An Environment for Graphical Models

Part I

Efficient Algorithms for Contingency Tables

Jens Henrik Badsberg

AALBORG UNIVERSITY

Institute of Electronic Systems
Department of Mathematics and Computer Science

An Environment for Graphical Models

Part I:

Efficient Algorithms for Contingency Tables

Jens Henrik Badsberg

March 1995

A thesis submitted to the Faculty of Technology and Science at Aalborg University for the degree of Doctor of Philosophy.

Institute for Electronic Systems
Department of Mathematics and Computer Science
Fredrik Bajers Vej 7 – DK 9220 Aalborg st – Denmark
Tel.: +45 98 15 85 22 – TELEX 69 790 aub dk

This book has been produced with LaTeX by the author and has been typeset in Times Roman typeface. The book has been prepared on Sun Sparcstations running SunOS (UNIX) and OpenWindows and it has been printed in PostScript on a Hewlett-Packard LaserJet 4m.

P r e f a c e

This thesis concerns the development of eŒcient algorithms for estimation in contingency tables, implementation of these algorithms in a computer program together with procedures for model selection, and inclusion of the program in a modern system for statistical analysis. The class of models is limited to models for discrete variables.

The thesis can be divided into three parts:

- EŒcient algorithms the analysis of contingency tables,
- The program CoCo,
- Inclusion of CoCo into the system XLISP-STAT.

The study has been partly theoretical and partly practical. CoCo and the integration of CoCo in XLISP-STAT is the result of the practical part. This software is available by anonymous ftp over the internet from ftp.iesd.auc.dk.

This book is volume 1 of the thesis, and contains besides the danish summary and the english summary and overview, chapter 1, four chapters on eŒcient algorithms for the analysis of contingency tables. (The paper Badsberg (1992) of the second part of the thesis is placed in this ørst volume as chapter øve.) ıA Guide to CoCoȷ and ıXlisp+CoCo — A Tool for Graphical Modelsȷ is respectively volume 2 and 3 of the thesis.

The items of the individual parts are self-contained, and thus the same text may appear in more than one item. The items of each of the parts have not been published, except the paper on the model selection strategies implemented in CoCo, which has appeared in a conference proceeding volume.

I wish to thank Steœen L. Lauritzen for inspiring the creation of CoCo, for encouraging me to apply for the scholarship, and for many stimulating discussions and valuable comments during the work.

I also wish to express my gratitude to the GRAM group, David Edwards, Poul Svante Eriksen, Morten Frydenberg, Svend Kreiner and Steœen L. Lauritzen, for providing a very stimulating environment. Also the ODIN group at Aalborg University has provided a stimulating environment.

From January to April 1990, I was a visiting student at the Department for Statistics and Actuarial Science, Waterloo University, Canada. I wish to thank all the people there for a very stimulating and memorable experience, and for making me feel very welcome. Special thanks to R. Wayne Oldford for stimulating discussions, and for aiming my focus on Lisp as an environment for statistical computing.

Many thanks goes to the users of CoCo, especially those who reported errors of CoCo.

Finally, I am indebted to my sister Annette Badsberg for helping me with the English language and for correcting the English of many parts of the thesis.

The study has been funded in part by the Danish Research Consils through the PIFT programme.

Aalborg, Danmark, March 15th 1995,

Jens Henrik Badsberg

I n d h o l d

Summary in Danish,

Dansk resume

Programmeringsomgivelser for graøske associationsmodeller

Graøske modeller for statistisk analyse af data har opnet betydelig opmrksomhed gennem det sidste rti. De graøske modeller giver en umiddelbar let forstelse af modellerne og muliggr hndtering af mere komplekse modeller. Endvidere er der ved graf-teoretiske resultater omkring disse modeller udviklet eœektive algoritmer for estimation og model-selektion under klassen af hierarkiske log-linere modeller.

Graøske modeller anvendes ikke alene indenfor det statistiske omrde, men bruges ogs af bl.a. dataloger i forbindelse med ekspertsystemer og kunstig intelligens.

Hvad er en graøsk model?

Dette projekt vedrrer analyse af kontingenstabeller: Betragt tabellen, der fremkommer ved at man tager enhver konøguration af en mngde af diskrete variable, dvs. kvalitative variable. Betragt f.eks. tabellen med de 8 celler, der er ndvendige, nr man vil klassiøcere en person efter hvorvidt personen ryger, personens far ryger og personens mor ryger. Kontingenstabellen er da tabellen, der fremkommer som antallet af observationer i enhver celle af tabellen, nr et st af personer klassiøceres med hensyn til tabellen.

Graøske modeller er de log-linere modeller for kontingenstabeller, der kan reprsenteres ved en simpel ikke-orienteret graf med s mange knuder som tabellen har dimension. Disse modeller kan fortolkes ved betingede uafhngigheder, og denne fortolkning kan lses direkte af grafen.

Graøske modeller er en delklasse af hierarkiske modeller, men indeholder klassen af dekomposable modeller. Disse graøske modeller vedrrer symmetriske associationer mellem diskrete variable, men ogs asymmetriske associationer kan hndteres ved orienterede grafer for rekursive modeller.

Andre modeller

Analogt til graøske modeller for diskrete variable, dvs. kontingenstabeller, kan graøske modeller for kontinuerte variable deøneres. Disse to klasser af modeller er delklasser af klassen af de graøske interaktions modeller, introduceret i Lauritzen & Wermuth (1984).

Afhandlingens dele

Denne afhandling vedrrer udvikling af eœektive algoritmer for estimation og modelselektion i kontingenstabeller, implementation af disse algoritmer i et afrundet program og integration af dette program i et moderne milj for statistisk analyse.

Afhandlingen kan derfor naturligt opsplittes i tre dele, hvoraf den frste kan betragtes som teoretisk og de to sidst som praktiske:

- Eœektive algoritmer for analyse af kontingenstabeller,
- Programmet CoCo,
- Integration af CoCo i systemet XLISP-STAT.

Eœektive algoritmer

Den frste del af afhandlingen bestr ud over den engelske sammenfatning kapitel 1 af tre kapitler: Kapitel 2 handler om hvorfor dekomposition af de log-linere modeller er vigtig for at opn eœektive algoritmer for estimation i disse modeller, samt om en plads-besparende implementation af den ndvendige iterative algoritme for ikke-dekomposable modeller. Kapitel 3 i denne del er om eœektive algoritmer for lsning af de graf-teoretiske problemer omkring dekomposition af log-linere modeller. Endelig indeholder denne frste del af afhandlingen kapitel 4, der giver algoritmer for bestemmelse af den mindste mngde, hvorp en log-liner model er kollapsibel, og som indeholder en given mngde.

CoCo

Anden del af afhandlingen bestr af programmet CoCo, et program for estimation, test og model-selektion i log-linere modeller p store kontingens-tabeller. Ud over de beskrevne eœektive algoritmer for estimation i kontingenstabeller er der i dette program implementeret algoritmer for model-selektion. Ved stepvis forward selektion eller backward elimination af kanter eller interaktioner kan en minimal acceptabel model identiøceres, eller hele rummet af modeller kan ved den skaldte EH-procedure deles i en klasse af acceptable og en klasse af kassable modeller. Derudover kan der i CoCo beregnes eksakte testsandsynligheder ved Monte-Carlo simulation for et test mellem en dekomposabel model og en vilkrlig dekomposable delmodel heraf, der kan gives initiale vrdier til IPS-algoritmen og tabeller med strukturelle nuller kan sledes hndteres, et justeret antal frihedsgrader kan beregnes og endelig er en af de betydelige egenskaber ved CoCo, at ufuldstndige observationer kan hndteres ved EM-algoritmen.

Del II er en guide til dette program. Denne del indeholder ogs artiklen Badsberg (1992) om strategier for modelselektion implementeret i CoCo.

Lisp-CoCo

I den tredje del af afhandlingen beskrives integrationen af CoCo i Lisp systemet XLISP-STAT. Denne integration muliggr en graøsk brugerÆade til CoCo og giver mulighed for at skrive Lisp-funktioner, der regner videre p estimater og andre resultater fra CoCo, hndtering af kausale modeller, etc. Dette system Lisp-CoCo er dokumenteret i del III af afhandlingen.

Den graøske brugerÆade til CoCo bestr af associations-diagrammerne: Associations-diagrammer er et meget tiltalende vrktj ved model-selektion. Associations-diagrammet kan editeres ved at knuder, punkter for variable i kontingens-tabellen, trkkes med musen, og kanterne til de Æyttede knuder vil da flge knuderne. Nye diagrammer kan skabes fra et diagram ved at kanter tilfjes eller fjernes med musen. To variable kan testes betingede uafhngige ved at kanten mellem de to knuder for variablene klikes med musen. Kanterne kan tegnes med en tykkelse, der er proportional med hvor signiøkante kanterne er. Kanter kan ogs forsynes med en vrdi beregnet fra testet for fjernelse af kanten, f.eks. med p-vrdien eller en IC-vrdi. Kanterne kan tegnes med en farve, der afhngiger af om kanten er besgt eller ikke. Hvis der under en model selektion p baggrund af et associations-diagram ikke beregnes test for en kant, dvs. hvis kanten ikke besges, tegnes kanten med en farve afhngig af hvorfor kanten ikke besges. Et besg af en kant kan undlades, fordi kanten er tvunget ind i modellen, fordi fjernelse af kanten er forkastet ved koherens, eller fordi fjernelse af kanten resultere i en ikke-dekomposable model, etc.

Kausale modeller kan hndteres i associations-diagrammerne, og metoder for backward elimination og forward selektion af kanter i associations-diagrammer er ogs implementeret i Lisp-kode.

Ud over associations-diagrammer for kausale modeller introducerer dette vrktj ogs ımodel dynamiske rotations plotsȷ og en ımodel managerenȷ. Et ımodel dynamiske rotations plotȷ er et dynamisk 3-dimensionalt plot, hvor vrdierne i plottet opdateres ved ndring af modellerne, hvorunder vrdierne i plottet beregnes. ıModel managerenȷ er en graf, der holder styr p alle aktive associations-diagrammer i Xlisp+CoCo. Test mellem to modeller kan i plottet for model manager udfres ved at en kant trkkes fra punktet for den ene model til punktet for den anden model i model manageren.

Del I

Efficient Algorithms for Contingency Tables

Kapitel 1

An Environment for Graphical Models

1.1 Introduction

Graphical models for statistical data have obtained considerable attention during the last decade. The use of graphs gives a better understanding of the models and enables handling more complex models. Besides, eŒcient algorithms for estimation and model selection have been developed by the use of graph-theoretical results.

Graphical models are not only used within the statistical community, but also among computer scientists for expert systems and artiøcial intelligence (Lauritzen & Spiegelhalter 1988, Spiegelhalter, Dawid, Lauritzen & Cowell 1993, Jensen, Lauritzen & Olesen 1990).

Graphical Models on Contingency Tables

Graphical models are for contingency tables log-linear interaction models that can be represented by a simple undirected graph with as many vertices as the table has dimension. Further, all these models can be given an interpretation in terms of conditional independence and the interpretation can be read directly oœ the graph in the form of a Markov property. In Darroch, Lauritzen & Speed (1980) graphical models are deøned by the close connection between the theory of Markov øelds and that of log-linear interaction models for contingency tables. The class of graphical model is a proper subclass of the hierarchical models, but the class strictly contains the decomposable models, e.g., Haberman (1974) and Lauritzen (1982).

These graphical models concern symmetric associations among discrete variables, but also asymmetric associations can be dealt with by directed graphs for recursive models (Wermuth & Lauritzen 1983).

Other Models

Similar to graphical models for discrete variables, i.e., contingency tables, graphical models can be deøned for continuous variables (Wermuth 1976a, Wermuth 1976b).

These two classes of models are subclasses of the graphical association models, introduced in Lauritzen & Wermuth (1984). Theoretical aspects of this general class of models for both discrete and continuous variables are further investigated in Lauritzen (1989), Lauritzen & Wermuth (1989), Wermuth (1988), Wermuth & Lauritzen (1990), Edwards (1990), Frydenberg & Lauritzen (1989), Frydenberg (1989) and Frydenberg (1990).

With the growing use of the above models, the demands for eŒcient state-of-art software for handling the graphical models have become more intense.

The Parts of the Thesis

This thesis concerns the development of eŒcient algorithms for estimation in contingency tables, implementation of these algorithms in a computer program together with procedures for model selection, and inclusion of the program in a modern system for statistical analysis. The class of models is limited to models on discrete variables.

The thesis can be divided into three parts:

Part 1: EŒcient algorithms for analysis of contingency tables:

This ørst part of the thesis consists of three papers: A paper describing how decomposition is crucial for achieving eŒcient algorithms for estimation in the log-linear models and also describing a space saving implementation of the necessary iterative algorithm for non-decomposable models, a paper on algorithms for solving the graph theoretical problems around decomposition of the models, and ønally a paper on algorithms for collapsing log-linear models.

Part 2: The program CoCo:

CoCo is a program for estimation, test and model search among hierarchical interaction models on large contingency tables. In this program the described methods for estimation are implemented. Badsberg (1991) and Part II is a guide to this program. This part also contains a small paper on the model selection strategies implemented in CoCo.

Part 3: Integration of CoCo in XLISP-STAT:

In the third part CoCo is included in the Lisp system XLISP-STAT. This enables a graphical user interface to CoCo, the ability to write functions for doing further computations on estimates and other results from CoCo, handling of causal models, etc.

The study has been partly theoretical and partly practical. CoCo and the integration of CoCo in XLISP-STAT is the result of the practical part. This software is available by ftp, see Part II or III.

The following three sections give a resume of each of the parts of the thesis:

1.2 EŒcient Algorithms for Analysis of Contingency Tables

1.2.1 Computation of Maximum Likelihood Estimates, Marginal Tables and the Deviance in Log-Linear Models

The Problem

Darroch et al. (1980) deøned graphical models for contingency tables, where every vertex of a graph is associated with a discrete random variable, and a missing edge in the graph corresponds to the conditional independence of the two variables associated with the two vertices of the missing edge. If a graph corresponds to a graphical model for a contingency table, and the graph is decomposable into subgraphs by a clique separator, then the maximum likelihood estimates for the parameters of the model can easily be derived from the models on the lower-dimensional tables and represented by the simpler subgraphs. The complexity of algorithms for computing the maximum likelihood estimates in log-linear models without decomposition is exponential in the dimension of the table, and thus the divide-and-conquer approach based on the decompositions will enable handling of much larger tables. For hierarchical log-linear models on contingency tables analogous results can be achieved by decomposing the hypergraph associated with the hierarchical model. Decompositions of hypergraphs are deøned in Lauritzen, Speed & Vijayan (1984).

Solution, Other Works and Applications

By application of graph-theoretical results, closed-form expressions of the maximum likelihood estimates in decomposable log-linear models for contingency tables are found. For the remaining graphical and hierarchical models the problem of ønding the estimates is reduced by decomposition of the models into the irreducible components, an approach also suggested in Malvestuto (1989). The application of the iterative proportional scaling procedure, the IPS procedure, is limited to these irreducible components, and by representing the distributions of the irreducible components in an economical way as described in Jirouek (1991), the iterative procedure is optimized. The IPS-algorithm is also known as the iterative proportional øtting or the Deming-Stephan algorithm (Deming & Stephan 1940, Fienberg 1970, Darroch & Ratcliœ 1972). An implementation of the IPS-algorithm is given in Haberman (1972).

In chapter 2 it is argued that by these methods it is possible to compute the deviance in hierarchical models for several hundreds of variables in each irreducible component (under some constraints). Also exact computations are possible in expert systems based on causal probabilistic networks by exploiting these methods (Lauritzen & Spiegelhalter 1988).

The problems of computing marginal tables of counts and estimated values are also addressed in chapter 2.

A crucial point of the complexity of the space saving implementation of the IPS-algorithm is ønding a øll-in giving a small state space, a problem considered in Rose (1970), Rose (1973), Beeri, Fagin, Maier & Yannakakis (1983), Wen (1990), Kjrulœ (1990) and Kjrulœ (1992). Junction trees can be used in the implementation of the space saving algorithm, see Jensen (1988) and Jensen & Jensen (1994) about junction trees.

