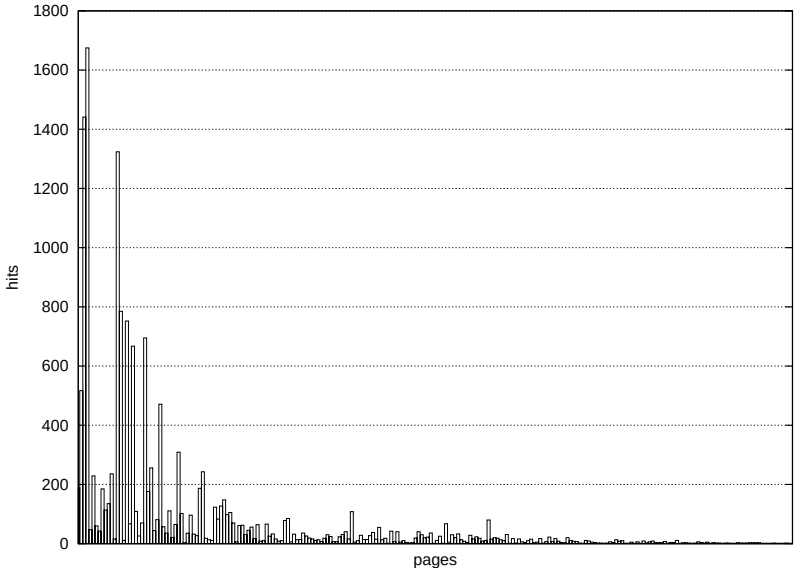


Figure 5.1: Frequency distribution of the pages.

and the low average number of different visited pages, we need to group pages into homogeneous categories, in order to avoid a very sparse contingency table.

By organizing the data in a contingency table we are losing the information about the order in which the pages were seen, which might be highly interesting. In order to analyze such information, one should use some other technique for sequence rule analysis.

Grouping may be done in several ways, for instance statistically (see Breese et al., 1998). On the other hand, we considered a grouping that is logically sensible, and we identified eight groups: Programs, Catalogue, Internet, Entertainment, Office, Development, Windows and Initials.

We also binarized the variable corresponding to the grouped page, with the level 1 indicating that the group has been visited at least once, and level 0 indicating that the group of pages was not visited during the week. Of course, in so doing, there is a loss of information, but there is an advantage in ease of analysis and understanding.

The cross-classification of the 5000 clients into the eight groups of pages, with two levels per group (visited/not visited) produces a 2^8 contingency table. The marginal association between groups is measured by the marginal pairwise odds ratio. For each odds ratio we have calculated, besides the maximum likelihood estimate, an approximate 95% confidence interval. Two variables are declared significantly associated if a value of 1 for the odds ratio, corresponding to marginal independence, falls outside the confidence interval.

To illustrate the marginal associations, one can draw a graph whose nodes correspond to the groups and where edges are inserted between a pair of nodes if the two corresponding groups are significantly associated. Such a graph is not a GMM (in the

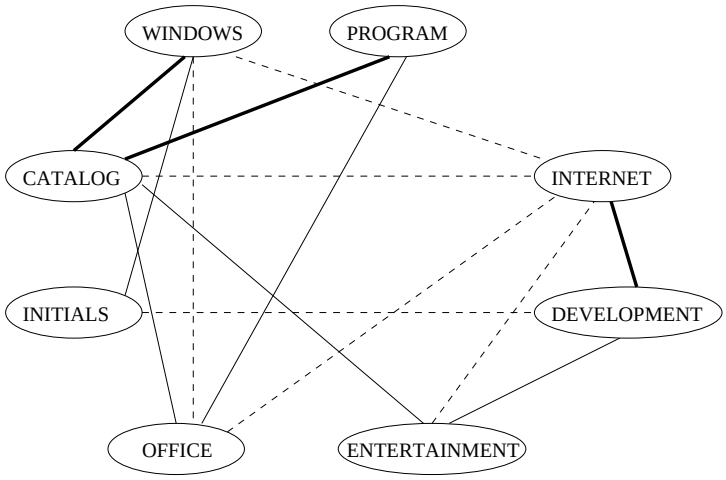


Figure 5.2: The exploratory graph: positive and negative associations are represented respectively by solid and dashed lines.

sense that no high order CI restriction is encoded in the graph), but a marginal independence graph, with no separation properties. However it sets a framework for subsequent analysis. Such a marginal independence graph is depicted in Figure 5.2.

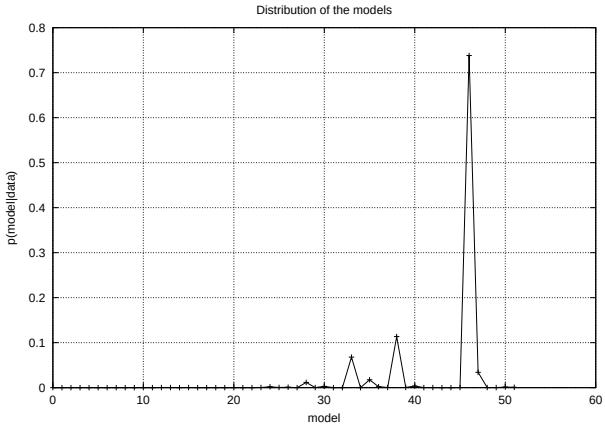
In this figure, solid lines represent positive associations and dashed lines negative associations. Those positive associations with an odds ratio greater than three, describing a strong association, are highlighted with a thick solid line. Note that the total number of edges estimated to be present is 14.

The preceding exploratory analysis is based on marginal independencies, as it considers all marginal two-way contingency tables corresponding to each pair of variables separately. A more correct study of the associations must consider directly the 2^8 joint contingency table with all variables being simultaneously analyzed. To achieve this we need a more structured, model based approach such as the one that GMMs provide. We are going to explore the approximate posterior distribution of the space of DEC Markov models on the eight groups of pages of the website.

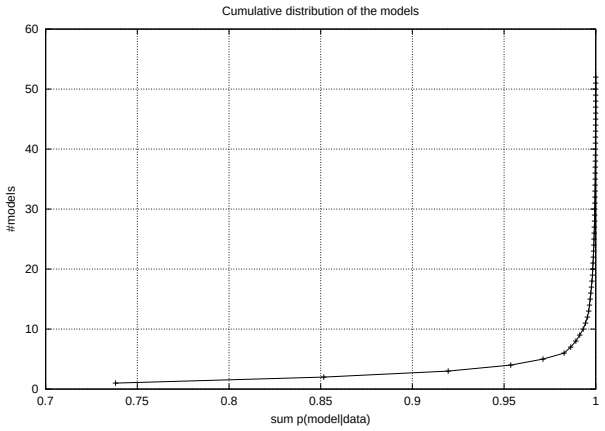
The posterior distribution on the space of models allows us to see how frequently the most likely model is better than the rest of the models. Also, it is interesting to see which parts of the model remain unchanged across those models that account for the largest portion of the distribution. Logically, these unchanged parts deserve a higher degree of confidence in the information they provide.

We have run the MC^3 algorithm for 10^5 iterations that provided convergence according to the diagnostics, which we do not show here. Figure 5.3a shows the posterior distribution of the models given the data. Next to it, in Figure 5.3b, we find this distribution accumulated, by ordering the models from larger to smaller probabilities. This latter plot allows to see the concentration of the distribution, and in this case we may see that only three models account for more than 90% of the distribution.

Figure 5.4 shows, from (a) to (c) in increasing probability, the three DEC Markov models that account for more than 90% of the probability distribution. These three mod-



(a)



(b)

Figure 5.3: Posterior and cumulative posterior distribution of the models.

els show that the page groups Internet (INT), Windows (WIN), and Office (OFF) separate Catalog (CAT) and Entertainment (ENT) from everything else in each of the graphs. Thus

$$\{\text{CAT}, \text{ENT}\} \perp\!\!\!\perp V \setminus \{\text{CAT}, \text{ENT}, \text{INT}, \text{WIN}, \text{OFF}\} \mid \{\text{INT}, \text{WIN}, \text{OFF}\}.$$

This means that to understand the behavior of the users in the pages under Catalog and Entertainment, consider what the users do in the pages under these groups plus the pages under Internet, Windows and Office. Everything else does not influence their behavior in Catalog and Entertainment. This does not mean that there is no correlation between, for instance, Entertainment and Development, but it does mean that this correlation becomes irrelevant in light of what happens on the pages under Internet, Windows and Office.