1.2.2 Decomposition of Graphs and Hypergraphs with Identification of Conformal Hypergraphs

The Problem

The graph theoretical problem of decomposing log-linear models is considered in chapter 3.

Other Works

In Geng (1989) the algorithm HiModel is presented. The algorithm checks whether or not a log-linear model is decomposable, collapses the model onto a given set, and decomposes the generating class into sub-generating classes as small as possible. The algorithm has a complexity around $O(nm^3)$ of a graph with n vertices and m generators.

Solution

But a complexity of $O(ne+nm\min(n,m))$ for identifying and decomposing models can be achieved by the algorithms presented in Rose, Tarjan & Lueker (1976), Tarjan & Yannakakis (1984) and Tarjan (1985). If the model class is restricted to graphical models, then decomposable models can be identified and decomposed in $O(e+nm\log m)$ with $m \leq n$, e is the number of edges in the 2-section graph. Compared to the complexity around $O(nm^3)$ of the algorithm of Geng (1989) this is much better, unless $e > nm^3$, i.e., the number of edges in the 2-section graph is very large compared to the number of edges in the hypergraph. Working directly on the hypergraph, decomposable models can be decomposed and the index of the graph computed in $O(n + m' + nw\log w)$, m' the total size of the generating class. Graphical models can often be handled in $O(ne + nm + nw\log w)$ with $w < \min(n,m)$ the number of decompositions with respect to minimal separators.

The algorithms for decomposing the model is based on the algorithm of Tarjan (1985) for finding the clique separators of a graph, an algorithm with a complexity of $O(ne+n^2)$. An optimal version of this algorithm, in the sense that the separators are minimal and that the graph is only decomposed into the irreducible components, is presented in Leimer (1993). In Tarjan (1985) and Leimer (1993) algorithms for finding the vertex-sets of the irreducible components of the graphs are given. In chapter 3 also the problem of decomposing hypergraphs and the problem of finding the cliques (generators) of the irreducible components are considered. Furthermore, we for each minimal clique separator count the number of components of the graph resulting from eliminating the separator.

1.2.3 Algorithms for Collapsing Log-Linear Models onto the Smallest Set Containing a Given Set

The Problem

Asmussen & Edwards (1983) deøned necessary and suŒcient conditions for collapsibility of hierarchical log-linear models for a multidimensional contingency table.

The problem addressed in chapter 4 is as follows: given a graphical (or a hierarchical) log-linear model $\mathcal{M}$ and a subset of the factors of the model, what is the minimal set containing the given subset, such that $\mathcal{M}$ is collapsed onto the set? The problem solved in this chapter can be given a pure discrete mathematical formulation: Given a graph (or hypergraph) and a subset of the set of vertices of the graph, what is the smallest set containing the given set such that the boundary of each connected component of the graph induced by the complement of the found set is complete (contained in an edge of the hypergraph)?

In chapter 4 two algorithms are presented. For graphical models and hierarchical models respectively we give an algorithm to ønd the smallest set containing a given set and onto which the model is collapsible.

Other Works

This set can be found by the algorithm HiModel of Geng (1989). This algorithm has a complexity of at least $O(nm^3)$, with m the number of generators of the model and n the number of variables. In Madigan & Mosurski (1990) an algorithm solving the problem in $O(m')$ time for decomposable models only is considered. m' is the total size of the generating class of the model. They show that an algorithm for selectively reducing an acyclic hypergraph given by Tarjan & Yannakakis (1984), in context of answering queries in relation databases, can be used to solve the problem here considered for decomposable models.

Solution and Applications

In chapter 4 an algorithm for solving the problem for graphical models in $O(ne)$ time is presented. By a simple modiøcation the algorithm can also handle non-graphical models in $O(ne + nmw)$. Here e is the number of edges in the 2-section graph of the model and w is the number of decompositions with respect to minimal separators. The algorithm uses the notion of graphs with marked vertices presented in Leimer (1989).

Madigan & Mosurski (1990) write that their algorithm has applications in expert systems: In the context of expert systems, by reducing a probability inÆuence network onto only the relevant nodes, the algorithms reduce the required computation and simplify interpretation. Chapter 4 gives eŒcient algorithms for an extended class of problems. The algorithm of this chapter is also crucial in computation of an adjustment of the number of degrees of freedom in tests between two log-linear models, see Badsberg (1991) or Part II.

1.3 CoCo

1.3.1 Introduction

CoCo is a program for estimation, test and model search among hierarchical interaction models for large contingency tables. CoCo works especially eŒciently on graphical models, and some of the commands are designed to handle graphical models, but also non-graphical models can be handled. Badsberg (1991) and Part II is a guide to this program. The work on CoCo was initiated in the master thesis Badsberg (1986). The name CoCo is derived from ıCoȷmplete ıCoȷntingency tables, since the initial program could only handled complete tables, but the program has been enhanced to handle incomplete tables as well.

CoCo is a program designed to perform estimation and tests in large contingency tables. By using graph-theoretical results (Rose et al. 1976, Tarjan & Yannakakis 1984, Tarjan 1985, Leimer 1993) the hierarchical log-linear interaction models are decomposed. The IPS-algorithm is not used on the full table, but only on the non-decomposable irreducible components (chapter 2). Furthermore, the optimized version of the IPS-algorithm of Jirouek (1991) is used on these non-decomposable atoms.

If one model is tested against another and the two models have common decompositions, then the test can be partitioned in tests on smaller tables (Goodman 1971). Collapsibility (Asmussen & Edwards 1983) of models and tests is used. If two large sparse models (many factors but few interactions) have no common decompositions, they can be tested against each other by computing the deviance for each model, not by summing over all cells in the full table, but by summing over only cells in suŒcient marginal tables (chapter 2).

By using these methods the following procedures, depending of the size of the tables, are useful:

Size of Tables

Tiny tables:	2 and 3 dimensional tables;
	In these tables a wide range of measures of associations on the 2-dimensional tables given other variables can be computed.
Small tables:	tables with up to between 7 and 10 variables;
	On small tables the global model search procedure of Edwards & Havrnek (1985) is useful, and will terminate after an acceptable computing time. Also exact tests by Monte Carlo simulation and the EM-algorithm are useful on these tables.
	Before any computation all the marginal tables are found to reduce the computing time (unless some cases have unobserved variables).
Medium tables:	tables with up to 20 variables;
	Any test between two hierarchical models can be performed. The procedures for backward elimination and forward selection of edges in graphical models are useful.

In these tables only suŒcient tables of observed counts and tables of estimated probabilities for few tables can be stored internally in the computer.

Large tables: tables with more than 20 variables;

We do not say that all models on large tables are large, e.g., a model with only main eœects is called a small model. If all the tables of the suŒcient marginals of a model cannot øt in the computer memory at one time together with the state spaces of all the non-decomposable components of the model, then the model is large. A model is very large, if any of the suŒcient marginal tables cannot øt in the computer memory. We cannot handle very large models, if we cannot at least store the state spaces of the non-decomposable components of the model one by one. Such models are called huge models. Thus, in very large models the largest clique of the øll-in graphs of non-decomposable atoms of the graph of the very large model cannot have more than 14 binary vertices (24 on workstations).

One model can be tested against another by using partitioning and collapsibility of tests, or by computing the deviance for each model by summing over only non-zero cells in the suŒcient marginal tables as described in chapter 2 . These tests can be performed between any two large (or very large) models, but if the parts of the test involve tests on large tables, then only the deviance can be computed, and the adjusted degrees of freedom cannot be computed, if the parts involve large models.

The procedures for backward elimination and forward selection of edges in graphical models are still useful on large tables. (Various runs of model selection by respectively forward selection and backward elimination has been performed on the 121 dimensional table with 10.000 cases of Wedelin (1993).)

In large tables the observations are placed as a case list on an internal øle.

This crude classiøcation of tables follows the classiøcation of datasets by Huber (1994): Tiny datasets are suitable for black-boards, a small set can be printed on a few pages, a medium set øts on a Æoppy disk, a large dataset requires a tape, and a huge dataset requires many tapes.

Model Control and Data Selection

Besides functions for tests and model search CoCo also has some procedures for model control. The test of one decomposable model against another decomposable model can be factorized into a sequence of tests with one edge (Sundberg 1975, Frydenberg & Lauritzen 1989), and the test between any two submodels can be factorized in tests with one interaction. Functions for producing these factorizations are available in

CoCo. CoCo also has a function for computing a wide range of measures of associations on 2-dimensional tables.

Observed counts, estimated counts and probabilities and residual (absolute, adjusted, standard, Freeman-Tukey, etc.) can be listed, plotted pairwise, printed in tables and given a univariate description with mean, variance, median, range, etc.

Finally, to make the use of CoCo somewhat easier and more Æexible some data selection is possible and cases with missing values (incomplete observations) can be excluded at various levels in CoCo or the EM-algorithm applied. CoCo can exclude all observations with any variable marked as unobserved, when reading the observations, exclude observations with variables unobserved in a given set, after reading the observations, or exclude observations with relevant variables for a given test marked as unobserved, when performing the test.

1.3.2 Model Selection

In the paper Badsberg (1992) a short presentation of the two main model selection strategies of CoCo is given: The incremental search by backward elimination and forward selection and the global search procedure, the EH-procedure, from Edwards & Havrnek (1985). By incremental search a single minimal acceptable model is identiøed. By the principles of weak acceptance and weak rejection the class of minimal acceptable models are found in the EH-procedure. In CoCo each of the model searches can be done by a single command, or CoCo can be guided through the search in a highly user controlled fashion.

Selection Criteria

In the model selection in CoCo the acceptance and rejection of a model can be based on the likelihood ratio test statistic, Pearson's χ^2 test statistic or the power divergence statistics (Read & Cressie 1988). For these test-statistics asymptotic p-values can be computed using either the an unadjusted degrees of freedom or using an adjusted number of degrees of freedom, or in tests between any two nested decomposable models exact p-values can be computed by Monte Carlo simulation.

Instead of p-values information criteria can be applied, e.g., Akaikes information criteria (Akaike 1974, Sakamoto & Akaike 1978) or the Bayesian information criteria (Schwarz 1978).

Variables can also be declared as ordinal, and when two ordinal variables are tested conditionally independent, Goodman and Kruskal's Gamma coeŒcient can be computed. The acceptance of a model can then be based on asymptotic or exact p-values of this coeŒcient.

Incremental Search

The incremental search in CoCo is performed by forward inclusion (Dempster 1972) and backward elimination (Wermuth 1976b) of edges (or interaction-terms). In the backward elimination of edges in graphical models, edges are sequentially eliminated from the independence graph. In the forward selection the edges are sequentially added to the independence graph.

Wermuth (1976b) considers stepwise edge elimination on decomposable models. Two recent books Christensen (1990) and Whittaker (1990) each contain a chapter on model selection. In Whittaker (1990) with the title ıGraphical Modelsȷ the model selection is of course treated in relation to graphical models, but also in Christensen (1990) an excellent discussion of model search on graphical log-linear models is given. Edwards (1993) considers some computational aspects of both the stepwise edge selection and the EH-procedure.

Incremental Search in CoCo

In CoCo the backward elimination and the forward selection can be started from any initial model. The backward elimination procedure is not restricted to work on edges, but can also perform stepwise elimination of interaction terms. Analogously with the forward selection. Steps, the action of visiting a set of edges and then, based on acceptance/rejection of the edges, remove or add a subset of the edges to the considered model, of the backward elimination and the forward selection can be mixed together in any order, and between each step any set of edges or interaction terms can be added or eliminated. Thus, the functions for stepwise edge selection can, together with some model-editing commands for adding and removing edges and interaction, be used in a highly user-controlled semi-automatic interactive model search. The procedures can be used to identify a single (or a few) minimal acceptable model(s).

The incremental search procedures can be controlled by several options: In backward elimination local tests, i.e., tests against the previously accepted model, or global tests, i.e., tests against a øxed base model, can be applied. Coherence can be applied, i.e., in backward elimination, once a model is rejected, no sub-models of the rejected model are considered. In forward selection elimination, in each step, either all edges signiøcant can be added, or only the most signiøcant edge added. Analogously in backward elimination, in each step, either only the least signiøcant, or all non-signiøcant edges are removed. Also options for additional model control are available: e.g., in backward elimination, for each edge to remove, besides the global test also test whether the pair of variables of the edge is conditionally independent given each minimal separator.

A recursive backward elimination strategy, where the ørst edge found to be non-signiøcant is eliminated in each step is implemented in CoCo. In each step of this headlong backward elimination, edges with p-value less than a given limit (e.g. 1% or 5%) are rejected, and, when the principle of weak rejection is applied, the are not considered in sub-sequential steps. The ørst edge found in each step with p-value greater than a second limit (e.g. 20%) is removed. Visited edges in the current step with a found p-value between the two limits and all not visited eligible edges in the current step are eligible in the next step. This gives a faster elimination than a backward elimination where all eligible edges in each step have to be visited to ønd the least signiøcant edge. Analogously a headlong forward selection is implemented.

Not to favour the removal of edges with, e.g., a low lexicographical order, the edges in this headlong backward elimination are visited in a random order. Hence several calls to the procedure might give diœerent results, and a sample of models with all edges signiøcant to a selected level of signiøcance can be generated.

The Model Selection in CoCo Compared with Model Selection in Lisp

The result of the model selection considered in this section, the model selection in the stand-alone version of CoCo, is printed on standard output and models are added to a list of models. In the Lisp Stat environment extended with CoCo, see the next chapter, also a function for stepwise edge selection on association diagrams is implemented. The results of the model selection on association diagrams can be new association diagrams.

For performing the same model selection, the selection in CoCo is the fastest, since it is performed in a compiled program, but in addition to giving a visual picture of the selection, the model selection on association diagrams is more Æexible: E.g., the user can program the selection criteria, and model selection on block-recursive models is possible.

Only in the model selection in the stand-alone CoCo forward selection and backward elimination of interaction terms are possible, and for each edge/interaction-term tested some model control is available.

The Model List

The resulting models from each cycle of the backward elimination and the resulting model from the forward selection are added to a linked list of models in CoCo. This list of models in CoCo has been found to be very useful. Besides models generated by the incremental search procedures the model list also contains all models read into CoCo. The list of models helps keeping track of the search. The models in the list can be edited: edges or interaction-terms added or removed, the models tested against each other, øtted values in models tabulated, plotted, etc.

Global Search

By the principles of weak acceptance and weak rejection, coherence, the search space of all hierarchical models on the contingency table is divided into two sets of models: the class of minimal acceptable models and the class of maximal rejected models. Hence, after using the 'Fast Procedure for Model Search', the EH-procedure, by Edwards & Havrnek (1985) and Edwards & Havrnek (1987) any model can be labeled as accepted or rejected.

By the EH-procedure the models (or sub-models of a speciøc base model) are classiøed into two regions $\mathcal{A}$ and $\mathcal{R}$: weakly accepted models and weakly rejected models. Weakly accepted models are models that include an accepted model, and weakly rejected models are models that are included in a rejected model. In turn, a boundary below the weakly accepted models and a boundary above the weakly rejected models, the R-dual $D_R(\mathcal{A})$ and the A-dual $D_A(\mathcal{R})$ respectively, are øtted. The search stops when either all models in $D_A(\mathcal{R}) \setminus \mathcal{A}$ are accepted or all models in $D_R(\mathcal{A}) \setminus \mathcal{R}$ are rejected.

In CoCo the global search can be started with the accepted class containing the saturated model and the class of rejected models containing the model with only main-eœects, or any models can be entered into the two classes. Three strategies for selecting the dual to øt is implemented. The EH-procedure is implemented in CoCo both in the graphical and the hierarchical version.

1.3.3 Miscellaneous

Exact Tests

In large sparse tables the approximate p-values are completely unreliable (Kreiner 1987). Thus exact tests are needed.

Mehta & Patel (1983) give a network algorithm for performing Fisher's exact test in $r \times c$ contingency tables, and Morgan & Blumenstein (1991) give an algorithm for exact conditional tests for hierarchical models in multidimensional contingency tables. Both algorithms depend on complete enumeration, and will thus give the exact p-value, but the feasibility of using the algorithms is thus limited to what is here termed very small tables.

But to any degree of accuracy approximation to exact p-values for tests between any two nested decomposable models can be achieved by Monte Carlo simulation.

Pateøeld (1981) give an eŒcient method of generating random $r \times c$ tables with given row and column totals. In Kreiner (1987) this algorithm is used to calculate approximations of exact p-values for tests for zero partial association, i.e., tests of two variables conditionally independent.

The Monte Carlo algorithm of Pateøeld (1981) can also be extended to give approximation to any degree of accuracy of exact p-values for tests between any two nested decomposable models (Lauritzen 1993). This is implemented in CoCo.

Also eŒcient methods for handling non-decomposable models are needed.

Incomplete Tables and Initial Values to the IPS-algorithm

Among the applications of arbitrary initial values for the IPS-algorithm is the ability to handle incomplete tables, i.e., tables with ıstructural zerosȷ. Thus, tables of initial values for the IPS-algorithm can be entered into CoCo, and a modiøed version of the IPS-algorithm is used on these tables.

Adjustment of the Number of Degrees of Freedom

In sparse tables, i.e., tables with a considerable number of cells with zero counts, and incomplete tables, an adjustment of the number of degrees of freedom of a test is necessary, if the dimensions of the models are computed by the usual formulas, e.g., the formula given in Lauritzen (1982) for the dimension of log-linear models without vanishing cell probabilities.

Haberman (1974) gives an algorithm for computing dimension of a log-linear model adjusted for zero cells. The computations involve Gaussian elimination on a set of size the unadjusted dimension of the model of vectors with as many entries as the table. Haberman (1974) notes that ıthe sort of calculations involved are hardly suitable for the back of an envelopeȷ.

Badsberg (1991) (Part II) gives a short formula for computing an adjusted number of the degrees of freedom of a test. The calculation of this adjustment, implemented in CoCo, involves visiting cells of expected suŒcient marginal tables, and can be computed also for fairly large tables. But the adjustment is probably only correct for decomposable models on non-separable tables.

EM-algorithm

Besides excluding cases with missing values (incomplete observations) at various levels, CoCo can also handle incomplete observations by the EM-algorithm. The implementation given in CoCo is straight forward after the deønition of the EM-algorithm, see, e.g., Fuchs (1982).

The EM-algorithm can be made more eŒcient by the methods described in Lauritzen (1991), Geng & Asano (1988) and Geng, Asano, Ichimura & Kimura (1993).

The EM-algorithm in CoCo is tested on data from Fuchs (1982) for an example with sporadic missing observations, and on Dawid & Skene (1979) for an example with latent class variables.

1.4 CoCo within XLISP-STAT and Association Diagrams

The guide Part III is about CoCo within XLISP-STAT. The system ıXlisp+CoCo — A Tool for Graphical Modelsȷ is obtained by loading CoCo into XLISP-STAT. This gives a graphical user interface to CoCo, and a model selection on graphical models can be performed by mouse interaction with plots of independence graphs.

The purpose of loading CoCo into XLISP-STAT is, besides Xlisp+CoCo to allow the end user to write Lisp programs using the procedures of CoCo as functions, to be able to do further computations on tables of øtted values and statistics computed in CoCo, plot output from CoCo by high-resolution graphics or, e.g., do a model search by a strategy implemented by the end user in Lisp, etc.

Why XLISP-STAT?

In Tierney (1990) of course a lot of reasons are given for using Lisp as a platform for a computer environment for statistical computing. Some of them are contained in the following:

- Interactive: The days of computer interaction limited to punctuation cards and batch jobs are long gone. In a computer environment for statistical computation it must be possible to write expressions such as, e.g., the mean of the elements of a vector divided by the square rote of the variance of the same elements, or a matrix product involving the inverse of a matrix. The values of these expressions should appear immediately on the screen without the tiresome task of, e.g., calling a compiler. Thus the computer environment for statistical computation must contain a high-level interactive programming language.
- Incremental: One may need to evaluate the same expressions over and over again, but on diœerent sets of data, and thus it should be possible to write functions in the computer environment, i.e., the language should be incremental.
- Object oriented: Experience has shown that object oriented programming is very useful for graphics programming in general and statistical graphics programming in particular (Stuetzle 1987, McDonald & Pedersen 1988, Hitz &

Hudec 1994). Object-oriented methods have also many other applications within a statistical system (Tierney 1990). They can be used for developing Æexible data structures (McDonald 1986, Stuetzle 1987) and for representing statistical models (Oldford & Peters 1988).

Instead of using an object oriented language one could have used a functional programming language. This approach is for generalized linear models tried in Gilchrist & Scallan (1988). Functional languages are in some sense the opposite of object-oriented languages. In a functional language there is no state, i.e., no objects holding variables, so there can be no confusion about which values are owned be which objects, and thus debugging is easier. But graphics programming would be a challenge in a functional language, since there is no way of storing the characteristics of the graph windows in a functional language.

- Extendable: A very important feature of the computer environment for this current project is that the language is extendable, meaning that functions and procedures written in some other languages, e.g., C, PASCAL and FORTRAN, can be loaded into the computer environment after compilation, and then used as builtin functions in the computer environment.

- ıStatistical functionsȷ: Vectorized algebra, matrices and operations on these, probability functions, graphics, etc. These are some of the features, important for statisticians, added to Xlisp by Luke Tierney to get XLISP-STAT.

Several other languages have been suggested and used as the basis for statistical environments. A language that has received considerable attention is APL. APL has many useful features especially for statisticians, including a wide range of functions for handling matrices and arrays. But it does not have the ability to easily handle high-level data, such as functions or expressions, nor does it lead itself readily to support the object-oriented programming style that is so important for graphical programming. Adding these features to APL appears to be considerably harder than adding matrices and functions to Lisp (Tierney 1990).

In stead of adapting an existing language for statistical computation one could develop a new high-level language from scratch. This approach has been taken in developing the S system (Becker, Chambers & Wilks 1988, Chambers & Hastie 1991).

The S system is also extendable. CoCo can be loaded into this system, and the most basic interface functions have been made. But S-Plus does not support graphics programming as well as XLISP-STAT, and I found the programming in XLISP-STAT more elegant than the programming in S-Plus. Thus the work of writing interface functions between CoCo and S-Plus has been terminated and no graphical interface is made.

1.4.1 Association Diagrams

The association diagram is a graph window with the independence graph of an association model. The model can be causal or not, and the variables can be discrete, ordinal, continuous etc. The independence graph, association or interaction graph, is a simple undirected graph with as many vertices as the table has dimension. A vertex for each variable. Two vertices are adjacent in the independence graph for the model unless the two variables are conditionally independent given the other variables.

The association diagram is a very appealing working tool for model selection. The layout of the graph, the association diagram, can be edited by dragging vertices, points for variables, with the mouse, and the edges will then follow the vertices. New association diagrams can be created from a diagram by adding or dropping edges by the mouse. Tests of two variables conditionally independent can be performed by clicking on the edge between the two variables by the mouse. Edges are then drawn with a width proportional to the signiøcance of the edges. Edges can also be labeled with a value computed from a test statistic for removing the edge, e.g., a p-value or an IC-value. Causal models can be handled in the association diagrams. Methods for backward elimination and forward selection of edges in association diagrams are implemented. In the backward elimination of edges the edges are redrawn with a width proportional to the signiøcance of the edges as they are visited. The edge will be drawn with a color depending on whether the edge is øtted or not. If the test of an edge is not computed, then the edge will be drawn with a color depending on whether the edge is øxed, the edge is rejected by coherence, removing the edge will result in a non-decomposable model, etc.

1.4.2 Block Recursive Models

Block recursive models, Chain graph models, are models with both symmetric and asymmetric associations between variables. The symmetric associations have been dealt with so far in this thesis.

At the asymmetric association, directed association, between two variables, the second variable might depend on the ørst variable, but the ørst variable is independent of the second variable. The ørst variable is then explanatory to the second variable, the response variable. In the graph we then draw an arrow from the ørst variable to the second variable. The explanatory variables are also called the exogenous variables and the response variables the endogenous variables. The endogenous variables have an undirected graph associated with them. Each endogenous factor is the end point for one or more direct edges. The directed edges can originate at either exogenous or endogenous variables.

In recursive models response variables of some explanatory variables cannot be explanatory variables to the same explanatory variables, i.e., in the graph there cannot be any oriented cycles, cycles where one goes along undirected edges and in the direction of arrows.

Conditional independence statements for recursive causal models are examined in Wermuth & Lauritzen (1983). The so-called block recursive graphical models or graphical chain models are further treated in Lauritzen & Wermuth (1989), Lauritzen (1989), Wermuth & Lauritzen (1990), Lauritzen, Dawid, Larsen & Leimer (1990). The chain graph Markov property is investigated in detail in Frydenberg (1989). Model selection methods for creating a diagnostic system by causal models on discrete variables are discussed in Lauritzen, Thiesson & Spiegelhalter (1992). Also the text books Lauritzen (1993), Whittaker (1990) and Christensen (1990) contain chapters on block recursive models.

The tool for association diagrams implemented in the Lisp extension of CoCo are able to handle block recursive models. In these diagrams incremental model search on block recursive models can be performed by backward elimination and forward selection.

1.4.3 Miscellaneous

Besides introducing association diagrams with the ability to handle block recursive models and dumping TeX picture-code for the diagrams, this tool also introduces the model dynamic spinning plot and the model manager. Also a method for doing a model selection with resampling is implemented.

Model Dynamic Spinning Plots

The model dynamic spinning plot is a spinning plot, where the values in the plot are updated when models are modiøed. Spinning plots are rotatable three-dimensional scatterplots. In the model dynamic spinning plot each of the variables are linked to a model, and when the model is modiøed the plot is redrawn with the new values of the model. These spinning plots can, of course, be linked to other spinning plots, histograms, etc. and points of one plot highlighted when the corresponding points are brushed in other plots or histograms.

Model Manager

The model search by association diagrams may generate numerous windows with graphs, and to handle these windows, the model manager has been made. The model manager is an overview graph, where each point in the graph corresponds to a model. Tests can be performed by dragging edges between points in the model manager.

Model Selection with Resampling

To show the advantages of including CoCo in XLISP-STAT, a method for resampling from the case list, and do a backward elimination on each sample, has been implemented. The edges of the association diagram can be drawn with a width growing with the reciprocal of the p-value or growing with BIC, the Bayesian information criterion. But we can also let the width of the edges be proportional to the number of times the edges are found in the ønal models of headlong backward eliminations on random subsets of the cases.

This method is implemented in a few lines of Lisp code, and the full implementation is shown in part III, ıXlisp+CoCo — A Tool for Graphical Modelsȷ, of the thesis as an example of what can be made by the end user in XLISP-STAT extended with CoCo.

1.4.4 Discussion

The current tool may be considered an ad-hoc development of CoCo.

To handle mixed interaction models another approach should be adopted. The tool should be built around the hierarchy of models. For each class of models an estimation function should be implemented. The CoCo object with the data should be removed. Functions for øtting models, returning øtted values from models, testing models against each other and search routines should be extracted from the code of CoCo. Necessary data to the resulting functions should be given as arguments to the functions. This will be at the cost of some computational speed, e.g., because the code for reusing already computed tests thus cannot be implemented as eŒciently as in CoCo.

It should be possible to put variables (objects with information whether they are discrete, ordinal, continuous, which variables the variable is a response variable to etc.) into a model, and then the model object should be specialized appropriately to whether the model is a general mixed interaction model, a linear regression, a log linear model on a contingency table, a block recursive model, etc. These variable objects could also know where to position themselves in the association diagram, etc. Hierarchy of models is discussed in Anglin & Oldford (1993) and Tierney (1991). The notion of data frames of Chambers & Hastie (1991), i.e., a collection of named vectors of the same length, may also be useful here.

- Speed: The current implementation of the association diagrams has been made to investigate the usefulness of these diagrams, and the diagrams are implemented so new features can easily be added, but no attempts to make fast code have been made.

  The performance of the current implementation is øne for small graphs, i.e., graphs with less than 10 variables. Editing the layout of a graph with 40 variables is bearable. The code for the drawing of the graphs should be optimized.

- Variables as objects: Variables should be objects (Oldford & Peters 1986, Oldford 1988), and when the variables are put into a model, the model object should be created appropriately as a pure discrete model (a CoCo model), a linear regression, a mixed interaction model, a block recursive model, etc. as discussed above.

- Graphical interface to variables: It should be possible to click the vertex for opening a dialog window for setting vertex position, vertex label, position of vertex label, variable type, stratum, list marginal table, list values, etc.

- Model formulas: Besides building the models by graphical interaction it should also be possible to specify the models by commands. An extension of the GLIM-language (Wilkinson & Rogers 1979) would be nice. This language is also used in Chambers & Hastie (1991). A language for mixed interaction models is considered in Edwards (1989) and Edwards (1990).

- Mixed models: The tool should be extended to handle mixed models: The symbols should depend on the type of variable: Continuous (circles) and discrete (dots). The association diagram is ready for this change. Discrete nominal variables are drawn with dots and ordinals with squares.

  The problem is the estimation in the mixed causal models and the hierarchy of models. Algorithms for estimation in mixed undirected models are implemented in MIM (Edwards 1989).

- Hypergraphs: The tool should be extended to handle hierarchical non-graphical models. It should also be possible to draw graphs as hypergraphs.

- Methods without graphs: It should be possible to declare a CoCo model object as a block recursive model, and then perform a model search among block recursive models on the model without the graph. This will also enable a model search on undirected models by a criteria programmed by the end user.

Kapitel 2

Computation of Maximum Likelihood Estimates, Marginal Tables and the Deviance in Log-Linear Models

Abstract: By application of graph-theoretical results, closed-form expressions of the maximum likelihood estimates in decomposable log-linear models for contingency tables are found. For the remaining graphical and hierarchical models the problem of ønding the estimates is reduced by decomposition of the models into irreducible components. The application of the iterative proportional øtting procedure is limited to these irreducible components, and by representing the distributions of the irreducible components in an economical way, the iterative procedure is optimized.

By these methods it is possible to compute the deviance in hierarchical models for several hundreds of variables in each irreducible component (under some constraints).

The problems of computing marginal tables of counts and estimated values are also addressed.

Key Words: Contingency Tables, Hierarchical Log-linear Model, Decomposition, Irreducible Components, Junction Tree, Fill-In, Iterative Proportional Fitting.

2.1 Introduction

In model selection for contingency tables (Edwards & Havrnek 1985, Edwards 1993) eŒcient algorithms for computing the maximum likelihood estimates are needed. The methods considered in this chapter will enable estimation and computation of the deviance in tables with several hundreds of variables. Also exact computations are possible in expert systems based on causal probabilistic networks by application of the methods (Lauritzen & Spiegelhalter 1988).

Darroch et al. (1980) deøned graphical models for contingency tables, where every vertex of a graph is associated with a discrete random variable, and a missing edge in the graph corresponds to the conditional independence of the two variables associated with the two vertices of the missing edge. If a graph corresponds to a graphical model for a contingency table, and the graph can be decomposed into subgraphs by a clique separator, then the maximum likelihood estimates for the parameters of the model can easily be derived from the models on the lower-dimensional tables and represented by the simpler subgraphs. The complexity of algorithms for computing the maximum likelihood estimates in log-linear models without decomposition is exponential in the dimension of the table, and thus the divide-and-conquer approach based on the decomposition algorithms of this chapter will enable handling of much larger tables. For example, by these methods it is possible to compute the deviance in hierarchical models with several hundreds of variables in each irreducible component (under some constraints to be given later). For hierarchical log-linear models on contingency tables analogous results can be achieved by decomposing the hypergraph associated with the hierarchical model.