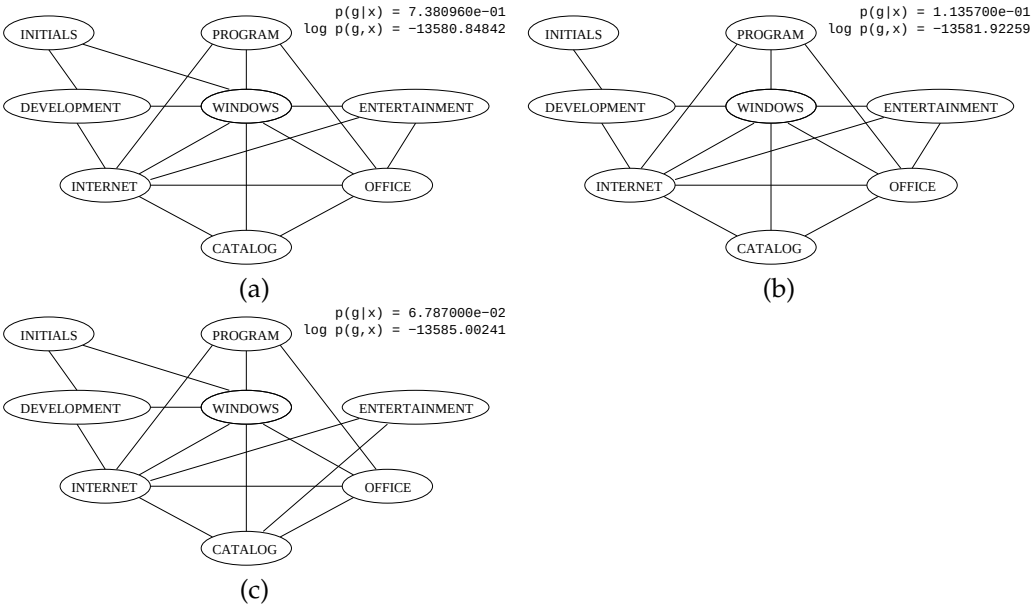


Figure 5.4: Three most probable models given the data. They differ in how hits on pages under Initials and Entertainment interact with hits on pages under Windows and Office.

To fully understand why this is so, it would be necessary to know the exact layout of all the pages of the website, since this might be the effect of the structure of the website itself. Let us assume that this has to do with user tastes and preferences. For instance, assume that we would like to increase the hits on certain pages under the group Entertainment.

Since what users do in Entertainment is related to everything else through their behavior in Windows, Internet and Office, it makes sense to concentrate efforts on increasing the hits to Entertainment by making some design decisions in the pages belonging to the three groups Windows, Internet, and Office. Of course, global major changes in the structure of the website will change the way users navigate, but we are discussing the implementation of less drastic options.

The conclusions we draw are not, in any case, the result of a causal interpretation of the associations. They are rather the result of using the notion of conditional independence to make reasonable assumptions about how our data is generated in order to try to shed some light on decisions we may need to take.

The scope of this data limits the conclusions we can draw. Nevertheless, we believe that this methodology could be more profitable if applied to more complete data where hits are linked to user profiles or marketing campaigns on the web.

The Bayesian approach implemented through the MCMC method allows to compute quantities of interest by averaging across the models. For this dataset we have compute the posterior probability of the edges which we may see in Table 5.1.

We may appreciate that the edges can be grouped in two sets, one above the threshold of 0.8, and another below. The significant pairwise associations detected in the frequen-

Table 5.1: Posterior distribution of the edges given the data.

edge	p(edge|data)
INTERNET−CATALOG	9.999600e-01
INTERNET−ENTERTAINMENT	9.999100e-01
DEVELOPMENT−INTERNET	9.998800e-01
INTERNET−OFFICE	9.998600e-01
INITIALS−DEVELOPMENT	9.998200e-01
OFFICE−PROGRAM	9.993100e-01
INTERNET−WINDOWS	9.990900e-01
INTERNET−PROGRAM	9.990800e-01
CATALOG−WINDOWS	9.990700e-01
OFFICE−WINDOWS	9.990500e-01
WINDOWS−PROGRAM	9.989600e-01
WINDOWS−DEVELOPMENT	9.951900e-01
OFFICE−CATALOG	9.616900e-01
ENTERTAINMENT−OFFICE	9.154200e-01
WINDOWS−ENTERTAINMENT	8.925100e-01
WINDOWS−INITIALS	8.634300e-01
CATALOG−DEVELOPMENT	1.079600e-01
CATALOG−PROGRAM	3.790000e-02
DEVELOPMENT−ENTERTAINMENT	4.780000e-03
OFFICE−DEVELOPMENT	3.880000e-03
ENTERTAINMENT−INITIALS	2.990000e-03
INTERNET−INITIALS	5.600000e-04
PROGRAM−DEVELOPMENT	5.200000e-04

tist analysis carried out before, basically agree with those edges in the group of higher probability.

5.3 Coronary Heart Disease

This set of data, well known in the statistical literature, was introduced by Edwards and Havránek (1985). It concerns $1,841$ cross-classified men, according to six binary coronary heart disease factors. The random variables of interest are:

A smoking

B strenuous mental work

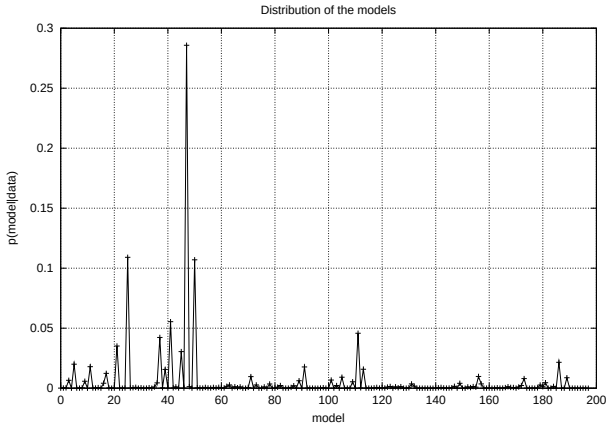
C strenuous physical work

D systolic blood pressure

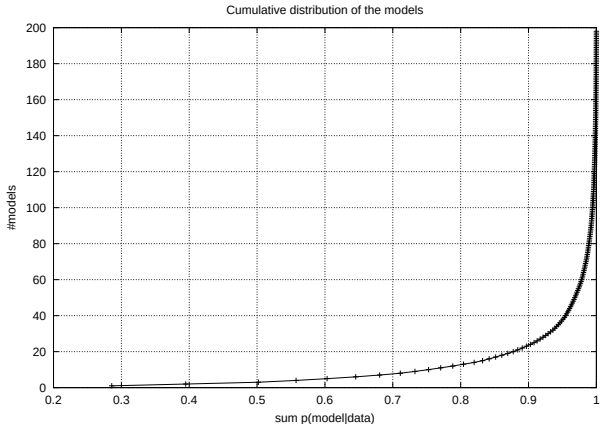
E ratio of β and α proteins

F family anamnesis of coronary heart disease

The aim of the analysis is to investigate the association structure among such risk factors. Several authors have also analyzed this dataset, as Madigan and Raftery (1994) who consider model search over the spaces of DEC and DAG Markov models. Madigan et al. (1996) extend the analysis considering model search over Markov equivalence classes of DAG models. Dellaportas and Forster (1999) consider a reversible jump MCMC algorithm over the broader class of hierarchical loglinear models.



(a)



(b)

Figure 5.5: Posterior and cumulative posterior distribution of the models.

We have analyzed this dataset with DEC Markov models, running the Markov chain for 10^5 iterations, without burn-in, and starting from the empty graph where all variables are marginally independent. This setup showed convergence according to the diagnostics which we do not show here.

In Figure 5.5a we have the posterior distribution of the models given the data. In this plot we see that three models have a posterior substantially larger than the rest. In Figure 5.5b we have the cumulative distribution from which we see that these three

models have 50% of the distribution.

Before we show the most probable models from this posterior distribution let's look first at the models selected in the earlier papers that have already analyzed this data. In Figure 5.6 we see two models selected by Edwards and Havránek (1985) and Madigan and Raftery (1994).

The classical stepwise procedure of Edwards and Havránek (1985) selected two UG models. Recall that DEC models form a subclass of UG models. Our results, and those from Madigan and Raftery (1994), reported here, use the space of DEC models. Nevertheless, it is possible to see that most of the two-way interaction terms coincide. More concretely, Madigan and Raftery (1994) select two DEC Markov models following a stepwise Bayesian procedure using an Occam's window.