The graph theoretical problem of decomposing log-linear models is considered in chapter 3. Based on Tarjan & Yannakakis (1984), Tarjan (1985) and Leimer (1993) algorithms are given for ønding the irreducible components of hypergraphs and the generating classes of these are presented. These algorithms for decomposing the models and the optimized IPS-algorithm of Jirouek (1991) are implemented in the program CoCo (Badsberg 1991), see chapter 5 and Part II and III.

2.2 Preliminaries

Contingency Tables

Consider a ønite set of classifying factors, variables, Δ. Let $i \in \mathcal{I} = \times_{\delta\in\Delta}\mathcal{I}_\delta$ and $i_a \in \mathcal{I}_{\mathrm{a}} = \times_{\delta\in\mathrm{a}}\mathcal{I}_\delta$ be vectors of indices, where $\mathcal{I}_\delta$ is the possible levels of the variable δ, $\delta \in \Delta$, and a is a subset of the variables Δ. Write the observed counts in the cell i and in marginal cell i_a as $N(i)$ and $N(i_a)$ respectively. The total count is denoted by N or $N(i_\emptyset)$. We are interested in the probabilities $p(i)$ of an object falling in the cell i, and the corresponding marginal probabilities $p(i_a)$ formed by summing $p(i)$ over a^c. The table of counts is assumed to be multinomially distributed. A generating class is a set of subsets of a ønite set such that no element in the generating class is a subset of another element. If $\mathcal{M}$ is a generating class, then a probability distribution p belongs to the hierarchical log-linear model $\mathcal{M}$, if and only if

$$\log p(i) = \sum_{a\subseteq\Delta} \phi_a(i), \tag{2.1}$$

where $\phi_a(i) = 0$ for all $i_a \in \mathcal{I}_{\mathrm{a}}$ unless $a \subseteq c$ for some $c \in \mathcal{M}$. In this chapter we consider limits of such probabilities denoted by $\mathcal{M}^\star$. Let $\mathcal{M}_{\mathrm{a}}$ denote the hierarchical model with the sub generating class which is formed by restricting $\mathcal{M}$ to a, i.e., deleting all variables not in a from the generating class $\mathcal{M}$ and subsequently, from the resulting set, removing elements which are subsets of other elements. Denote the maximum likelihood estimates of the probability in the cell i_a under $\mathcal{M}_{\mathrm{a}}$ as $\hat{p}_a(i_a)$.

For a given log-linear model $\mathcal{M}$ we define the interaction graph $G = (V(G), E(G))$ of $\mathcal{M}$ as the undirected graph whose vertices $V(G) \subseteq \Delta$ correspond to the classifying factors Δ and whose edges $E(G)$ are given by the 2-factor interactions present in the model. The interaction graph is the 2-section graph of the generating class $\mathcal{M}$. One may interpret the interaction graph in the following way, (Darroch et al. 1980, Asmussen & Edwards 1983): If two disjoint subsets of vertices a and b are separated by a subset c in the sense that all paths from a to b go through c, then the variables in a are conditionally independent of those in b given the variables in c.

Clearly, many different log-linear models may have the same interaction graph, as long as they contain the same 2-factor interactions. Models with the maximal permissible higher-order interactions corresponding to a given graph are termed graphical models; it is shown by Darroch et al. (1980) that all decomposable models are graphical. More specifically, decomposable models are graphical models whose graphs are triangulated, i.e., contain no cycles of length greater than 3 without a chord.

Graphs

In this chapter we consider simple undirected graphs $G = (V(G), E(G))$ with vertices $V(G) \subseteq \Delta$ and edges $E(G)$. A graph is simple, if it does not contain multiple edges and loops, i.e., no identical edges and no edges with identical vertices.

A path in G between vertices $v, w \in V(G)$ is a sequence $v_0, v_1, \cdots, v_n \in V(G)$ with $\{v, w\} = \{v_0, v_n\}$ and $\{v_{i-1}, v_i\} \in E(G)$ for $i = 1, 2, \cdots, n$. Two vertex sets $a \subseteq V(G)$ and $b \subseteq V(G)$ are separated by $c \subseteq V(G)$, if every path from a vertex in a to a vertex in b contains a vertex from c.

We say that two vertices in a graph are adjacent or neighbours, if there is an edge between them, and we define the boundary of a subset a of Δ, written ∂a, as those vertices that are not in a but are adjacent to some vertex in a. A set a is called complete, if all possible edges between the vertices of a are present in the graph. If a is complete and a is not a subset of another complete subset of the graph, then a is called a clique. A vertex v is perfect, if the set ∂v of neighbours of v is complete. The subgraph induced by a subset $A \subseteq V(G)$ of the vertices of a graph $G = (V(G), E(G))$ is the graph $G_A = (A, E_A)$, where $E_A \subseteq E(G)$ are the edges $\{v, w\} \in E(G)$ with both $v \in A$ and $w \in A$. The connected components of a graph are a partitioning of the graph into subgraphs such that two vertices are in the same connected component, if and only if there is a path between the vertices. If a graph only contains one connected component, then the graph is connected.

Definition 1 Two subsets a and b of $V(G)$ form a decomposition of a graph $G = (V(G), E(G))$, if $a \cup b = V(G)$, $a \setminus b \neq \emptyset$, $b \setminus a \neq \emptyset$, $a \setminus b$ and $b \setminus a$ are separated by $a \cap b$ in the graph, and $a \cap b$ is a complete subset.

In words, for the 2-section graph of a graphical model, two subgraphs of the 2-section graph form a decomposition of the graph with respect to a subset c of the vertices of the graph, if the graph is the union of two subgraphs and the intersection

c between the two subgraphs is complete. The set c is a clique separator. The graph is decomposed into the two subgraphs. The subgraphs may be further decomposed into subgraphs.

Deønition 2 If a graph and its subgraphs can be decomposed recursively until all the subgraphs are complete, then the graph is decomposable.

Note that a graph may be decomposed without being decomposable. We say that the graph is reducible, if it can be decomposed, i.e., its vertex set contains a clique separator, otherwise the graph is said to be irreducible, a prime or a non-separable atom. A subgraph G_A of a graph G is an irreducible component or a maximal prime subgraph of G, if G_A is irreducible and G_B is reducible for all B with $A \subset B \subseteq V(G)$.

A graph is triangulated, if it contains no cycles of length greater than 3 without a chord. It is a well-known fact (Lauritzen et al. 1984) that the decomposable graphs are the triangulated graphs, the chordal graphs or rigid circuit graphs.

Hypergraphs

Restating, a generating class is a set of subsets of a ønite set so no element in the generating class is a subset of another element.

The generating class $\mathcal{M}$ of a model may be viewed as the edges of a generating class hypergraphs, in this chapter called hypergraphs: a graph $\mathcal{M} = (\mathrm{V}(\mathcal{M}), \mathcal{M})$ with vertices $V(\mathcal{M}) \subseteq \Delta$ the variables of the model and edges $\mathcal{M}$ the maximal permissible interaction terms.

The edges $\mathcal{M}$ are a generating class and are not only subsets of cardinality 2 of the variables, but subsets of any size of the set of vertices.

The 2-section graph of a hypergraph $\mathcal{M} = (\mathrm{V}(\mathcal{M}), \mathcal{M})$ is a simple undirected graph $G^{\mathcal{M}} = (V(G^{\mathcal{M}}), E(G^{\mathcal{M}}))$ with vertices $V(G^{\mathcal{M}}) = V(\mathcal{M})$ and edges $E(G^{\mathcal{M}}) = \{\{v, w\}| \; \exists c \in \mathcal{M} : \{\mathrm{v}, \mathrm{w}\} \subseteq \mathrm{c}\}$. To distinguish the edges of the hypergraph from the edges of the 2-section graph, we call the edges of the hypergraph, i.e., the elements of the generating class, generators.

A hypergraph is conformal, if the cliques of it's 2-section graph are the edges of the hypergraph. Thus a model $\mathcal{M}$ is graphical, if the hypergraph with generators, i.e., edges, $\mathcal{M}$ is conformal.

Deønition 3 Two subsets a and b form a decomposition of $V(\mathcal{M})$ relative to a hypergraph $\mathcal{M} = (\mathrm{V}(\mathcal{M}), \mathcal{M})$, if $a \cup b = V(\mathcal{M})$, $a \setminus b \neq \emptyset$, $b \setminus a \neq \emptyset$, a and b are separated by $a \cap b$ in the 2-section graph of $\mathcal{M}$, and $a \cap b \subseteq c$ for some generator of $\mathcal{M}$.

If a hypergraph is conformal and further the 2-section graph of the hypergraph is decomposable, then the hypergraph is acyclic. For a hierarchical model to be decomposable, the hypergraph of the model has to be acyclic, i.e., the hypergraph has to be conformal and the 2-section graph of the model has to be decomposable. (A hypergraph need not be a generating class hypergraph to be an acyclic hypergraph, e.i., an acyclic hypergraph may contain edges that are subsets of other edges.)

The terms reducible, irreducible (prime) and irreducible component (maximal prime subgraph) are for hypergraphs deøned analogously to graphs.

The Fill-In Graph

An elimination ordering π is an ordering $\{v_1, \cdots, v_n\}$ of the vertices in a graph. The øll-in F_π caused by the ordering π is the set of edges deøned as follows:

$$F_\pi = \{\{v, w\} |\ v \neq w,\ \text{there is no edge between } v \text{ and } w,\ \text{and there is a path } v = v_{(1)}, v_{(2)}, \cdots, v_{(k)} = w \text{ such that } \pi(v_{(i)}) < \min\{\pi(v), \pi(w)\} \text{ for } i = 2, \cdots, k-1\}. \tag{2.2}$$

An elimination ordering π is perfect, if the øll-in caused by the ordering is empty, minimum, if $|F_\pi|$ is minimum over all possible orderings, and minimal, if there is no ordering σ such that $F_\sigma \subset F_\pi$. It is a well known fact, see e.g. Rose et al. (1976), that a graph has a perfect vertex ordering if and only if the graph is decomposable, and that an ordering $\{v_1, \cdots, v_n\}$ of the vertices $v_{(i)}$, $i = 1, \cdots, n$, in a graph is perfect if $\partial v_i \cap \{v_i, \cdots, v_n\}$ is complete for each vertex v_i, $i = 1, \cdots, n$, in the graph, i.e., if and only if v_i is perfect in the subgraph induced by $\{v_i, \cdots, v_n\}$. The øll-in graph of a graph $G = (V(G), E(G))$ for an ordering π is the graph $F = (V(G), E(G) \cup F_\pi)$ with the øll-in F_π added. The boundary $\partial_\pi v$ are the vertices adjacent to v in the øll-in graph.

RIP orderings

If a system $\zeta = \{R_1, \cdots, R_J\}$ of J, $J \geq 1$, subsets $R_j \subseteq \Delta$ are ordered in a sequence $(R_1, \cdots, R_J)$ such that

$$\forall\ j = 2, \cdots, J\ \ \exists\ k, 1 \leq k < j :\ \ (R_j \cap \bigcup_{l=1}^{j-1} R_l) \subseteq R_k, \tag{2.3}$$

then the sequence $(R_1, \cdots, R_J)$ fuløls the running intersection property.
I.e., if R_j, $j = 1, \cdots, J$, are the cliques of a graph, then the sequence $(R_1, \cdots, R_J)$ fuløls the running intersection property, if the cliques in the sequence are ordered such that for each clique the intersection between the clique and the union of previous cliques is a subset of a previous clique. A graph is decomposable, if and only if such an ordering of the cliques exists.

Junction trees

Let $\mathcal{M}$ be a ønite collection of subsets of a set Δ, e.g., a generating class. The junction graph $J(\mathcal{M}) = (\mathcal{M}, \mathrm{E(J)})$ for $\mathcal{M}$ is a graph with vertices $V(J) = \mathcal{M}$ the elements of $\mathcal{M}$ and edges $E(J) = \{\{c_i, c_j\} \subseteq \mathcal{M} |\ \mathrm{c_i \cap c_j} \neq \emptyset\}$, i.e., there is an edge between two nodes of the junction graph, if the intersection of the two vertex sets of the two nodes is not empty.

A spanning tree for $J(\mathcal{M})$ is a junction tree for $J(\mathcal{M})$, if for any pair $c_i, c_j \in \mathcal{M}$ all vertices on the path between c_i and c_j contain $c_i \cap c_j$, see also Jensen (1988). If $J(\mathcal{M})$ is not connected, then the graph $J(\mathcal{M})$ does not have a junction tree, but each connected component of $J(\mathcal{M})$ may have. A connected component of a graph has a junction tree, if and only if the connected component is decomposable, (Jensen 1988).

A junction tree can be constructed from a sequence $(R_1, \cdots, R_J)$ fulølling the running intersection property: the tree is given vertices $(R_1, \cdots, R_J)$, and for $j = 2, \cdots, J$ the vertex R_j is connected to one of the vertices R_k fulølling (2.3). Also a sequence fulølling the running intersection property can be read oœ a junction tree: Pick any vertex as root, and then by either Breadth-ørst search or Depth-ørst search visit all vertices of the junction tree letting R_i be the elements of the i-th visited node of the tree.

Through this chapter m will denote the number of generators in the generating class $\mathcal{M}$, m' will denote the total size of the generating class, i.e., the sum $m' = \sum_{i=1,\cdots,m} |c_i|$, where $\mathcal{M} = \{\mathrm{c_i}, \mathrm{i} = 1, \cdots, \mathrm{m}\}$, n the number of vertices in the 2-section graph $G^{\mathcal{M}} = (V(G^{\mathcal{M}}), E(G^{\mathcal{M}}))$ for the hypergraph $\mathcal{M} = (\mathrm{V}(\mathcal{M}), \mathcal{M})$, e the number of edges in the 2-section graph, and e' will denote the number of edges in a øll-in graph of the 2-section graph. w will denote the number of decompositions with respect to minimal separators. q will denote the number of cycles in the IPS-algorithm, $|\mathcal{I}|$ will denote the size of the full table, and s will denote the size of the state space, see expression 2.18.

Collapsibility and Decompositions of Models

Deønition 4 $\mathcal{M}$ is collapsible onto a, if one of the two following equivalent properties hold:

i) for all $p = p(i) \in \mathcal{M}$, we have that $p(i_a) \in \mathcal{M}_{\mathrm{a}}$,

ii) for all $i_a \in \mathcal{I}_{\mathrm{a}}$, $\hat{p}(i_a) = \hat{p}_a(i_a)$.

For proof of equivalence, see Asmussen & Edwards (1983). Note that $\hat{p}(i_a)$ is the marginalized maximum likelihood estimate whereas $\hat{p}_a(i_a)$ is the maximum likelihood estimate in the restricted model $\mathcal{M}_{\mathrm{a}}$.

Theorem 2.3 of Asmussen & Edwards (1983) states that a hierarchical model $\mathcal{M}$ is collapsible onto a, if and only if the boundary of every connected component of a^c is contained in a generator of $\mathcal{M}$.