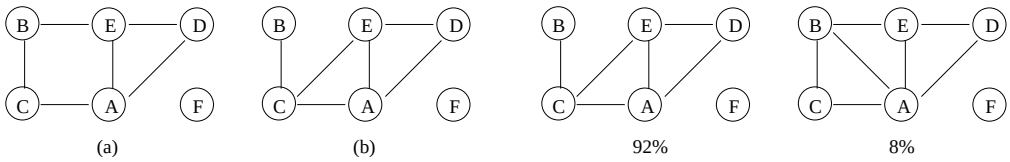


Figure 5.6: UG Markov models selected by Edwards and Havránek (1985) and DEC Markov models selected by Madigan and Raftery (1994).

We depart from these two approaches to model selection by sampling a substantial part of the probability distribution of the models given the data. This leads to the result we see on Figure 5.7 on which the four most probable models account for 57% of the distribution, while Madigan and Raftery (1994) restrict the distribution only to those models that fall in the Occam's window and their best model alone accounts for 92% of the distribution.

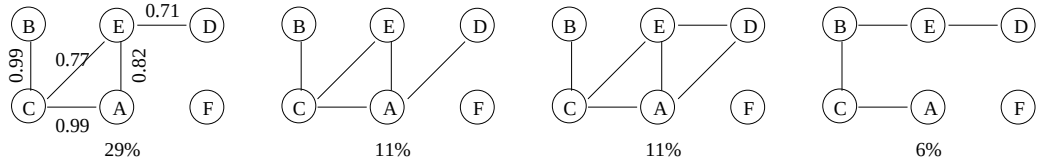


Figure 5.7: Four most probable DEC models for the CHD dataset.

The three most probable models of Figure 5.7 differ in the way the systolic blood pressure (D) interacts with the ratio of α and β proteins (E) and smoking (A). They have a similar support with a preference for the ratio of α and β proteins (E) as the factor that better predicts systolic blood pressure (D). This feature remains in the fourth model, that accounts for a 6% of the probability, but this latter model differs from the previous three in the presence of the edge B–E and the absence of the edge C–E, i.e. the way the ratio of α and β proteins (E) are directly related to strenuous mental or physical work (B or C). This disagreement coincides with the one between the two models selected by Edwards and Havránek (1985) and, as they point out, it may be caused by the high

negative association between strenuous mental and physical work, i.e. strenuous work is either physical or mental but not both at the same time. A feature common to all models selected in the earlier and our current work, is the marginal independence of the family anamnesis of coronary heart disease (F).

A feature that holds throughout the four models in Figure 5.7, is the conditional independence of smoking (A) from strenuous mental work (B) given strenuous physical work (C). This CI restriction is also supported by the posterior distributions of the CI restrictions of the form $v \perp\!\!\!\perp V \backslash cl(v) \mid bd(v)$. These posteriors are computed by setting $p(v \perp\!\!\!\perp V \backslash cl(v) \mid bd(v) | M, D) = 1$ if such CI restriction can be read in M, and 0 otherwise, in expression (3.28). We may see the posteriors that account for more than 95% of the distribution of CI restrictions in Table 5.2.

Table 5.2: Posterior distributions of conditional independencies.

A	$p(\perp\!\!\!\perp \mid \text{data})$	B	$p(\perp\!\!\!\perp \mid \text{data})$
$\perp\!\!\!\perp BDF \mid CE$	0.48	$\perp\!\!\!\perp ADEF \mid C$	0.78
$\perp\!\!\!\perp BF \mid CDE$	0.32	$\perp\!\!\!\perp ADE \mid CF$	0.11
$\perp\!\!\!\perp BDEF \mid C$	0.12	$\perp\!\!\!\perp ADF \mid CE$	0.09
$\perp\!\!\!\perp BEF \mid CD$	0.04	$\perp\!\!\!\perp AD \mid CEF$	0.01
C	$p(\perp\!\!\!\perp \mid \text{data})$	D	$p(\perp\!\!\!\perp \mid \text{data})$
$\perp\!\!\!\perp DF \mid ABE$	0.76	$\perp\!\!\!\perp ABCF \mid E$	0.54
$\perp\!\!\!\perp DEF \mid AB$	0.22	$\perp\!\!\!\perp BCEF \mid A$	0.20
		$\perp\!\!\!\perp ABCEF \mid AE$	0.16
		$\perp\!\!\!\perp ABC \mid \emptyset$	0.07
E	$p(\perp\!\!\!\perp \mid \text{data})$	F	$p(\perp\!\!\!\perp \mid \text{data})$
$\perp\!\!\!\perp BF \mid ACD$	0.49	$\perp\!\!\!\perp ABCDE \mid \emptyset$	0.76
$\perp\!\!\!\perp BDF \mid AC$	0.18	$\perp\!\!\!\perp ACDE \mid B$	0.12
$\perp\!\!\!\perp BCF \mid AD$	0.07	$\perp\!\!\!\perp ABCD \mid E$	0.05
$\perp\!\!\!\perp ACF \mid BD$	0.07	$\perp\!\!\!\perp ABCE \mid D$	0.02
$\perp\!\!\!\perp ABF \mid CD$	0.04	$\perp\!\!\!\perp BCDE \mid A$	0.02
$\perp\!\!\!\perp BCDF \mid A$	0.03	$\perp\!\!\!\perp ABDE \mid C$	0.01
$\perp\!\!\!\perp B \mid ACDF$	0.03		
$\perp\!\!\!\perp ACDF \mid B$	0.03		
$\perp\!\!\!\perp ABDF \mid C$	0.02		

We believe that $A \perp\!\!\!\perp B \mid C$ may follow from the fact that smoking may be more common for physical work than for mental work. This conditional independence statement is in one of the models of Edwards and Havránek (1985) and in the model of Madigan and Raftery (1994) that accounts for the 92% of the probability distribution.

Madigan et al. (1996) show some of these posteriors as well, computed using the posterior distribution of Markov equivalence classes of DAG models. Specifically, their method assigns probability 0.77 to $F \perp\!\!\!\perp ABCDE | \emptyset$, 0.60 to $D \perp\!\!\!\perp ABC | E$ and 0.35 to $B \perp\!\!\!\perp E | C$. From the properties of conditional independence (Pearl, 1988), it follows that $D \perp\!\!\!\perp ABCF | E \Rightarrow D \perp\!\!\!\perp ABC | E$ and that $B \perp\!\!\!\perp ADEF | C \Rightarrow B \perp\!\!\!\perp E | C$. As we see from Ta-

ble 5.2, these statements have here the probabilities 0.76, 0.54 and 0.78 respectively. Note that the first two have very similar values to those from Madigan et al. (1996), although they have been computed using quite a different method and type of model. On the other hand, this is not surprising from the fact that these posteriors are unconditional on the model.

A final observation common to all the analyses discussed here is the inability of smoking (A), as a single factor, to render strenuous physical work (C) conditionally independent from the ratio of α and β proteins (E). Table 5.2 shows that this conditional independence statement is not supported by the data. It does not fall under the 95% of the distribution for statements separating C, and among those separating E it just has a 3% of support. This may be interpreted as if, in a situation of physical work, smoking does not suffice to predict the ratio of α and β proteins. It is also necessary to know whether this physical work is carried out under strenuous conditions or not.

Some authors (e.g. Dellaportas and Forster, 1999), point out that even in this apparently small model selection exercise, one should look not only at the most probable graphs, as a considerable number of models may be quite alike in terms of support, but also at the posterior probabilities of edge presence[1].

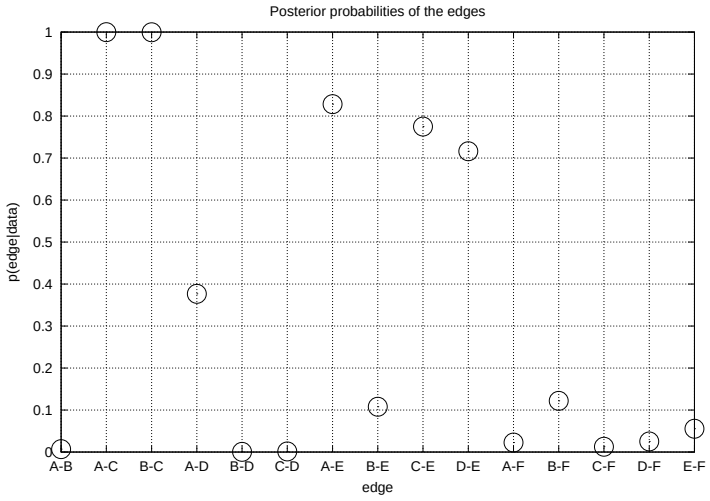


Figure 5.8: Posterior distribution of the edges given the data.

We see the posterior probabilities of the edges plotted in Figure 5.8. The highest probability of presence (0.99) is associated to the edges (B, C) and (A, C), followed by (A, E) (0.83), (C, E) (0.77), (D, E) (0.71) and, finally, by (A, D) (0.37). The remaining estimated posterior probabilities of arc presence are very low. The probabilities on the edges may help to confirm some of the conclusions one draws from the models, as for instance the low probabilities of edges where variable F is involved, support the marginal independency of variable F. Also the very low probability of the edge A–B supports the CI restriction $A \perp\!\!\!\perp B \mid C$.

[1] As we already did in the previous section with Web Mining.

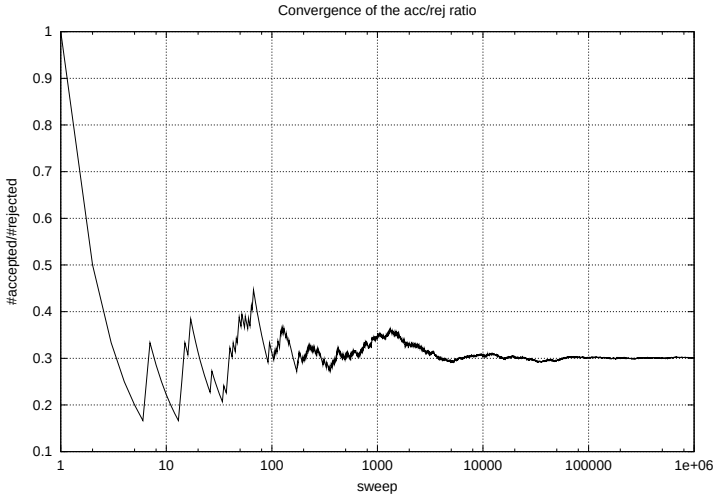


Figure 5.9: Convergence of the ratio of accepts/rejects for the MBA dataset.

A way of improving the interpretability of the posterior in Figure 5.8 is by taking those edges above a certain threshold and drawing them as a graph. This graph shows marginal relationships of a certain strength, therefore no independencies may be read off from such graph.

In the plot in Figure 5.8 we may see that there are five edges above 0.7 and the rest lie below 0.4. We have taken 0.7 as a threshold, and this set of edges coincides with the set of edges of the most probable DEC model we report in Figure 5.7. Since the graph is the same in both cases we have written in Figure 5.7 the probabilities of these five edges. Of course, one can expect that that the most probable edges appear in those DEC models that account for the largest part of the posterior distribution.

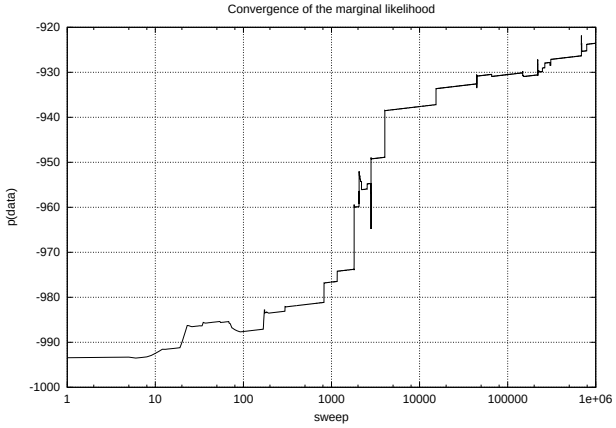
As we may see, the Bayesian analysis of this data using DEC Markov models has allowed us to discuss the relationships between the variables and make reasonable assumptions about the mechanism that generated the data.

5.4 Market Basket Analysis

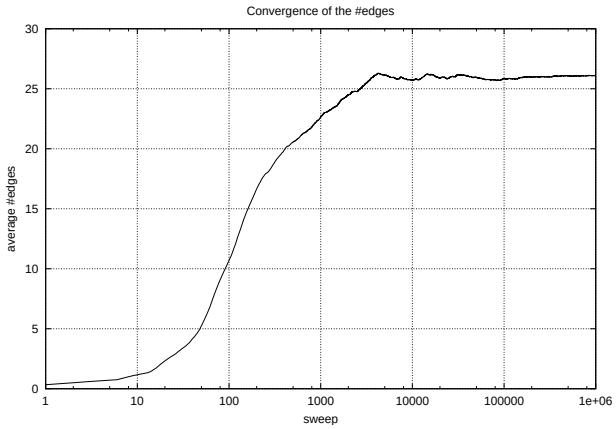
The term *market basket analysis* refers to the application of data analysis techniques to databases that store transactions from consumers buying choices of different products inside a specific unit, such as a supermarket. The aim of the analysis is to understand the association structure between the sales of the different products available. Once the associations are found, they may help planning marketing policies. For instance, if two products turn out to be heavily associated, it should be sufficient to put only one of them on discount to increase sales on both. This is the type of data typically analyzed with association rules (see chapter 4). In this section we will analyze it using DEC Markov models.

Our available data consists of the sales of a sample of 26 products (grouping different

brands for the same good) in 52 periods of one week each, for the year 1997, in one large Italian supermarket (about 12,000 square meters), in the Piedmont region. All products are food and have been chosen from those most sensitive to promotional effects. For each week, data are summarized in 26 binary variables indicating whether a certain product was sold above or below the median of the year.



(a)



(b)

Figure 5.10: Convergence of the marginal of the data (a) and of the average number of edges (b).

The classical analysis of this very sparse dataset, using an UG model over the 26 binary variables available, has led to a rather complex model, containing 44 significant edges, 9 of which describe negative associations. More details on this analysis are contained in (Giudici and Passerone, 2001) to which we refer for further details.

As in the previous sections, we have started the Markov chain from the empty graph and we have used no burn-in iterations. Because of the large sparseness of this dataset as well as the large number of competing DEC models, we have run a long Markov chain

of 10^6 iterations to achieve a good degree of convergence, according to the diagnostics we see in Figures 5.9, 5.10 and 5.11.

In Figure 5.9 we have the ratio of accepted over rejected models that converges to 0.3. In plot 5.10a we have the convergence of the marginal likelihood of the data, which does not show a degree of convergence as good as the other diagnostics but seems to stabilize for the last $200,000$ iterations. In plot 5.10b we have the convergence of the average number of edges. Finally, in Figure 5.11 we have the posterior distribution of the number of edges that takes a normal shape, as expected, centered around 26 edges.

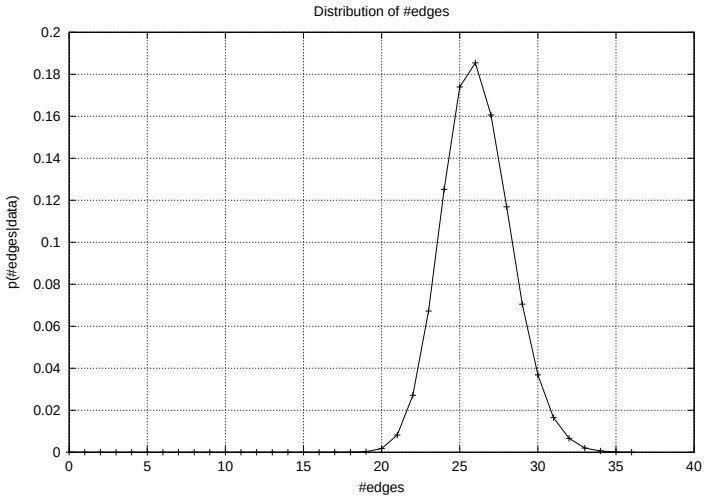


Figure 5.11: Convergence of the distribution of the numbers of edges.

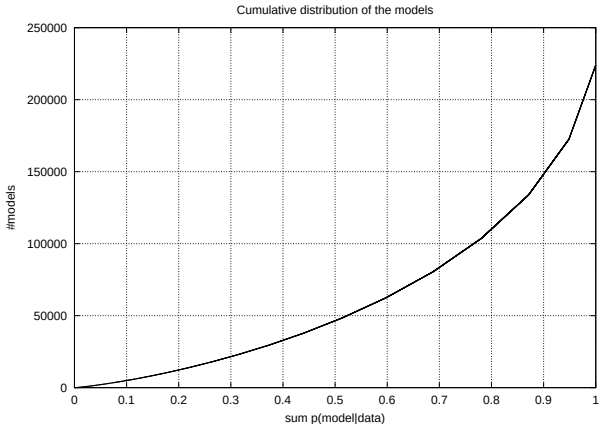
The sparseness of the data implies usually a lack of strong support towards a particular model, and we may see this effect in the cumulative posterior distribution in Figure 5.12a.