Theorem 1 If a and b form a decomposition of Δ relative to a hierarchical log-linear model $\mathcal{M}$, then

$$\hat{p}_\Delta(i) = \frac{\hat{p}_a(i_a)\hat{p}_b(i_b)}{N(i_{a\cap b})/N}, \quad \text{where } \frac{0 \cdot 0}{0} = 0 \tag{2.4}$$

and

$$\hat{p}_\Delta(i_a) = \hat{p}_a(i_a). \tag{2.5}$$

Proof This is Theorem 2.1 of Asmussen & Edwards (1983). Proof of the ørst formula is given by Haberman (1974), Andersen (1974), and Lauritzen (1982), and for the graphical situation in Darroch et al. (1980). Partitioning of the maximum-likelihood estimates was introduced by Goodman (1971). See also Sundberg (1975). If the denominator $N(i_{a\cap b})$ is equal to zero, then also both numerators are zero. The second formula follows by summing over $b \setminus a$ and noting that $\hat{p}_b(i_{a\cap b}) = N(i_{a\cap b})/N$ since $a \cap b$ is contained in a generator of $\mathcal{M}_{\mathrm{b}}$. □

Deønition 5 $\mathcal{M}$ is decomposable with respect to the system $\zeta = \{R_1, \cdots, R_J\}$ of J, $J \geq 1$, subsets $R_j \subseteq \Delta$, if there exists an ordering $(R_{\sigma(1)}, \cdots, R_{\sigma(J)})$ of the J subsets such that

$$p(i) = p(i_{R_{\sigma(1)}}) \frac{p(i_{R_{\sigma(2)}})}{p(i_{R_{\sigma(2)} \cap R_{\sigma(1)}})} \cdots \frac{p(i_{R_{\sigma(J)}})}{p(i_{R_{\sigma(J)} \cap (R_{\sigma(1)} \cup \cdots \cup R_{\sigma(J-1)})})} \text{ for all } p \in \mathcal{M}. \tag{2.6}$$

Formulas for Maximum Likelihood Estimates

If the model is decomposable and the ordering $\{v_1, \cdots, v_n\}$ of the vertices in the graph is perfect, then repeated use of the above formula gives the following closed form expression of the maximum likelihood estimates for the cell-probabilities

$$\hat{p}_\Delta(i) = c \prod_{k=1..n} \frac{N(i_{\{v_k\} \cup (\partial v_k \cap \{v_k, \cdots, v_n\})})}{N(i_{\partial v_k \cap \{v_k, \cdots, v_n\}})}, \tag{2.7}$$

which can be reduced to

$$\hat{p}_\Delta(i) = c \prod_{k=1..u} N(i_{a_k})^{\nu(a_k)}, \tag{2.8}$$

where a_k are cliques or intersections of cliques of the 2-section graph of the model and the index ν is also known as the adjusted replication number (Haberman 1974, Darroch et al. 1980). Thus the maximum-likelihood estimates are simple to calculate from marginal tables when the index is known for each subset of the vertices of the graph associated with the model. The algorithms of chapter 3 calculates for decomposable models precisely this index. The constant c is equal to $1/|\mathcal{I}_{\mathrm{A^C}}|$, where A is the union of variables in the model.

If the model can be decomposed, but the model is not decomposable, then the maximum likelihood estimate for the cell-probabilities can be expressed as

$$\hat{p}_\Delta(i) = c \prod_{k=1..u} N(i_{a_k})^{\nu(a_k)} \prod_{l=1..v} \hat{p}_{\mathcal{B}_\mathrm{l}}(i_{b_l}), \tag{2.9}$$

where $\mathcal{B}_\mathrm{l}$ are the generating classes for irreducible components that cannot be decomposed further. The sets a_k and b_l are vertex-sets for the subgraphs, into which the graph is decomposed, or for some of the a_k's, separators. The maximum likelihood estimates $\hat{p}_{\mathcal{B}_\mathrm{l}}$ for the cell-probabilities on the irreducible components b_l are found by the IPS-algorithm, the iterative proportional scaling algorithm, here $\mathcal{M} = \mathcal{B}_\mathrm{l}$ and $\mathcal{I} = \mathcal{I}_{\mathrm{b_l}}$ for simplicity:

i) Set the initial values,

$$\hat{p}^{(0)}(i) = \frac{1}{|\mathcal{I}|}, \quad \text{for all} \quad i \in \mathcal{I}; \tag{2.10}$$

ii) For each generator c in $\mathcal{M}$,

$$\hat{p}^{(r+1)}(i) = \hat{p}^{(r)}(i)\frac{N(i_c)/N}{\hat{p}^{(r)}(i_c)} \quad \text{for all} \quad i \in \mathcal{I}, \quad \text{where} \quad 0\frac{0}{0} = 0. \tag{2.11}$$

Repeat step ii) until convergence. Note that all cells in the table $\mathcal{I}$ have to be visited for each generator c in $\mathcal{M}$ for each cycle of step ii). Thus, if q cycles of ii) are necessary for convergence, the complexity of the IPS-algorithm is of the order $O(qmn|\mathcal{I}|)$. The factor n is because the cells of the tables cannot be addressed in constant time. If all variables are binary, and there are n of them, then the complexity is $O(qmn2^n)$, i.e., the complexity is exponential in the dimension n in the version here described. See, however, section 2.4 for an eŒcient implementation

The adjustment need not be with marginal tables of observed counts, but can be with any consistent system of marginal distributions, see Malvestuto (1989).

The IPS-algorithm is also known as the iterative proportional øtting or the Deming-Stephan algorithm (Deming & Stephan 1940, Fienberg 1970, Darroch & Ratcliœ 1972). An implementation of the IPS-algorithm is given in Haberman (1972). Alternatively, e.g., the Newton-Raphson method could be applied. The Newton-Raphson method has the advantage over the IPS-algorithm, which is very simple to implement, of needing very few iterations. In some situations the IPS-algorithm after some initial iteration will converge slowly to the maximum likelihood estimates. But the Newton-Raphson method needs to compute the matrix of second derivatives of the log-likelihood and the inverse of this matrix. For a higher dimensional contingency table with many parameters, the calculation of the inverse matrices, whose orders may be up to thousands, aœects the eŒciency of the Newton-Raphson method. Because the IPS-algorithm is insensitive to the initial values, and in a few iterations gives an estimate fairly close to the maximum likelihood estimates, and because the Newton-Raphson method with good initial values converges fast, one could consider a øtting procedure which combines the IPS-algorithm and the Newton-Raphson method for small irreducible components (without empty cells). The Newton-Raphson method can not handle tables with empty cells.

The eŒciency of both the IPS-algorithm and the Newton-Raphson method will be improved by reducing the use of the iterative algorithm to the irreducible components.

2.3 Identiøcation and Decomposition of the Model

By the algorithms presented in Tarjan & Yannakakis (1984) and Tarjan (1985) hierarchical log-linear models can be identiøed and decomposed into irreducible components in $O(ne + nm\min(n, m))$ time, see chapter 3 for this algorithm. If the model class is restricted to graphical models, then decomposable models can be identiøed and decomposed in $O(e + nm\log m)$ with $m \leq n$. Graphical models can often be handled in $O(ne + nm + nw\log w)$ with $w < \min(n, m)$. Working directly on the hypergraph decomposable models can be decomposed and the index of the graph computed in $O(n + m' + nw\log w)$ time by the algorithm of Tarjan & Yannakakis (1984).

2.4 Partitioning of the Irreducible Components

The space saving modiøcation of the IPS-algorithm

In Jirouek (1991) a space saving modiøcation of the IPS-algorithm is given. Consider any hierarchical model $\mathcal{M}$. Then Theorem 1 of Jirouek (1991) say that the maximum likelihood estimate of p under the model $\mathcal{M}$ is decomposable with respect to any system $\{R_1, \cdots, R_J\}$ fulølling the following two conditions:

(1) there exists a permutation σ such that $(R_{\sigma(1)}, \cdots, R_{\sigma(J)})$ meets the running intersection property,

(2) for every $c \in \mathcal{M}$ there exists $j \in \{1, \cdots, J\}$ such that $c \subseteq R_j$.

Note that a model need not to be decomposable to be decomposable with respect to the system $\{R_1, \cdots, R_J\}$.

E.g., consider for n variables a model consisting of a cycle, i.e., consider the model $[[1,2][2,3]\cdots[n-1,n][n,1]]$. If all edges from 1 to all other variables not adjacent to 1 are added, then we get the system $\{\{1,2,3\},\{1,3,4\},\{1,4,5\},\cdots\{1,n-1,n\}\}$. The edges added such that the resulting system can be permuted to meet the running intersection property is a øll-in. If all variables are binary, then the un-decomposed maximum likelihood estimate requires 2^n reals to store. Applying the decomposition of the maximum likelihood estimate the estimate only requires $(n-2)2^3$ reals to store.

Together with Lemma 2 and Theorem 2 of Jirouek (1991), which ensures that the probabilities of each step of the following algorithm also are decomposable with respect to the system $\{R_1, \cdots, R_J\}$, the above theorem gives the following modiøed IPS-algorithm. $\mathcal{M}$ is a hierarchical model and $\{R_1, \cdots, R_J\}$ is the cliques of a decomposable graph such that for every $c_i \in \mathcal{M}$, $i = 1, \cdots, m$, there exists a $j \in \{1, \cdots, J\}$ such that $c_i \subseteq R_j$. For eŒciency, $\mathcal{M}$ is a irreducible component, i.e., the model contains no variables only in one generator and the model cannot be decomposed:

i) Set the initial values,

$$\hat{p}_j^{(0)}(i_{R_j}) = \frac{1}{|\mathcal{I}_{\mathrm{R_j}}|}, \quad \text{for all} \quad i_{R_j} \in \mathcal{I}_{\mathrm{R_j}}; \quad \text{for all} \quad \mathrm{j} = 1, \cdots, \mathrm{J}. \tag{2.12}$$

ii) In the r-th cycle, for each generator $c_i \in \mathcal{M}$, $i = 1, \cdots, m$,

  a) Choose any permutation $\sigma_i(1), \cdots, \sigma_i(J)$ such that $(R_{\sigma_i(1)}, \cdots, R_{\sigma_i(J)})$ meets the running intersection property and such that $c_i \subseteq R_{\sigma_i(1)}$.

  b) Adjust the part $R_{\sigma_i(1)}$ with the margin c_i:

$$\hat{p}_{\sigma_i(1)}^{(r+1)}(i_{R_{\sigma_i(1)}}) = \hat{p}_{\sigma_i(1)}^{(r)}(i_{R_{\sigma_i(1)}}) \frac{N(i_{c_i})/N}{\hat{p}_{\sigma_i(1)}^{(r)}(i_{c_i})} \quad \text{for all} \quad i_{R_{\sigma_i(1)}} \in \mathcal{I}_{\mathrm{R}_{\sigma_i(1)}}. \tag{2.13}$$

  The marginal $\hat{p}_{\sigma_i(1)}^{(r)}(i_{R_{\sigma_i(1)}})$ is before the adjustment further marginalized to get $\hat{p}_{\sigma_i(1)}^{(r)}(i_{c_i})$ within an auxiliary store.

c) Distribute the adjustment:
For $j = 2, \cdots, J$: Let $d_{i,j}$ be the intersection $R_{\sigma_i(j)} \cap (R_{\sigma_i(1)} \cup \cdots \cup R_{\sigma_i(j-1)})$ between the part $R_{\sigma_i(j)}$ to update and the parts $(R_{\sigma_i(1)} \cup \cdots \cup R_{\sigma_i(j-1)})$ already updated in this cycle, i.e., this adjustment with the marginal table $N(i_{c_i})$.

1) The marginal $\hat{p}^{(r)}_{\sigma_i(j)}(i_{R_{\sigma_i(j)}})$ is further marginalized to get the marginal table $\hat{p}^{(r)}_{\sigma_i(j)}(i_{d_{i,j}})$.
2) The marginal $\hat{p}^{(r+1)}_{\sigma_i(k_{i,j})}(i_{R_{\sigma_i(k_{i,j})}})$ is further marginalized to get the marginal table $\hat{p}^{(r+1)}_{\sigma_i(k_{i,j})}(i_{d_{i,j}})$, where $k_{i,j}$ is the one of the indices whose existence is guaranteed by the running interception property:

$$1 \leq k_{i,j} < j \quad \text{and} \quad d_{i,j} \subseteq R_{\sigma_i(k_{i,j})}. \tag{2.14}$$

3) Compute

$$\hat{p}^{(r+1)}_{\sigma_i(j)}(i_{R_{\sigma_i(j)}}) = \hat{p}^{(r)}_{\sigma_i(j)}(i_{R_{\sigma_i(j)}}) \frac{\hat{p}^{(r+1)}_{\sigma_i(k_{i,j})}(i_{d_{i,j}})}{\hat{p}^{(r)}_{\sigma_i(j)}(i_{d_{i,j}})} \text{ for all } i_{R_{\sigma_i(j)}} \in \mathcal{I}_{\mathrm{R}_{\sigma_i(j)}}. \tag{2.15}$$

Repeat step ii) until convergence.

The adjustment need not be with marginal tables of observed counts but can be with any marginal distributions, see Jirouek (1991).

The permutation

Assume ørst that the øll-in is given.

For each marginal c_i to adjust we in step ii) need a permutation $\sigma_i(1), \cdots, \sigma_i(J)$ of the indices $1, \cdots, L$ such that the sequence $(R_{\sigma_i(1)}, \cdots, R_{\sigma_i(J)})$ meets the running intersection property and such that $c_i \subseteq R_{\sigma_i(1)}$. These $m = |\mathcal{M}|$ permutations can of course be found once and for all and stored together the $k_{i,j}$'s of step 2) for each margin before starting the iterations. The permutations can be found with the help of the restricted maximum cardinality search algorithm described in Tarjan & Yannakakis (1984) by ørst numbering the vertices of c_i, or junction trees can be used. About junction trees, see also Jensen (1988), Lauritzen & Spiegelhalter (1988) or Jensen et al. (1990).

The algorithm restricted maximum cardinality search runs in $O(m')$, and thus the m permutations can by this algorithm be found in $O(mm')$.

Choosing an optimal permutation given the øll-in

Given the øll-in graph with cliques $\{R_1, \cdots, R_J\}$ and the permutation $\sigma_i(1), \cdots, \sigma_i(J)$ such that $(R_{\sigma_i(1)}, \cdots, R_{\sigma_i(J)})$ meets the running intersection property and such that $c_i \subseteq R_{\sigma_i(1)}$, the sets $d_{i,j}$ of step c) are uniquely determined. We also observed that the complexity of computing the marginal table $\hat{p}^{(r)}_{\sigma_i(j)}(i_{d_{i,j}})$ and the adjustment of step 3) is independent of the size of $\mathcal{I}_{\mathrm{d}_{i,j}}$, when the marginal table $\hat{p}^{(r+1)}_{\sigma_i(k_{i,j})}(i_{d_{i,j}})$ is found. But the time used in step 2) to marginalize the marginal table $\hat{p}^{(r+1)}_{\sigma_i(k_{i,j})}(i_{R_{\sigma_i(k_{i,j})}})$ to get $\hat{p}^{(r+1)}_{\sigma_i(k_{i,j})}(i_{d_{i,j}})$ depends on the size of $\mathcal{I}_{\mathrm{R}_{\sigma_i(k_{i,j})}}$, and the $k_{i,j}$ of this step is not uniquely determined. So we want to choose $k_{i,j}$ such that $|\mathcal{I}_{\mathrm{R}_{\sigma_i(k_{i,j})}}|$ is as small as possible. By a trivial algorithm these $k_{i,j}$'s for all permutations and all intersections $d_{i,j}$ can be found in $O(nm^3)$ given the permutations.

If the choice of $k_{i,j}$ is ambiguous, then the smallest marginal table containing the factors $d_{i,j}$ already has been found in the current adjustment with a margin: Assume $d_{i,j} \subseteq R_{\sigma_i(s)}$ and $d_{i,j} \subseteq R_{\sigma_i(t)}$ with $s < t < j$. Then $D = R_{\sigma_i(s)} \cap R_{\sigma_i(t)} \neq \emptyset$ and $d_{i,j} \subseteq D$, and the marginal table determined by D was found in the t-th step of this adjustment.

We also observe, that among permutations $\sigma_i(1), \cdots, \sigma_i(J)$ such that the sequence $(R_{\sigma_i(1)}, \cdots, R_{\sigma_i(J)})$ meets the running intersection property and such that $c_i \subseteq R_{\sigma_i(1)}$ we should choose the one such that if $d_{i,t} \subseteq d_{i,s}$, then $s < t$.