As we see, we need more than 50000 models to account for the 60% of the probability distribution. This leads to a huge number of models with similar support from the data, which makes it infeasible to discriminate between them on the basis of their posterior probabilities. However, if we take a look to the posterior of the edges given the data, in Figure 5.12b, we may conclude that sensible conclusions can be drawn from the two-way interactions.

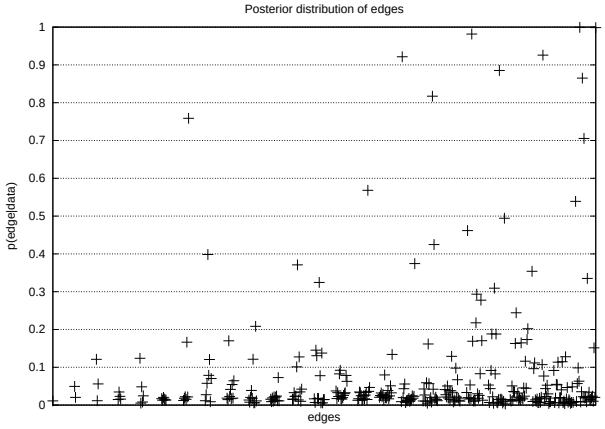
There is a limited number of edges, out of the 325 possible ones, which have a high probability of being present. For instance if we set a threshold at 0.7, only 10 edges have a higher probability. We have put them together in what we call, the representative graph, which has no interpretation in terms of conditional independence, and which appears in Figure 5.13.

The representative graph of Figure 5.13 breaks up in five disconnected components, corresponding to different consumer behaviors. All associations are positive (i.e. with an odds ratio greater than 1) with the exception of three, plotted with a dashed line and are negative.

The first cluster to the left identifies milk, cookies and rice. The association between



(a)



(b)

Figure 5.12: Cumulative distribution of models (a) and posterior distribution of the edges (b).

milk and cookies is clear, less interpretable is that with rice products. A possible reason is that these products have quite stable sales across the year. We remark that our dataset was built by simultaneously considering whether, in each week of the given year, products were sold below or above the median. This introduces a latent explanatory variable in the model, which may induce the observed associations.

The second cluster to the bottom left identifies fruit juices and beer. This association reflects a common use of these drinks in the household. Both of these products have very variable sales along the year.

The third cluster, at the right bottom of the figure, associates soft drinks, mayonnaise and frozen fish. This connected component, contains products which are quite occasionally bought, for instance to organize parties (soft drinks) or to have quick meals (this is the case of the frozen fish and the mayonnaise).

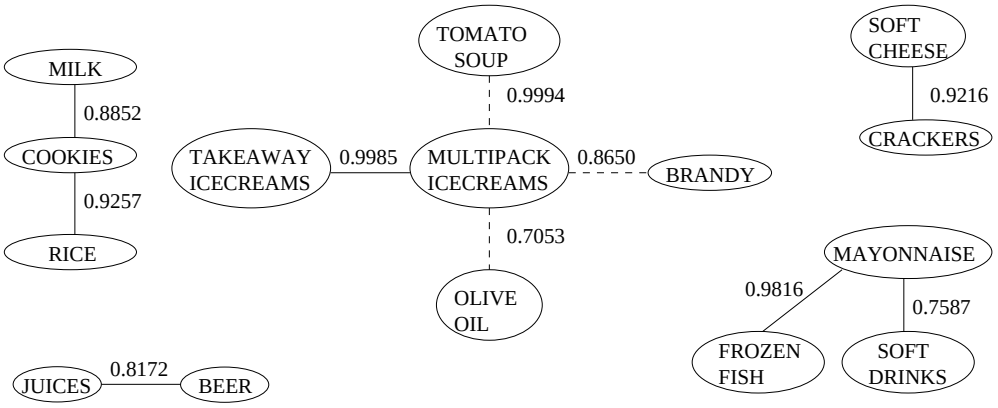


Figure 5.13: Representative graph for the MBA dataset.

The fourth cluster to the top right also identifies a category of fast-food customers, associating soft cheese and crackers.

Finally, the biggest cluster contains 5 nodes, describing seasonal sales, where four out of the five product interact only with multipack icecreams. The other nodes are take away icecreams, tomato soup, brandy and olive oil. Tomato soup, brandy and olive oil are negatively associated with multipack icecreams, reflecting different peak buy seasons.

For further information we should look at, are the posterior distributions of CI restrictions, as we did in the previous section. The 45 largest posterior probabilities are given in Table 5.3. One of the ways of using such information in the context of market basket analysis is, by focusing on products with a positive relationship. Among these products we can find out for which combinations, an increase of sales, obtained for instance by setting a price discount, may potentially increase the sales of another product.

For instance, we knew from the edge posterior probabilities that soft cheese and crackers were strongly related. If we look at the posteriors on CI restrictions, we may see that considering crackers and frozen fish together we almost double the chance that we can increase the sales of soft cheese in an indirect way.

5.5 Concluding Remarks

In this chapter we have shown three applications of the Bayesian approach to learn DEC Markov models, described in chapter 3 using the MC^3 algorithm. The possibility of approximating posterior probabilities provides us with a way to handle the uncertainty of the models discovered. In problems such as those tackled in this chapter, this becomes crucial as we do not know which mechanism generates the data with certainty.

The notion of conditional independence explains part of this mechanism. The possibility of manipulating CI restrictions using GMMs and accounting for the uncertainty of both, GMMs and consequently CI restrictions, is a powerful device for the extraction of

Table 5.3: Forty-five most probable conditional independencies for the MBA dataset. The set V refers to the whole set of products.

$U \perp\!\!\!\perp V \backslash C \mid C$	variable (U)	conditioning set (C)
4.647010e-01	BEER	JUICES
4.199370e-01	TA-ICECREAM	MP-ICECREAM
3.826470e-01	TOMATO SOUP	MP-ICECREAM
3.765420e-01	OLIVE OIL	MP-ICECREAM
3.333900e-01	SOFT-CHEESE	CRACKERS
3.302110e-01	TUNA-FISH	SODAS
3.103860e-01	SOFT-CHEESE	CRACKERS F-FISH
3.000540e-01	F-VEGETABLES	MILK
2.989510e-01	CRACKERS	SOFT-CHEESE
2.926120e-01	RICE	COOKIES
2.491750e-01	COFFEE	BRANDY
2.046430e-01	SOFT-DRINKS	MAYONNAISE
2.000360e-01	YOGURT	CRACKERS
1.912180e-01	BRANDY	MP-ICECREAM
1.839820e-01	MOZZARELLA	$\emptyset$
1.710120e-01	TIN-MEAT	MP-ICECREAM
1.672330e-01	JUICES	BEER COOKIES
1.654170e-01	MAYONNAISE	SOFT-DRINKS F-FISH
1.648010e-01	TIN-MEAT	RICE
1.612000e-01	MINERAL-WATER	$\emptyset$
1.599890e-01	RICE	TIN-MEAT COOKIES
1.534770e-01	CRACKERS	SOFT-CHEESE YOGURT
1.477690e-01	PASTA	$\emptyset$
1.439260e-01	SODAS	TUNA-FISH
1.365680e-01	MINERAL-WATER	MP-ICECREAM
1.361750e-01	MILK	COOKIES
1.327520e-01	JUICES	BEER
1.300640e-01	PARTY SAUSAGES	$\emptyset$
1.299300e-01	BRANDY	COFFEE MP-ICECREAM
1.280260e-01	PASTA	MILK
1.272230e-01	MILK	COOKIES F-VEGETABLES
1.178720e-01	YOGURT	$\emptyset$
1.152760e-01	PARTY SAUSAGES	SOFT-DRINKS
1.135050e-01	PARTY SAUSAGES	MILK
1.007980e-01	MP-ICECREAM	TOMATO-SOUP BRANDY OLIVE-OIL TA-ICECREAM
1.005020e-01	MAYONNAISE	SOFT-DRINKS SODAS F-FISH
9.947400e-02	MP-ICECREAM	TIN-MEAT TOMATO-SOUP BRANDY OLIVE-OIL TA-ICECREAM
9.736600e-02	F-FISH	SOFT-CHEESE MAYONNAISE
9.607600e-02	CRACKERS	SOFT-CHEESE F-FISH
9.380700e-02	SODAS	MAYONNAISE F-FISH
9.262100e-02	TUNA-FISH	$\emptyset$
8.995700e-02	MOZZARELLA	PASTA
8.856300e-02	COOKIES	MILK RICE
8.790000e-02	SODAS	MAYONNAISE TUNA-FISH F-FISH
8.742100e-02	BEER	JUICES F-FISH

useful knowledge from databases.

The extraction of useful knowledge from databases is a substantial part of the goals that the area of data mining sets. In the problems typically tackled in data mining there is often little subject-matter knowledge on which models are substantially important and, therefore, it is advisable to report conclusions from more than one model. Hence, a model averaging procedure, as the one that the Bayesian approach provides, is necessary.

Chapter 6

Conclusions

The analysis of high-order interactions among variables, by using GMMs, is a powerful tool to gather insight into our data and uncover potentially useful relationships that may be unknown before the analysis. However, the number of competing GMMs grows exponentially in the number of variables considered. Similar problem arises, for instance, with association rules, and a first approach is to introduce constraints such that the problem becomes tractable and the results still may be of interest.

In the case of the DAG Markov model, a first approach to make the problem tractable was to restrict the models considered by using a fixed causal order among the variables. Nevertheless, in contrast with the case of association rules, this constraint is difficult to elicitate and a wrong setting may lead to very deceptive results. Other approaches that avoid to use such constrain represent a heavy computational burden, that makes them unattractive for their use in a data mining context.

In the same vein, the implementation of the Bayesian approach to structural learning of GMMs using the MCMC method lacks of a substantial body of literature treating some kind of methodology to assess the convergence of such an iterative method. The assessment of convergence is important to enable the Bayesian learning approach to be used in a more systematic way.

We identify these circumstances as part of the reasons that make difficult the deployment of GMMs in data mining. This thesis provides important contributions to alleviate these problems and improve the current state of the art in structural learning of GMMs.

We have thoroughly described some of the different subclasses of the DAG Markov model providing an unified view of them, and identified a new subclass, the TCI Markov model. We have gathered substantial new insight into the problem of structural learning of DAG models by studying in depth the graphical Markov model inclusion order. From this study followed the introduction of two new algorithms, in the context of heuristic search and the MCMC method, that outperformed earlier work and do not compromise the computational cost of the learning process.

We have shown how the notion of conditional independence in general, and GMMs in particular, provide an appealing way to tackle the problem of learning association rules from data. The approach taken is interesting as it combines the discovery of local patterns with the use of a global modeling methodology as GMMs.

There are, of course, lots of work to be done. From a non-causal point of view, we ultimately want to use equivalence classes of DAG models. The manipulation of these

equivalence classes is still not easy nor computationally cheap. Although we have an operational criterion (Conjecture 3.1 partially proved by Kočka et al. (2001), and fully proved by Chickering (2002)) to decide the inclusion order among equivalence classes of DAG models, we still lack of a cheap graphical characterization of such order, as well as an efficient enumeration of the equivalence classes in general, and the inclusion boundary in particular.

Despite our contribution, it remains a challenge to keep scaling up the structural learning algorithms as we are confronted with massive datasets from, for instance, the biological domain, where the dimension of the datasets easily reaches an order of 10^3 variables.

Bibliography

Agrawal, R., Imilinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings ACM SIGMOD*, pages 207–216.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. (1995). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, Cambridge, MA. AAAI/MIT Press.

Andersson, S., Madigan, D., and Perlman, M. (1996). An alternative Markov property for chain graphs. In Jensen, F. and Horvitz, E., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 40–48. Morgan Kaufmann.

Andersson, S., Madigan, D., and Perlman, M. (1997a). A characterization of markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25:505–541.

Andersson, S., Madigan, D., and Perlman, M. (1997b). On the markov equivalence of chain graphs, undirected graphs, and acyclic digraphs. *Scandinavian Journal of Statistics*, 24:81–102.

Andersson, S., Madigan, D., and Perlman, M. (2001). Alternative Markov properties for chain graphs. *Scandinavian Journal of Statistics*, 28:33–85.

Andersson, S., Madigan, D., Perlman, M., and Triggs, C. (1995). On the relation between conditional independence models determined by finite distributive lattices and by directed acyclic graphs. *Journal of Statistical Planning and Inference*, 48:25–46.

Andersson, S., Madigan, D., Perlman, M., and Triggs, C. (1997c). A characterization of lattice conditional independence models. *Annals of Mathematics and Artificial Intelligence*, 21:27–50.

Andersson, S. and Perlman, M. (1993). Lattice models for conditional independence in a multivariate normal distribution. *Annals of Statistics*, 21:1318–1358.

Angelopoulos, N. and Cussens, J. (2001). Markov chain monte carlo using tree-based priors on model structure. In Breese, J. and Koller, D., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 16–23. Morgan Kaufmann.

Asmussen, S. and Edwards, D. (1983). Collapsability and response variables in contingency tables. *Biometrika*, 70(3):567–578.

Beeri, C., Fagin, R., Maier, D., Mendelzon, A., Ullman, J., and Yannakakis, M. (1981). Properties of acyclic database schemes. In *Proc. of the Thirteenth Annual ACM Symposium on Theory of Computation*, pages 355–362.

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

Boncz, P. and Kersten, M. (1995). Monet: An Impressionist Sketch of an Advanced Database System. In *Proceedings Basque International Workshop on Information Technology*, San Sebastian, Spain.

Bouckaert, R. (1992). Optimizing causal orderings for generating dags from data. In Dubois, D., Wellman, M., D'Ambrosio, B., and Smets, P., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 9–16. Morgan Kaufmann.

Bouckaert, R. (1995). *Bayesian Belief Networks: from Construction to Inference*. PhD thesis, University of Utrecht.

Breese, J., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Brooks, S. (1998). Markov chain monte carlo method and its application. *The Statistician*, 47:69–100.

Buntine, W. (1991). Theory refinement on Bayesian networks. In B. D'Ambrosio, P. S. and Bonissone, P., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann.

Buntine, W. (1996a). Graphical models for discovering knowledge. In Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 59–82. AAAI Press.

Buntine, W. (1996b). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210.

Carlin, B. and Chib, S. (1995). Bayesian model choice via Markov chain monte carlo methods. *Journal of the Royal Statistical Society B*, 57(3):473–484.

Castelo, R., Feelders, A., and Siebes, A. (2001). Mambo: Discovering association rules based on conditional independencies. In Hoffmann, F., Hand, D., Adams, N., Fisher, D., and Guimarães, G., editors, *Proceedings 4th Symposium on Intelligent Data Analysis*, volume 2189 of *Lecture Notes in Computer Science*, pages 289–298. Springer.

Castelo, R. and Kočka, T. (2002). Towards an inclusion driven learning of Bayesian networks. Technical Report UU-CS-2002-05, Institute for Computing and Information Sciences, University of Utrecht, The Netherlands. *Submitted to the Journal of Machine Learning Research*.

Castelo, R. and Siebes, A. (2000). Priors on network structures. biasing the search for Bayesian networks. *International Journal of Approximate Reasoning*, 24(1):39–57.

Castelo, R. and Siebes, A. (2001). A characterization of moral transitive acyclic directed graph Markov models as labeled trees. *Journal of Statistical Planning and Inference, to appear*.

Castelo, R. and Wormald, N. (2001). Enumeration of P_4-free chordal graphs. Technical Report UU-CS-2001-12, Institute for Computing and Information Sciences, University of Utrecht, The Netherlands. *Submitted to the Journal of Graphs and Combinatorics*.

Castillo, E., Ferrandiz, J., and Sanmartin, P. (1998). Marginalizing in undirected graph and hypergraph models. In Cooper, G. and Moral, S., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 69–78. Morgan Kaufmann.

Castillo, E., Hadi, A., and Solares, C. (1997). Learning and updating of uncertainty in dirichlet models. *Machine Learning*, 26:43–63.

Chib, S. and Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *American Statistician*, 49(4):327–335.

Chib, S. and Jeliazkov, I. (2001). Marginal likelihood from the metropolis-hastings output. *Journal of the American Statistical Association*, 96(453):270–281.

Chickering, D. (1995). A transformational characterization of equivalent Bayesian networks. In Besnard, P. and Hanks, S., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 87–98. Morgan Kaufmann.

Chickering, D. (1996a). Learning Bayesian networks is NP-complete. In Fisher, D. and Lenz, H.-J., editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag.

Chickering, D. (1996b). Learning equivalence classes of Bayesian network structures. In Horvitz, E. and Jensen, F., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 150–157. Morgan Kaufmann.