We wish to ønd the permutation $\sigma_i(1), \cdots, \sigma_i(J)$ such that $\sum_{j=2,\cdots,J} |\mathcal{I}_{\mathrm{R}_{\sigma_i(k_{i,j})}}|$ is as small as possible. The computations of step c) are analogously to that of distributing evidence in expert systems based on causal probabilistic networks, see Lauritzen & Spiegelhalter (1988) or Jensen et al. (1990). An algorithm to construct a junction tree such that the sum of a ısymmetric measureȷ on the separators, i.e, the intersections $d_{i,j}$, e.g., the sum $\sum_{j=2,\cdots,J} |\mathcal{I}_{\mathrm{d}_{i,j}}|$, is minimized, is presented in Jensen & Jensen (1994). But since the weights of the links in our situation are not symmetric, the permutations giving the the minimal sums $\sum_{j=2,\cdots,J} |\mathcal{I}_{\mathrm{R}_{\sigma_i(k_{i,j})}}|$ cannot be read oœ a junction tree constructed by the algorithm of Jensen & Jensen (1994). We need to construct a maximal spanning tree for the directed (junction) graph with weights on directed edges, i.e., the graph achieved by replacing each edge of the junction graph with two opposite directed edges from and to the vertices of the edge to be replaced.

Workspace and Complexity

It is clear that in each adjustment of step ii) we have to visit all the cells of the tables of probabilities at least once. Thus we start the analysis of the complexity by computing the size of the required workspace.

Our tables $N(i_{c_i})$, $i_{c_i} \in \mathcal{I}_{\mathrm{c_i}}$, $i = 1, \cdots, m$, to which the margins should be adjusted have a total size of

$$s_{\mathcal{M}} = \sum_{c_i \in \mathcal{M}} |\mathcal{I}_{\mathrm{c_i}}|. \tag{2.16}$$

The size s_R of the state space, the tables $\hat{p}_j(i_{R_j})$, $i_{R_j} \in \mathcal{I}_{\mathrm{R_j}}$, $j = 1, \cdots, J$, is

$$s_R = \sum_{j=1,\cdots,J} |\mathcal{I}_{\mathrm{R_j}}|. \tag{2.17}$$

Thus the total size s of workspace is then given by

$$s_W = s_{\mathcal{M}} + s_R + s_A = \sum_{c_i \in \mathcal{M}} |\mathcal{I}_{\mathrm{c_i}}| + \sum_{\mathrm{j=1,\cdots,J}} |\mathcal{I}_{\mathrm{R_j}}| + 2 \max_{\mathrm{j=1,\cdots,J}} |\mathcal{I}_{\mathrm{R_j}}|, \tag{2.18}$$

where s_A is some auxiliary store of the order $2 \cdot \max_{j=1,\cdots,J} |\mathcal{I}_{\mathrm{R_j}}|$ needed in step b), step 1) and step 2).

Thus given the øll-in, the complexity of the algorithm is then of the order $O(qms)$, where q is the number of cycles needed for convergence, m is the number of generators in the generating class and $s = s_R$ is the size of the state space. For a detailed comparison of the complexity of the modiøed IPS-algorithm with the complexity of the ordinary IPS-algorithm, see Jirouek (1994).

The Fill-in

The crucial point for the complexity of the algorithm is thus to determine a øll-in such that state space is as small as possible. Having implemented the algorithm Maximum Cardinality Search of Tarjan & Yannakakis (1984) to decompose graphical models, it is tempting to use the vertex ordering produced by this algorithm and then ønd the øll-in by the algorithm Fill-In Computation of the same paper. But since Maximum Cardinality Search is designed to test for a zero øll-in fast, and not to ønd a vertex ordering giving a small state space, this is not recommended. The øll-in produced this way is not even minimal, i.e., a subset of the produced øll-in may also be a øll-in giving a decomposable graph. Finding the optimal øll-in, i.e., a øll-in giving the smallest state space, is NP-hard (Wen 1990). The following simple greedy algorithms will often give a smaller state space than Maximum Cardinality Search: Choose a vertex ordering $\{v_1, \cdots, v_n\}$, such that the i-the vertex v_i is selected, such that either

- deøciency: the number of øll-in edges resulting from picking v_i at this step, i.e., the number of edges missing to complete the subgraph induced by $\partial v_i \cap \{v_i, \cdots, v_n\}$ in the graph with the øll-in edges resulting of previously ordered vertices added,
- degree: the number $|\partial v_i \cap \{v_i, \cdots, v_n\}|$ of vertices adjacent to v_i and not yet ordered, or
- weight: the size $|\mathcal{I}_{\{\mathrm{v}\} \cup \partial \mathrm{v_i} \cap \{\mathrm{v_i}, \cdots, \mathrm{v_n}\}}|$ of the state space of v_i and the vertices adjacent to and following v_i

is minimized among the vertices not yet ordered, and then ønd the øll-in corresponding to that vertex elimination ordering. The algorithms based upon the last tree heuristic rules will be called the minimum deøciency heuristic, minimum degree heuristic and minimum weight heuristic, respectively. Minimum deøciency heuristic has the advantage to minimum degree heuristic of producing a perfect ordering, if such an ordering exists, but minimum degree heuristic is faster, Rose (1973). In Kjrulœ (1990) minimum weight heuristic is found to be superior to the minimum deøciency heuristic and to the minimum degree heuristic of Rose (1973), (Kjrulœ 1992). In Kjrulœ (1992) algorithms for ønding a øll-in given a small state space based on simulated annealing are discussed. These algorithms are better, but very slow.

The result of the algorithm

The output of the above algorithm will be a set of distributions $\hat{p}_{R_j}^{(\infty)}(i_{R_j})$, $j = 1, \cdots, J$ such that the maximum likelihood estimate can be determined by:

$$\hat{p}_\Delta(i) = \hat{p}_{R_1}^{(\infty)}(i_{R_1}) \frac{\hat{p}_{R_2}^{(\infty)}(i_{R_2})}{\hat{p}_{R_2}^{(\infty)}(i_{R_2 \cap R_1})} \cdots \frac{\hat{p}_{R_J}^{(\infty)}(i_{R_J})}{\hat{p}_{R_J}^{(\infty)}(i_{R_J \cap (R_1 \cup \cdots \cup R_{J-1})})}, \tag{2.19}$$

when the cliques R_j, $j = 1, \cdots, J$, are ordered such that $(R_1, \cdots, R_J)$ meets the running intersection property.

If the ıintersecting distributionȷ is divided into the denominator:

$$\psi_j(i_{R_j}) = \frac{\hat{p}_{R_j}^{(\infty)}(i_{R_j})}{\hat{p}_{R_j}^{(\infty)}(i_{R_j \cap (R_1 \cup \cdots \cup R_{j-1})})}, \quad \text{for all} \quad i_{R_{\sigma_i(j)}} \in \mathcal{I}_{\mathrm{R}_{\sigma_i(j)}}, \quad \text{for all} \quad \mathrm{j} = 2, \cdots, \mathrm{J}, \tag{2.20}$$

then the maximum likelihood estimate can be computed as

$$\hat{p}_\Delta(i) = \hat{p}_{R_1}^{(\infty)}(i_{R_1}) \psi_2(i_{R_2}) \cdots \psi_J(i_{R_J}). \tag{2.21}$$

For the huge models, the algorithm of Jirouek (1991) makes the diœerence of being able to handle these models or not. Although among all hierarchical models there may only be a few that can be partitioned further by the algorithm of Jirouek (1991), as decomposable models are rare among hierarchical models, the algorithm extends considerably the class of models, that can be handled.

2.5 Computation of Tables of Marginal Counts

We need to compute the tables $N(i_a)$ for some subsets a of Δ to ønd the maximum likelihood estimates of the cell probabilities and to ønd the deviance. Assume ørst that the observations are given in the form of the saturated table $N(i)$, $i \in \mathcal{I}$. To compute the table $N(i_a)$, $i_a \in \mathcal{I}_{\mathrm{a}}$, we ørst have to make all cells of the marginal table zero, and then visit all cells of $N(i)$, $i \in \mathcal{I}$, for adding up. To be precise, for each count to add in the marginal table, we cannot identify the cell in the marginal table in constant time, but computing the index of the table takes time of the order $O(n)$. Thus the complexity of computing a marginal table $N(i_a)$, $i_a \in \mathcal{I}_{\mathrm{a}}$, is of the order $O(n|\mathcal{I}|)$, or, if all variables are binary $O(n2^n)$. If $a \subset b$ and the marginal table $N(i_b)$ is already computed, then the table $N(i_a)$ of course can be found by adding the counts of the b table, i.e., in $O(n|\mathcal{I}_{\mathrm{b}}|)$. Since the separators are subsets of the vertex-sets of the components, this will be the case for about half of the marginal tables to ønd to compute the deviance of decomposable models.

Now assume that the observed counts are only available as a case list, but that we can still store the marginal table $N(i_a)$, $i_a \in \mathcal{I}_{\mathrm{a}}$. We have to clear the table, which takes $O(|\mathcal{I}_{\mathrm{a}}|)$. Then all cases have to be visited for adding them up in the marginal table. For each case we have to identify the cell to which the case belongs. Thus we get the complexity $O(n(|\mathcal{I}_{\mathrm{a}}| + \mathrm{N}))$.

Finally, assume that neither the marginal or the saturated table can be stored. We are only interested in the counts in cells with non-zero count. These counts can be found by sorting the cases according to the vectors I_a, i.e., the values of the variables a for each case, and then for each conøguration found of the variables a in the case list count the number of cases with that conøguration. If the case list is not too long, i.e., for the time being it may be stored in a few Mbytes, then this sorting can be performed internally in a computer by a sorting algorithm as Quick-Sort or Heap-Sort in $O(nN\log N)$ or Shell-Sort in $O(nN^{1.5})$. For huge datasets an external sorting procedure, as, e.g., Merge-Sort in $O(nN\log N)$, is necessary.

The complexity of computing the marginal table $N(i_a)$, $i_a \in \mathcal{I}_\mathrm{a}$ (for non-zero cells) can then be expressed by:

$$\begin{cases} O(n|\mathcal{I}|) & \text{if } |\mathcal{I}| \le \mathrm{N}, \\ O(n(|\mathcal{I}_\mathrm{a}| + \mathrm{N})) & \text{if } |\mathcal{I}| > \mathrm{N} \text{ and } |\mathcal{I}_\mathrm{a}| \le \mathrm{N}\log\mathrm{N}, \\ O(nN\log N) & \text{if } |\mathcal{I}_\mathrm{a}| > \mathrm{N}\log\mathrm{N}. \end{cases}$$

2.6 Computation of the Deviance

Since the maximum likelihood estimate can be expressed as

$$\hat{p}_\Delta(i) = c \prod_{k=1..u} N(i_{a_k})^{\nu(a_k)} \prod_{l=1..v} \hat{p}_{\mathcal{B}_\mathrm{l}}(i_{b_l}) \tag{2.22}$$

the deviance can be computed as:

$$\begin{aligned} \log L &= \sum_{i\in I} \log(p_\Delta(i)^{N(i)}) \\ &= \sum_{i\in I} N(i) \log\left(c \prod_{k=1..u} N(i_{a_k})^{\nu(a_k)} \prod_{l=1..v} \hat{p}_{\mathcal{B}_\mathrm{l}}(i_{b_l}) \right) \\ &= N\log(c) \\ &\quad + \sum_{k=1..u} \nu(a_k) \sum_{i_{a_k}\in I_{a_k}} N(i_{a_k}) \log(N(i_{a_k})) \\ &\quad + \sum_{l=1..v} \sum_{i_{b_l}\in I_{b_l}} N(i_{b_l}) \log(\hat{p}_{\mathcal{B}_\mathrm{l}}(i_{b_l})). \end{aligned} \tag{2.23}$$

Hence we only need to sum over non-zero cells in suŒcient marginal tables for the decomposable models.

The formula for the maximum likelihood estimates, i.e., the vertex-sets a_k with indices $\nu(a_k)$ and the vertex-sets b_l of the non-decomposable and irreducible components with generating classes $\mathcal{B}_\mathrm{l}$ (2.22) are found in $O(ne+n^2+nm\min(n,m))$ by the algorithms of chapter 3.

Assume ørst that the model is decomposable. Then the deviance can be computed as

$$\log L = N\log(c) + \sum_{k=1..u} \left(\nu(a_k) \sum_{i_{a_k}\in I_{a_k}} N(i_{a_k}) \log(N(i_{a_k})) \right). \tag{2.24}$$

The terms $\sum_{i_{a_k}\in I_{a_k}} N(i_{a_k})\log(N(i_{a_k}))$ can be computed one by one, and we only have to know counts in non-zero cells. The previous section shows that the non-zero counts in table $\mathcal{I}_{\mathrm{a}}$ can be computed in $O(n|\mathcal{I}|)$, if the saturated table $\mathcal{I}$ is small, in $O(n(|\mathcal{I}_{\mathrm{a}}|+\mathrm{N}))$, if the saturated table $\mathcal{I}$ is too large to handle, but the marginal table can be stored, and in $O(nN\log N)$, if the table $\mathcal{I}_{\mathrm{a}}$ is also too large to handle. Thus, unless we only have a very few observations, the computation of the marginal tables is much harder than the determination of the expression. In statistical applications the number of observations is often so that the computation of the marginal tables is the hardest, but in expert system applications one can have networks with several hundred variables on only a few dozen cases.

The total complexity can be expressed as

$$O(ne+n^2+nm\min\{n,m\}+nm\cdot\min\{\max(\max_{a\in\mathcal{M}}(|\mathcal{I}_{\mathrm{a}}|),\min\{|\mathcal{I}|,\mathrm{N}\}),\mathrm{N}\log\mathrm{N}\}). \tag{2.25}$$

Note that $u<2m$. This can also be written as:

$$O(ne+n^2+nm\min(n,m))+\begin{cases} O(nm|\mathcal{I}|) & \text{if } |\mathcal{I}|\le \mathrm{N},\\ O(n(\sum_{a\in\mathcal{M}}|\mathcal{I}_{\mathrm{a}}|+\mathrm{mN})) & \text{if } |\mathcal{I}|>\mathrm{N} \text{ and}\\ & \quad \max_{a\in\mathcal{M}}(|\mathcal{I}_{\mathrm{a}}|)\le \mathrm{N}\log\mathrm{N},\\ O(nmN\log N) & \text{if } \max_{a\in\mathcal{M}}(|\mathcal{I}_{\mathrm{a}}|)>\mathrm{N}\log\mathrm{N}. \end{cases} \tag{2.26}$$

If $\max_{a\in\mathcal{M}}(|\mathcal{I}_{\mathrm{a}}|)>\mathrm{N}\log\mathrm{N}$, then only some of the tables have to be found in $O(nN\log N)$, others, the small, can be found in $O(n(|\mathcal{I}_{\mathrm{a}}|+\mathrm{N}))$.

If the model is not decomposable, then also the last sum of (2.22) can be computed by ønding the terms $\sum_{i_{b_l}\in I_{b_l}} N(i_{b_l})\log(\hat{p}_{\mathcal{B}_1}(i_{b_l}))$ one by one. As discussed in section 4, computing the probabilities of these terms has complexity $O(qms)$, where q is the number of cycles needed for convergence in the IPS-algorithm, m is the number of generators in the generating class and s is the size of the state space.

In Wedelin (1993) a fast method for computing an approximation to the deviance is considered.

2.7 Marginalization of Estimates

Consider the problem of ønding the marginal table of probabilities $\hat{p}(i_b)$ under some model $\mathcal{M}$ for some subset b of Δ. We then ørst ønd the smallest model $\mathcal{M}_{\mathrm{a}}$ onto which $\mathcal{M}$ is collapsible and such that $b\subseteq a$. This set can be found by the algorithm of Geng (1989), an algorithm with a complexity of at least $O(nm^3)$.

But the set can for decomposable models be found in $O(m')$ time by the algorithm of Tarjan & Yannakakis (1984) for selectively reducing an acyclic hypergraph, see also Madigan & Mosurski (1990).

For non-decomposable models we use algorithms investigated in details in chapter 4: To the graph we ørst add a new vertex ω not in the 2-section graph of the model and connect it to all the vertices of b. We then, by Lex M of Rose et al. (1976), ønd a minimal vertex ordering such that the new vertex ω is given the highest ordering, i.e., the new vertex ω is ordered ørst. By a lemma of Leimer (1989) we have a decomposition eliminating vertices not in b if and only if we have a decomposition eliminating the same vertices from the new graph. Finally we by the algorithms of chapter 3 perform decompositions eliminating components not containing any vertices from b by visiting the vertices with respect to the found vertex ordering. This algorithm has for graphical models a complexity of $O(ne)$, for non-graphical models a complexity of $O(ne+nmw)$.

After ønding the smallest model $\mathcal{M}_{\mathrm{a}}$ onto which $\mathcal{M}$ is collapsible, and such that $b \subseteq a$ we then for each cell $i_b \in I_b$ have to sum the expression

$$\hat{p}_{\mathcal{M}_{\mathrm{a}}}(i_a) = c \prod_{k=1..u} N(i_{a_k})^{\nu(a_k)} \prod_{l=1..v} \hat{p}_{\mathcal{B}_{\mathrm{l}}}(i_{b_l}) = c\ \phi_1(i_{c_1}) \cdots \phi_{u+v}(i_{c_{u+v}}) \tag{2.27}$$

over cells $i_{a\setminus b} \in I_{a\setminus b}$. Since

$$\sum_{i_{a\setminus b} \in I_{a\setminus b}} \prod_k \phi_k(i_{c_k}) = \sum_{i_{a\setminus\{b\cup\delta\}} \in I_{a\setminus\{b\cup\delta\}}} \left(\sum_{i_\delta \in I_\delta} \prod_{k:\delta\in c_k} \phi_k(i_{c_k}) \right) \prod_{k:\delta\notin c_k} \phi_k(i_{c_k}). \tag{2.28}$$

we marginalize variables in $a \setminus b$, ıthe things sticking outȷ, before multiplying the expressions, and ønally for each cell $i_b \in I_b$ we multiply the expressions together.

Obviously the variables, vertices, only in one clique (of the graph induced by a and with øll-ins added to the non-decomposable irreducible components) should be marginalized out ørst. Such vertices must belong to non-decomposable irreducible components: If the boundary of a vertex v is complete, then the graphical model $\mathcal{M}$ is also collapsible onto $a \setminus \{v\}$.

For variables δ in more than one of the sets, c_k, the tables $\phi_k(i_{c_k})$, $k : \delta \in c_k$, have to be replaced by a table $\phi_\delta(i_{\cup_{k:\delta\in c_k} c_k \setminus\{\delta\}})$. The size $|\mathcal{I}_{\cup_{\mathrm{k}:\delta\in \mathrm{c_k}} \mathrm{c_k}\setminus\{\delta\}}|$ of this table is usually greater than the total size $\sum_{k:\delta\in c_k} |\mathcal{I}_{\mathrm{c_k}}|$ of the replaced tables, and the required workspace may become too large. The computing time and space requirements depend on the order, in which these variables are marginalized out.

The øll-ins added to the non-decomposable irreducible components should be such that the model associated with the graph induced by a and with øll-ins added to the non-decomposable irreducible components can be collapsed onto a set as small as possible containing b. This is possible, if the state space does not become too large, simply in each non-decomposable irreducible component, start by completing the boundary of each connected component of $a\setminus b$, and then add a øll-in. Then subsets of vertices can be eliminated without marginalization and computing new marginal tables.

Concerning computing a table of marginal probabilities given some of the variables, see Lauritzen & Spiegelhalter (1988).

Kapitel 3

Decomposition of Graphs and Hypergraphs with Identiøcation of Conformal Hypergraphs

Abstract: Decompositions of graphs and hypergraphs by clique separators are investigated. Not only the problem of determining the vertex-sets of the irreducible components is considered, but we also consider the problem of ønding the cliques (the edges in hypergraphs) of the irreducible components. Furthermore, for each minimal clique separator we count the number of components of the graph resulting from removing the separator, i.e., the index.

Key Words: Graphs, Hypergraphs, Decomposition, Index, Irreducible components, Cliques and edges of irreducible components, Hierarchical log-linear models.

3.1 Introduction

Tarjan (1985) presents an algorithm for ønding the clique separators of a graph with n vertices and e edges in a total time of $O(ne + n^2)$. An optimal version of this algorithm is presented in Leimer (1993). The algorithm is in Leimer (1993) optimized in the sense that the separators are minimal and that the graph is only decomposed into the irreducible components. The algorithm presented here is also a little faster than the algorithm of Tarjan (1985), although the complexity of the algorithm remains the same. Tarjan (1985) and Leimer (1993) give algorithms for ønding the vertex-sets of the irreducible components of the graphs. Here we also will consider the problem of decomposing hypergraphs and the problem of ønding the cliques (generators) of the irreducible components.

In Tarjan (1985) four examples are given, where decomposition by clique separators can be used to solve graph problems eŒciently. Tarjan considered minimizing the øll-in caused by Gaussian elimination, øfinding a maximum clique, graph coloring, and øfinding a maximal independent set.

Decompositions of a graph are of similar importance in a statistical context. Darroch et al. (1980) deøned graphical models for contingency tables, where every vertex of a graph is associated with a discrete random variable, and a missing edge in the graph corresponds to the conditional independence of the two variables associated with the two vertices of the missing edge. If a graph corresponds to a graphical model for a contingency table and the graph can be reduced to subgraphs by a clique separator, then the maximum likelihood estimates for the parameters of the model can easily be derived from those in the models for the lower-dimensional tables, represented by the simpler subgraphs. The complexity of algorithms for computing the maximum likelihood estimates in log-linear models without decomposition is exponential in the dimension of the table, and thus the divide-and-conquer approach based on the decomposition algorithms of this chapter will enable handling much larger tables. For hierarchical log-linear models on contingency tables analogous results can be achieved by decomposing the hypergraph associated with the hierarchical model.

By these methods it is possible to compute the deviance in hierarchical models with several hundreds of variables in each irreducible component (under some constraints; see Chapter 2).

The algorithms of this chapter for decomposing the log-linear models on contingency tables are implemented in the program CoCo (Badsberg 1991). The chapter is written with the applications in contingency tables in mind, but the problems are given a pure discrete mathematical formulation for possible application in other areas.

3.2 Notation

Graphs

See the corresponding subsection of chapter 2.

The Fill-In Graph

See the corresponding subsection of chapter 2.

The Index

In decomposable log-linear models for contingency tables, the problem of ønding a closed form expression of the maximum likelihood estimates is a matter of computing the index or the adjusted replication number for subsets of the graph associated with the model (Darroch et al. 1980).

To deøne the index we also have to deøne the pieces of the graph relative to some subset of the vertices: Let $G = (V(G), E(G))$ be a connected graph and $d \subseteq E(G)$ be a complete subset. The pieces of G relative to d are deøned as follows: remove d from G and form the subgraph $G_{V(G)\setminus d}$ with vertices $V(G) \setminus d$ and edges which are those in $E(G)$ that do not involve vertices in d. $G_{V(G)\setminus d}$ now has one or more connected components A_t, $t \in T$, say. Let G_t be the subgraphs of G obtained be rejoining d to the subgraphs A_t, i.e., G_t has the vertex set $A_t \cup d$ and edges which are those in $E(G)$ that only involve vertices in $A_t \cup d$. G_t, $t \in T$, are the pieces of G relative to d.

Then for each complete subset d of $E(G)$ the index $\nu(d)$ is deøned as follows:

$$\nu(d) = 1 - \text{the number of pieces of } G \text{ relative to } d \text{ in which } d \text{ is not a clique.} \tag{3.1}$$

If a graph G with index ν_G is decomposed into the two subgraphs A and B, and these are both connected graphs with indices ν_A and ν_B and with vertex sets a and b respectively, then the indices ν_A, ν_B and ν_G will satisfy

$$\nu_G(d) = \begin{cases} \nu_A(d) + \nu_B(d) & \text{for } d \neq a \cap b, \\ \nu_A(d) + \nu_B(d) - 1 & \text{for } d = a \cap b. \end{cases} \tag{3.2}$$

This is Lemma 8 of Lauritzen et al. (1984).

In this chapter we for a decomposable graph for every (complete) subset d of the vertices of the graph will compute the index deøned by the sum of the indices of the connected components of the graph. If d is not a subset of the vertices of a connected component, then $\nu(d) = 0$. For $d = \emptyset$ we have $\nu(d) = \nu(\emptyset) = 1 - |T|$, where $|T|$ is the number of connected components of the graph.

Hypergraphs

See the corresponding subsection of chapter 2.

The index ν of the hypergraph of any hierarchical log-linear model will together with the maximum likelihood estimates of the irreducible components of the model and marginal tables of counts for subsets for which the index is diœerent from zero enable a fast computation of the maximum likelihood estimates in the model.

On contingency tables a decomposable model is a log-linear model associated with a decomposable graph or an acyclic hypergraph, a graphical model is associated with a graph or a conformal hypergraph, and each (generating class) hypergraph corresponds to a hierarchical model.

Through this chapter m will denote the number of generators in the generating class $\mathcal{H}$, m' will denote the total size of the generating class, i.e., the sum $m' = \sum_{i=1,\cdots,m} |c_i|$, where $\mathcal{H} = \{\mathrm{c_i}, \mathrm{i} = 1, \cdots, \mathrm{m}\}$, n the number of vertices in the 2-section graph $G^{\mathcal{H}} = (V(G^{\mathcal{H}}), E(G^{\mathcal{H}}))$ for the hypergraph $\mathcal{H} = (\mathrm{V}(\mathcal{H}), \mathcal{H})$, e the number of edges in the 2-section graph, and e' will denote the number of edges in a øll-in graph of the 2-section graph.

3.3 A Simple Algorithm for Decomposition

In Geng (1989) the algorithm HiModel is presented. The algorithm checks whether or not a hypergraph is decomposable and decomposes the hypergraph into sub-hypergraphs as small as possible. In this recursive algorithm an algorithm for eliminating vertices only in one clique is used. This sub-algorithm will handle and give the ønal result of decomposable hypergraphs.

A ørst look at this sub-algorithm suggests by the number of nested loops a complexity of $O(m^4)$ set operations. For each set, i.e., generator, in the generating class of the hypergraph all other generators are subtracted from the considered set. If the resulting set is non-empty, then vertices that are in one and only one clique are identiøed and eliminated. Such a step of checking all generators for vertices in one and only one clique thus requires $O(m^2)$ set operations. This step is repeated until no more vertices that are in one and only one clique are found. Ignoring the removal of generators which are subsets of other generators, then, if only one generator is eliminated in each step, the complexity of the algorithm is $O(m^3)$ in set operations. Consider, e.g., a sequence of variables with each pair of variables conditionally independent given two consecutive and intermediate variables, i.e., the generating class $[[1,2,3][2,3,4][3,4,5]\cdots[n-2,n-1,n]]$. To prevent the algorithm from eliminating more than one vertex in each step remove the ørst 2-order interaction, i.e., consider the generating class $[[1,2][1,3][2,3,4][3,4,5]\cdots[n-2,n-1,n]]$. Empirical studies, see Model 3e in section 3.6, of this generating class with n the integers from 16 to 127 suggest a complexity of $O(m^3)$ in set operations. Thus the complexity of the sub-algorithm is probably $O(nm^3)$. Section 3.6 also gives empirical studies of other models.

The above gives a simple implementation of Graham's algorithm to decompose a hypergraph. Graham's algorithm applies to decide acyclicity of a hypergraph the following two operations repeatedly until neither can be applied:

- Eliminate a vertex that occurs in only one generator,
- Eliminate a generator that is contained in another generator.

For references, see Beeri et al. (1983) or Tarjan & Yannakakis (1984). This problem can be solved more eŒciently for decomposable models by the algorithm for selectively reducing an acyclic hypergraph as discussed in Tarjan & Yannakakis (1984).

To handle non-acyclic hypergraphs, i.e., non-decomposable and/or non-graphical models, Geng gives a recursive algorithm. Let $\mathcal{C} = (\mathrm{V}(\mathcal{C}), \mathcal{C})$ be the hypergraph to decomposed. After eliminating successively all vertices only in one generator, this algorithm searches for a generator $c \in \mathcal{C}$ according to which the hypergraph $\mathcal{C} = (\mathrm{V}(\mathcal{C}), \mathcal{C})$ can be decomposed, and then calls itself recursively with the two sub-hypergraphs $\mathcal{A} = (\mathrm{V}(\mathcal{A}), \mathcal{A})$ and $\mathcal{B} = (\mathrm{V}(\mathcal{B}), \mathcal{B})$ resulting from the decomposition.

Generators $c \in \mathcal{C}$ are visited until a generator c, according to which the hypergraph $\mathcal{C}$ can be decomposed, is found. For each generator c of the hypergraph $\mathcal{C}$ (and the empty set to decompose hypergraphs which are not connected) the following is performed to test whether the hypergraph is decomposable with respect to c. First an arbitrary generator $a \in \mathcal{C} \setminus \{\mathrm{c}\}$ diœerent from c is allocated together with c to an empty generating class, say $\mathcal{A}$. Then all other generators $d \in \mathcal{C} \setminus \mathcal{A}$ of $\mathcal{C}$ are visited, and generators d intersecting the set diœerence between the vertices of $\mathcal{A}$ and the separator c, i.e., generators d with $d \cap (\cup_{a \in \mathcal{A}} a \setminus c) \neq \emptyset$, are added to $\mathcal{A}$. Such steps of searching for generators to add to the generating class $\mathcal{A}$ are repeated until no generators are added to $\mathcal{A}$ in a complete step of visiting all generators not on $\mathcal{A}$, i.e., all generators of $\mathcal{C} \setminus \mathcal{A}$. The hypergraph can be decomposed according to the considered set c, if part $\mathcal{A}$ is not the whole hypergraph to decompose, i.e., if $\mathcal{A} \neq \mathcal{C}$.

If only one generator is added in each step, then the process of checking whether the hypergraph is decomposable with respect to a generator thus requires $O(m^2)$ set operations. If the number of generators to visit before ønding a set according to which the hypergraph is decomposable is proportional to m, and if in each decomposition the size of the greatest of the two hypergraphs resulting of the decomposition is only a few generators smaller than the hypergraph to decompose, then the total complexity of the algorithm is $O(m^4)$ in set operations. Consider, e.g., a sequence of variables conditionally independent given two consecutive variables and all 2-order interactions removed, i.e., the generating class $[[1,2][1,3][2,3][2,4][3,4][3,5]\cdots[n-2,n-1][n-2,n][n-1,n]]$. Empirical studies, see Model 3a in section 3.6, of this generating class with n the integers from 16 to 127 suggest a complexity of $O(m^{3.1})$ in set operations. Maybe it is possible to ønd generating classes showing that the complexity of the algorithm is worse than $O(m^{3.1})$, but the above considerations show that the complexity is limited by $O(nm^4)$. The complexity of the recursive algorithm seems to be $O(nm^3)$.

If the hypergraph by each decomposition is divided into hypergraphs with half the number of generators, then the computing time to solve a problem of size n, m by the algorithm is given by $T(n,m) = nm^3 + 2T(n, m/2)$, and the complexity of the algorithm is $O(nm^3)$.

But a complexity of $O(ne + nm\min(n,m))$ for identifying and decomposing hypergraphs and returning the generators of the irreducible subgraphs can be achieved by the algorithms presented in Tarjan & Yannakakis (1984) and Tarjan (1985). If the class of generating class hypergraphs is restricted to conformal hypergraphs, then decomposable graphs can be identiøed and decomposed (with computation of the index) in $O(e + nm\log m)$ with $m \leq n$. Compared to the complexity around $O(nm^3)$ of the algorithm of Geng (1989) this is much better, unless $e > nm^3$, i.e., the number of edges in the 2-section graph is very large compared to the number of edges in the hypergraph.

Working directly on the hypergraph, acyclic hypergraphs can be decomposed in $O(n + m')$ time by the algorithm of Tarjan & Yannakakis (1984). Here m' is the total size of the generating class, i.e., the sum of the cardinality of the generators.

3.4 Graphs

Assume ørst that the hypergraph is conformal. That is the cliques of its' 2-section graph are the edges of the hypergraph, and so the model is a graphical model. An algorithm for decomposing non-conformal hypergraphs is given in section 3.5.