Chickering, D. (2001). Learning equivalence classes of Bayesian-network structures. Technical report, Microsoft Research.

Chickering, D. (2002). *Personal Communication*.

Chickering, D., Geiger, D., and Heckerman, D. (1995). Learning Bayesian networks: Search methods and experimental results. In *Proc. of the Int'l. Workshop on Artificial Intelligence and Statistics*, pages 112–128.

Chung, K. (1967). *Markov Chains with Stationary Transition Probabilities (2nd ed)*. Springer-Verlag.

Cohen, E., Datar, M., Funjiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J., and Yang, C. (2001). Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):64–78.

Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–405.

Cowell, R., Dawid, A., Lauritzen, S., and Spiegelhalter, D. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

Cox, D. and Wermuth, N. (1996). *Multivariate Dependencies – Models, Analysis and Interpretations*. Chapman & Hall, London.

D. Geiger, A. P. and Pearl, J. (1993). Learning simple causal structures. *International Journal of Intelligent Systems*, 8:231–247.

Davey, B. and Priestley, H. (1990). *Introduction to Lattices and Order*. Cambridge University Press, Cambridge.

Dawid, A. (1979). Conditional independence in statistical theory (with discussion). *Journal of the Royal Statistical Society B*, 41(1):1–31.

Dawid, A. and Lauritzen, S. (1993). Hyper-Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21(3):1272–1317.

de Campos, L. and Huete, J. (1992). Efficient algorithms for learning simple belief networks. Technical report, DECSAI, Universidad de Granada, Departamento de Ciencias de la Computacion e Inteligencia Artificial.

de Campos, L. and Huete, J. (1997). Algorithms for learning decomposable models and chordal graphs. In Geiger, D. and Shenoy, P., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 46–53. Morgan Kaufmann.

DeGroot, M. H. (1970). *Optimal Statistical Decisions*. McGraw-Hill.

Dellaportas, P. and Forster, J. (1999). Markov chain monte carlo model determination for hierarchical and graphical log-linear models. *Biometrika*, 86(3):615–633.

Deshpande, A., Garofalakis, M., and Jordan, M. (2001). Efficient stepwise selection in decomposable models. In Breese, J. and Koller, D., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Draper, D. (1995). Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society B*, 57:45–97.

Edwards, D. and Havránek, T. (1985). A fast procedure for model search in multidimensional contingency tables. *Biometrika*, 72(2):339–351.

Etxeberria, R., Larrañaga, P., and Pikaza, J. (1997). Analysis of the behaviour of the genetic algorithms when searching Bayesian networks from data. *Pattern Recognition Letters*, 18(11–13):1269–1273.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery: An overview. In Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press.

Friedman, J. (1997). Data mining and statistics: What's the connection? In *Proc. of the 29th. Symposium on the Interface: Computing Science and Statistics*, Houston, Texas.

Friedman, N. and Koller, D. (2000). Being Bayesian about network structure. In Boutilier, C. and Goldszmidt, M., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*. Morgan Kaufmann.

Frydenberg, M. (1990a). The chain graph Markov property. *Scandinavian Journal of Statistics*, 17:333–353.

Frydenberg, M. (1990b). Marginalization and collapsability in graphical interaction models. *Annals of Statistics*, 18(2):790–805.

Frydenberg, M. and Lauritzen, S. (1989). Decomposition of maximum likelihood in mixed interaction models. *Biometrika*, 76(3):539–555.

Geiger, D. and Heckerman, D. (1995). A characterization of the dirichlet distribution with application to learning Bayesian networks. In Besnard, P. and Hanks, S., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 999–999. Morgan Kaufmann.

Gillispie, S. and Perlman, M. (2001). Enumerating Markov equivalence classes of acyclic digraph models. In Breese, J. and Koller, D., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 171–177. Morgan Kaufmann.

Giudici, P. and Castelo, R. (2001a). Association models for Web Mining. *Journal of Data Mining and Knowledge Discovery*, 24(1):39–57.

Giudici, P. and Castelo, R. (2001b). Improving Markov chain monte carlo model search for data mining. *Machine Learning*, 50(1/2).

Giudici, P. and Green, P. (1999). Decomposable graphical gaussian model determination. *Biometrika*, 86(4):785–801.

Giudici, P. and Passerone, G. (2001). Data mining of association structures to model consumer behaviour. *Journal of Computational Statistics and Data Analysis, to appear*.

Glymour, C. (1995). Available technology for discovering causal models, building bayes nets, and selecting predictors: The tetrad ii program. In Fayyad, U. and Uthurusamy, R., editors, *Proc. of the Int'l. Conf. on Knowledge Discovery and Data Mining*, pages 130–135, Montreal, Quebec.

Golumbic, M. (1978). Trivially perfect graphs. *Discrete Mathematics*, 24:105–107.

Goodman, R., Smyth, P., Higgins, C., and Miller, J. (1992). Rule-based neural networks for classification and probabity estimation. *Neural Computation*, 4(6):781–804.

Grätzer, G. (1978). *General Lattice Theory*. Birkhäuser Verlag, Basel.

Green, P. (1995). Reversible jump Markov chain monte carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732.

Han, J. and Kamber, M. (2001). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco.

Hand, D., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining*. MIT Press, Cambridge, MA.

Harary, F., Kabell, J., and McMorris, F. (1992). Subtree acyclic digraphs. *Ars Combinatoria*, 34:93–95.

Harary, F. and Palmer, E. (1973). *Graphical Enumeration*. Academic Press, New York.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer.

Hastings, W. (1970). Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Havránek, T. (1984). A procedure for model search in multidimensional contingency tables. *Biometrics*, 40:95–100.

Hearne, T. and Wagner, C. (1973). Minimal covers of finite sets. *Discrete Mathematics*, 5:247–251.

Heckerman, D. (1996). Bayesian networks for discovering knowledge. In Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 273–305. AAAI Press.

Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R., and Kadie, C. (2000). Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, 1:49–75.

Heckerman, D., Geiger, D., and Chickering, D. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:194–243.

Herskovits, E. (1991). *Computer-Based Probabilistic Network Construction*. PhD thesis, Medical Information Sciences, Stanford University.

Jensen, F. and Jensen, F. (1994). Optimal junction trees. In de Mantaras, R. L. and Poole, D., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 360–366. Morgan Kaufmann.

Kass, R. and Raftery, A. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795.

Kiiveri, H., Speed, T., and Carlin, J. (1984). Recursive causal models. *J. Austral. Math. Soc. Ser. A*, 36:30–52.

Kočka, T. (2001). *Graphical Models: learning and applications*. PhD thesis, Faculty of Informatics and Statistics, University of Prague.

Kočka, T., Bouckaert, R., and Studený, M. (2001). On characterizing inclusion of Bayesian networks. In Breese, J. and Koller, D., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 261–268. Morgan Kaufmann.

Kočka, T. and Castelo, R. (2001). Improved learning of Bayesian networks. In Breese, J. and Koller, D., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 269–276. Morgan Kaufmann.

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

Larrañaga, P., Kuijpers, C., Murga, R., and Yurramendi, Y. (1996a). Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics*, 26(4):487–493.

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R., and Kuijpers, C. (1996b). Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926.

Lauritzen, S. (1996). *Graphical Models*. Oxford University Press, Oxford.

Lauritzen, S., Dawid, A., Larsen, B., and Leimer, H. (1990). Independence properties of directed Markov fields. *Networks*, 20:491–505.

Lauritzen, S. and Wermuth, N. (1989). Graphical models for association between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57.

Leimer, H. (1989). Triangulated graphs with marked vertices. *Annals of Discrete Mathematics*, 41:311–324.

Madigan, D., Andersson, S., Perlman, M., and Volinsky, C. (1996). Bayesian model averaging and model selection for markov equivalence classes of acyclic digraphs. *Communications in Statistics (theory and methods)*, 25(11):2493–2512.

Madigan, D. and Raftery, A. (1994). Model selection and accounting for model uncertainty in graphical models using occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546.

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

Meek, C. (1997). *Graphical models, selecting causal and statistical models*. PhD thesis, Carnegie Mellon University.

Melançon, G., Dutour, I., and Bousquet-Melou, M. (2000). Random generation of dags for graphs drawing. Technical report, Centrum voor Wiskunde en Informatica.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, California.

Pearl, J. and Paz, A. (1987). Graphoids: A graph-based logic for reasning about relevancy relations. In Boulay, B. D., editor, *Advances in Artificial Intelligence-II*. North-Holland.

Pearl, J. and Verma, T. (1987). The logic of representing dependencies by directed graphs. In *Proc. of the Conf. of the American Association of Artificial Intelligence*, pages 374–379.

Sangüesa, R. and Cortés, U. (1997). Learning causal networks from data: a survey and a new algorithm for recovering possibilistic causal networks. *AI Communications*, 10:31–61.

Silverstein, C., Brin, S., and Motwani, R. (1998). Beyond market baskets: Generalizing associations rules to dependence rules. In *Data Mining and Knowledge Discovery*.

Singh, M. and Valtorta, M. (1993). An algorithm for the construction of Bayesian network structures from data. In Heckerman, D. and Mamdani, A., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 259–265. Morgan Kaufmann.

Smith, A. and Roberts, G. (1993). Bayesian computation via the gibbs sampler and related Markov chain monte carlo methods. *Journal of the Royal Statistical Society B*, 55(1):3–23.

Smyth, P. (2001). Data mining at the interface of computer science and statistics (invited chapter). In *"Data Mining for Scientific and Engineering Applications"*.

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

Spiegelhalter, D., Thomas, A., and Best, N. (1996). Computation on Bayesian graphical models. In Bernardo, J., Berger, J., Dawid, A., and Smith, A., editors, *Bayesian Statistics 5*, pages 407–425, Oxford, UK. Clarendon Press.

Spirtes, P., Glymour, C., and Scheimes, R. (1993). *Causation, Prediction and Search*. Springer-Verlag, New York.

Spirtes, P. and Meek, C. (1995). Learning Bayesian networks with discrete variables from data. In Fayyad, U. and Uthurusamy, R., editors, *Proc. of the Int'l. Conf. on Knowledge Discovery and Data Mining*, pages 294–299, Montreal, Quebec. AAAI Press.

Studeny, M. (1997). On marginalization, collapsability and precollapsability. In Benes, V. and Stepan, J., editors, *Distributions with Given Marginals and Moment Problems*, pages 191–198, Dordrecht. Kluwer.

Tarjan, R. and Yannakakis, M. (1984). Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing*, 13:566–579.

Verma, T. and Pearl, J. (1988). Influence diagrams and d-separation. Technical report, Cognitive Systems Laboratory, UCLA.

Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In Bonissone, P., Henrion, M., Kanal, L., and Lemmer, J., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 255–268. Morgan Kaufmann.

Wermuth, N. (1980). Linear recursive equations, covariance selection, and path analysis. *Journal of the American Statistical Association*, 75:963–972.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. Wiley, New York.

Wolk, E. (1962). The comparability graph of a tree. *Proc. Am. Math. Soc.*, 13:789–795.

Wolk, E. (1965). A note on "the comparability graph of a tree". *Proc. Am. Math. Soc.*, 16:17–20.

Wormald, N. (1985). Counting labeled chordal graphs. *Graphs and Combinatorics*, 1:193–200.

Xiang, Y., Wong, S., and N.Cercone (1996). Critical remarks on single link search in learning belief networks. In Horvitz, E. and Jensen, F., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 564–571. Morgan Kaufmann.

List of Publications

Castelo, R., Feelders, A., and Siebes, A. (2001). Mambo: Discovering association rules based on conditional independencies. In Hoffmann, F., Hand, D., Adams, N., Fisher, D., and Guimarães, G., editors, *Proceedings 4th Symposium on Intelligent Data Analysis*, volume 2189 of *Lecture Notes in Computer Science*, pages 289–298. Springer.

Castelo, R. and Kočka, T. (2002). Towards an inclusion driven learning of Bayesian networks. Technical Report UU-CS-2002-05, Institute for Computing and Information Sciences, University of Utrecht, The Netherlands. *Submitted to the Journal of Machine Learning Research*.

Castelo, R. and Siebes, A. (2001). A characterization of moral transitive acyclic directed graph markov models as labeled trees. *Journal of Statistical Planning and Inference, to appear*.

Castelo, R. and Wormald, N. (2001). Enumeration of P_4-free chordal graphs. Technical Report UU-CS-2001-12, Institute for Computing and Information Sciences, University of Utrecht, The Netherlands. *Submitted to the Journal of Graphs and Combinatorics*.

Giudici, P. and Castelo, R. (2001). Association models for Web Mining. *Journal of Data Mining and Knowledge Discovery*, 24(1):39–57.

Giudici, P. and Castelo, R. (2003). Improving markov chain monte carlo model search for data mining. *Machine Learning*, 50:1/2.

Kočka, T. and Castelo, R. (2001). Improved learning of bayesian networks. In Breese, J. and Koller, D., editors, *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, pages 269–276. Morgan Kaufmann.

Curriculum Vitae

Robert Castelo, born in 1972 in Lleida, Spain, graduated on 1994 in Computer Engineering (Bachelor's level) by the University of Lleida and on 1997 in Computer Engineering (Master's level) by the Technical University of Catalonia, in Barcelona. He spent his last year as undergraduate doing his Master's thesis at CWI, the Dutch national research center for mathematics and computer science, in Amsterdam. From 1997 till 2000 he remained at CWI as PhD student under supervision of Prof. Arno Siebes. On September 2000, jointly with Prof. Arno Siebes, he moved to the University of Utrecht where, till December 2001, he followed his work to complete this PhD thesis.

He is currently working as a postdoc at the Research Group on Biomedical Informatics from the Pompeu Fabra University, in Barcelona.

Index

Titles in the SIKS Dissertation Series

98-1 Johan van den Akker,*DEGAS - An Active, Temporal Database of Autonomous Objects*.
98-2 Floris Wiesman, *Information Retrieval by Graphically Browsing Meta-Information*.
98-3 Ans Steuten, *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*.
98-4 Dennis Breuker, *Memory versus Search in Games*.
98-5 Eduard Oskamp, *Computerondersteuning bij Straftoemeting*.
99-1 Mark Sloof, *Physiology of Quality Change Modelling: Automated modelling of Quality Change of Agricultural Products*.
99-2 Rob Potharst, *Classification using decision trees and neural nets*.
99-3 Don Beal, *The Nature of Minimax Search*.
99-4 Jacques Penders, *The practical Art of Moving Physical Objects*.
99-5 Aldo de Moor, *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*.
99-6 Niek Wijngaards, *Re-design of compositional systems*.
99-7 David Spelt, *Verification support for object database design*.
99-8 Jacques Lenting, *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation*.
2000-1 Frank Niessink, *Perspectives on Improving Software Maintenance*.
2000-2 Koen Holtman, *Prototyping of CMS Storage Management*.
2000-3 Carolien Metselaar, *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief*.
2000-4 Geert de Haan, *ETAG, A Formal Model of Competence Knowledge for User Interface Design*.
2000-5 Ruud van der Pol, *Knowledge-based Query Formulation in Information Retrieval*.
2000-6 Rogier van Eijk, *Programming Languages for Agent Communication*.
2000-7 Niels Peek, *Decision-theoretic Planning of Clinical Patient Management*.
2000-8 Veerle Coupé, *Sensitivity Analysis of Decision-Theoretic Networks*.
2000-9 Florian Waas, *Principles of Probabilistic Query Optimization*.
2000-10 Niels Nes, *Image Database Management System Design Considerations, Algorithms and Architechture*.
2000-11 Jonas Karlsson, *Scalable Distributed Data Structures for Database Management*.
2001-1 Silja Renooij, *Qualitative Approaches to Quantifying Probabilistic Networks*.
2001-2 Koen Hindriks, *Agent Programming Languages: Programming with Mental Models*.
2001-3 Maarten van Someren, *Learning as problem solving*.
2001-4 Evgueni Smirnov, *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*.
2001-5 Jacco van Ossenbruggen, *Processing Structured Hypermedia: A Matter of Style*.
2001-6 Martijn van Welie, *Task-based User Interface Design*.
2001-7 Bastiaan Schonhage, *Diva: Architectural Perspectives on Information Visualization*.
2001-8 Pascal van Eck, *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*.
2001-9 Pieter Jan 't Hoen, *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*.
2001-10 Maarten Sierhuis, *Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design*.
2001-11 Tom M. van Engers, *Knowledge Management: The Role of Mental Models in Business Systems Design*.
2002-01 Nico Lassing, *Architecture-Level Modifiability Analysis*.
2002-02 Roelof van Zwol, *Modelling and searching web-based document collections*.
2002-03 Henk Ernst Blok, *Database Optimization Aspects for Information Retrieval*.
2002-04 Robert Castelo, *The Discrete Acyclic Digraph Markov Model in Data Mining*.