3.4.1 Identiøcation of the Decomposable Graphs

The algorithm Maximum Cardinality Search of Tarjan & Yannakakis (1984) ønds an ordering of the vertices in a graph in $O(e + n)$ time.

In Maximum Cardinality Search the vertices are numbered from n to 1 in decreasing order as follows. Give an arbitrary vertex the ordering n. As the next vertex to give

```
Index-Decomposable-Graph(Δ, ∂, π^{-1})
    local a, i, C;
    Expression ::= Null;
    for i ::= 1 to |Δ|
        C(π^{-1}(i)) ::= {π^{-1}(i+1), ···, π^{-1}(|Δ|)} ∩ ∂π^{-1}(i);
    i ::= 1;
    while i ≤ |Δ|
    {
        a ::= C(π^{-1}(i));
        Update index(a ∪ {π^{-1}(i)}, 1, Expression);
        while i < |Δ| and C(π^{-1}(i+1)) ∪ {π^{-1}(i+1)} ⊆ a
            i ::= i + 1;
        Update index(C(π^{-1}(i)), −1, Expression);
        i ::= i + 1
    }
return(Expression);
```

Figur 3.1: Index of Decomposable Graphs.

a number, select the vertex adjacent to the highest number of previously numbered vertices, breaking ties arbitrarily.

The ordering found is a perfect vertex elimination ordering, if the graph is decomposable. With the ordering found, decomposability can be checked in time $O(e+n)$ by the algorithm Test for Zero Fill-In of the same paper.

3.4.2 Decomposition of the Decomposable Graphs

Given a perfect elimination ordering of vertices in a decomposable generating class it is a trivial task to find a closed form expression of the maximum likelihood estimates, i.e., the index, in $O(e+n)$ time. Vertices are eliminated one by one according to the perfect elimination ordering with recursive application of (3.2). Also from the graph represented as adjacency lists for all vertices v_k to eliminate in the graph the separators $\partial v_k \cap \{v_{k+1}, \cdots, v_n\}$ and vertex sets $\{v_k\} \cup (\partial v_k \cap \{v_{k+1}, \cdots, v_n\})$ of the eliminated subgraphs can be found in a total time of $O(e)$.

The index of the graph can also be found in $O(e + nm \log m)$ time. Instead of eliminating vertices one by one, we in each step decompose with respect to a minimal separator. Vertices are still visited according to the perfect elimination ordering. The minimal separators are found analogously to how they are found for the non-decomposable graphs, see the next section. A list of sets a_k and indices $\nu(a_k)$ of those sets is maintained. The procedure Update index$(a, \iota, \text{Expression})$ inserts $\nu(a) = \iota$ in the list, if the index is not defined for that set. If the index is already defined for a in the list, then the index $\nu(a)$ for a is updated to $\nu(a) + \iota$. If the sets in the list are stored in a binary search tree, then the complexity of inserting a set or updating the index of a set in the binary search tree will be the logarithm of the number of sets for which the index is already defined multiplied by the time used to determine for two sets whether the one precedes the other. The list of sets has length less than $2m-1$, and has to be updated for each decomposition. Since the generating class is

```
Index-Acyclic-Hypergraph((R_1, ..., R_J), β, γ)
    local j;
    Expression ::= Null;
    for j ::= 1 to k
    {
        Update index(R_j, 1, Expression);
        Update index(R_j ∩ ⋃_{l=1}^{j-1} R_l, -1, Expression);
    }
return(Expression);
```

Figur 3.2: Index of Acyclic Hypergraphs.

decomposable, we have $m \leq n$, and we cannot perform more than m decompositions with respect to minimal separators. Thus the generating class can be decomposed and the index of the 2-section graph found in $O(e + nm \log m)$ time.

Using Lex M instead of Maximum Cardinality Search to ønd the perfect elimination ordering would ensure that, when a separator occurs more than once, then the graph is likely to be decomposed with respect to that separator in succession with no decompositions with respect to other separators in between (see the next section). If a separator occurs more than once, then the separator is contained in separators occurring between its instances. Thus, if the separators are inserted in a list ordered according to when the separators are used, we do not, when applying a separator, have to scan the list to see, whether the separator is already in the list. But Lex M runs in $O(ne)$ time which should be compared to the complexity $O(nm \log m)$ of updating the list of the index. Since $m < n$ for decomposable graphs and $e \sim n^2$, the latter is preferred.

By the orderings produced by the algorithm Restricted Maximum Cardinality Search on Hypergraphs of Tarjan & Yannakakis (1984) it is possible to write an algorithm for ønding the index of the graph in $O(m' + nm \log m)$ time:

The algorithm Restricted Maximum Cardinality Search on Hypergraphs computes for acyclic hypergraphs a perfect elimination ordering α of the vertices: $\alpha(v_i) = i$ and the ordering $\{v_1, \cdots, v_n\}$ is perfect. The ordering is computed as follows. Pick as the ørst edge an arbitrary edge, and number all the vertices of that edge in increasing order. When all the vertices of the selected edge are ordered, select as the next edge the edge having as many numbered vertices as possible. Number all the unnumbered vertices of the selected edge. Continue until all the vertices are ordered. The algorithm has a complexity of $O(m')$.

In addition to numbering the vertices, the program performs the following computations: It numbers the selected edges from 1 to k in order of their selection; if S is the ith edge selected, then $S = R_i$ and $\beta(S) = i$, $\beta(S)$ is an index (diœerent from ν) of S. It extends this numbering to the vertices by deøning

$$\beta(v) = \min\{\beta(R) |\ R \text{ is selected and } v \in R\}. \tag{3.3}$$

The integer $\beta(v)$ is the minimum number of the indices of the edges containing v. Finally, for each edge S it computes $\gamma(S)$, deøned by

$$\gamma(S) = \max\{\beta(v) |\ v \in S\}, \tag{3.4}$$

if S is not among the selected edges, and

$$\gamma(S) = \max\{\beta(v) |\ v \in S \text{ and } \beta(v) < \beta(S)\}, \tag{3.5}$$

if S is among the selected edges. (All edges are selected for acyclic hypergraphs.) The integer $\gamma(S)$ is the index of the edge selected latest before S such that the edge when selected contained unnumbered vertices also in S.

Theorem 5 of Tarjan & Yannakakis (1984) states that a hypergraph $\mathcal{H}$ is acyclic if and only if for each $i \in [1, k]$ and for each edge S of $\mathcal{H}$ such that $\gamma(S) = i$ we have $S \cap \{v |\ \beta(v) < i\} \subseteq R_i$. (Since $\beta(v) = i$ implies $v \in R_i$, this condition is equal to $S \cap \{v |\ \beta(v) \leq i\} \subseteq R_i$.) This theorem enables an $O(n+m)$-time acyclicity test, see Tarjan & Yannakakis (1984).

Since

$$R_j \cap \bigcup_{l=1}^{j-1} R_l = R_j \cap \{v |\ \beta(v) < \beta(R_j)\} = R_j \cap \{v |\ \beta(v) \leq \gamma(R_j)\} \tag{3.6}$$

we by the above theorem for acyclic hypergraphs have

$$(R_j \cap \bigcup_{l=1}^{j-1} R_l) \subseteq R_{\gamma(R_j)}. \tag{3.7}$$

Thus the algorithm Restricted Maximum Cardinality Search on Hypergraphs orders the cliques R_j of an acyclic hypergraph in the sequence $(R_1, \cdots, R_J)$ such that

$$\forall\ j = 2, \cdots, J \ \ \exists\ k, 1 \leq k < j : \ \ (R_j \cap \bigcup_{l=1}^{j-1} R_l) \subseteq R_k, \tag{3.8}$$

i.e., the sequence $(R_1, \cdots, R_J)$ fulfills the running intersection property.

Thus R_j and $\bigcup_{l=1}^{j-1} R_l$ forms a decomposition of the hypergraph induced by $\bigcup_{l=1}^{j} R_l$, and we have an algorithm (figure 3.2) for finding the index ν of an acyclic hypergraph.

Expression (3.6) gives various ways to compute the sets $R_j \cap \bigcup_{l=1}^{j-1} R_l$, all in a total time of $O(m')$, and thus the index of an acyclic hypergraph can be found in $O(m' + nm \log m)$ time.

3.4.3 Decomposition of the Non-Decomposable Graphs

Decomposition

In Tarjan (1985) an algorithm for finding the clique separators of a graph in a total time of $O(ne + n^2)$ is presented. Leimer (1993) presents a version of this algorithm that is optimal in the sense that the separators are minimal and that the graph is only decomposed into the irreducible components.

The vertices of the graph are visited according to a minimal vertex elimination ordering π. When visiting the i-th vertex v_i, then, if the vertex set

$$C(v_i) = \partial_\pi v_i \cap \{v_{i+1}, \cdots, v_n\}, \tag{3.9}$$

i.e., the subgraph induced vertices adjacent to v_i in the øll-in graph and following v_i is complete, and if $C(v_i)$ separates the graph into two non-empty components of which one component contains v_i, then the graph can be decomposed with respect to v_i, and the component with v_i can be discarded as an irreducible subgraph.

This gives us the following decomposition algorithm:

- Find a minimal ordering π of G and compute $C(v_i)$ for each vertex v_i.
- Repeat the following step for each vertex v_i in increasing order with respect to π: Let A be the vertex set of the connected component of $G_{V(G)\setminus C(v_i)}$ containing v_i, and let $B = V(G) \setminus (C(v_i) \cup A)$. If $C(v_i)$ is complete in G and $B \neq \emptyset$, decompose G into $G' = G_{A\cup C(v_i)}$ and $G'' = G_{B\cup C(v_i)}$, separated by $C(v_i)$. Replace G by G'', and discard G' as a prime.

Visiting all the vertices according to this algorithm will perform all possible decompositions of the graph (Tarjan 1985):

Lemma 1 If G contains a clique separator, some decomposition step will be successful.

Proof This is Lemma 2 of Tarjan (1985). □

Lemma 2 The decomposition algorithm is correct.

Proof This is Theorem 2 of Tarjan (1985). □

Decomposition into the irreducible components with minimal separators can be achieved by, if $C(v_i) \subseteq \partial v_i$, i.e., there are no øll-in edges from v_i to vertices following v_i, not decomposing with respect to $C(v_i)$, but decomposing with respect to $C(v_j)$, where

$$j = \max_{j=i+1,\cdots,n} (C(v_j) \cup \{v_j\}) \subseteq C(v_i). \tag{3.10}$$

Lemma 3 Let there be given an elimination ordering $\{v_1, \cdots, v_n\}$ of the vertices $v_{(i)}$, $i = 1, \cdots, n$, in a graph G, and let i be the smallest index such that $C(v_i)$ is complete. If $C(v_i) \subseteq \partial v_i$, then v_i is a perfect vertex in G.

Proof Let a be the vertices of the connected component of $G_{V(G)\setminus C(v_i)}$ containing v_i, and let b be $V(G) \setminus (a \cup C(v_i))$. $G_{V(G)\setminus C(v_i)}$ is the subgraph of G induced by $V(G) \setminus C(v_i)$. Since $C(v_i)$ is complete, then $G_{a\cup C(v_i)}$, $G_{b\cup C(v_i)}$ is a decomposition of G when $b \neq \emptyset$. Assume that $C(v_i)$ is a subset of ∂v_i. Then $a = \{v_i\}$ because, if $a \setminus \{v_i\} \neq \emptyset$, then let v_j be the vertex with the greatest order in $a \setminus \{v_i\}$. From the deønition of the øll-in graph, $C(v_i)$ and a it follows that the order of v_j is less than the order of v_i. (Assume that there are vertices in $a \setminus \{v_i\}$ with an order greater than that of v_i, and let v_j be the vertex of these with the smallest order. Since v_j is in the connected component of $G_{V(G)\setminus C(v_i)}$ containing v_i there must be a path from v_i to v_j not parsing $C(v_i)$. If $\{v_i, v_j\}$ is not an edge of G, then $\{v_i, v_j\}$ is an øll-in edge for the vertex ordering, and thus $v_j \in C(v_i)$ regardless of $\{v_i, v_j\}$ is an edge of G contradicting $v_j \in a$.) Since $G_{a\cup C(v_i)}$, $G_{b\cup C(v_i)}$ is a decomposition of G, and since

$v_j \in a$ there can be no edges in the øll-in graph from v_j to vertices in b. $C(v_j)$ is thus a subset of $\{v_i\} \cup C(v_i)$, and thus $C(v_j)$ is complete contradicting the assumption that v_i should be the vertex with the smallest order such that $C(v_i)$ is complete. □

The above theorem is also given in Badsberg (1986).

Lemma 4 Let $\{v_1, \cdots, v_n\}$ be an minimal vertex elimination ordering of the vertices $v_{(i)}$, $i = 1, \cdots, n$, in a graph G, and let v_i be the minimum vertex such that $C(v_i)$ is a clique separator of G. If $C(v_{i+1}) \cup \{v_{i+1}\} \subseteq C(v_i)$ then $C(v_{i+1})$ is also a clique separator of G.

Proof Let a be the vertices of the connected component of $G_{V(G) \setminus C(v_i)}$ containing v_i, and let b be $V(G) \setminus (a \cup C(v_i))$. After eliminating a the decomposition algorithm will be successful at v_{i+1} since $C(v_{i+1}) \subseteq C(v_i)$ and thus is complete. By lemma 2 the vertex v_{i+1} is thus separated from b by $C(v_{i+1})$.

(Without reference to lemma 2 we can show that the vertex v_{i+1} not is adjacent to any vertex of b. Assume $v_{i+1} \sim_\pi v_k$ for some $v_k \in b$, $k > i+1$. But then $v_k \in C(v_{i+1})$ contradicting $v_k \in b$. Assume $v_{i+1} \sim_\pi v_h$ for some $v_h \in b$, $h < i$. Then $v_h \sim_\pi v_k$ for some $v_k \in b$, $k > i+1$, else $C(v_h) \subseteq C(v_i)$ and thus complete contradicting v_i be the minimum vertex such that $C(v_i)$ is a clique separator of G. Since thus $v_h \sim_\pi v_{i+1}$, $h < i$, and $v_h \sim_\pi v_k$, $h < k$, we must then have an edge $\{v_{i+1}, v_k\}$ of the øll in graph of G, and thus $v_k \in C(v_{i+1})$ contradictiong $v_k \in b$.)

We then have that $a \cup \{v_{i+1}\}$ is separated from b by $C(v_{i+1})$, and thus $G_{a \cup \{v_{i+1}\} \cup C(v_{i+1})}$, $G_{b \cup C(v_{i+1})}$ is a decomposition of G.

Since $C(v_i)$ is complet the vertex v_{i+1} is adjacent to each vertex of $C(v_i)$ and thus $C(v_i) = C(v_{i+1}) \cup \{v_{i+1}\}$. We then have that the subgraph $G_{a \cup \{v_{i+1}\} \cup C(v_{i+1})}$ eliminated by $C(v_{i+1})$ the same subgraph as $G_{a \cup C(v_i)}$ eliminated by C_i.

□

Theorem 2 Let $\{v_1, \cdots, v_n\}$ be an elimination ordering of the vertices $v_{(i)}$, $i = 1, \cdots, n$, of the graph G, let i be the smallest index such that $C(v_i)$ is complete, assume $C(v_i) \subseteq \partial v_i$, that G is decomposable with respsect to $C(v_i)$, and set

$$j = \min_{j=i+1,\cdots,n} (C(v_j) \cup \{v_j\}) \not\subseteq C(v_i) - 1. \tag{3.11}$$

Then $C(v_j)$ is a minimal separator of G.

Proof By the above lemma $C(v_i)$ is a clique separator of G. □

If j is determined as above, then by the construction of Lex M there cannot be vertices in $C(v_j)$ following v_j and not connected to at least one of the vertices following the vertices of $C(v_j)$.

Because, if there were such a vertex v_k in $C(v_j)$, then, when ordering v_k in the lexicographical search (with a higher order) before v_j, the label of v_j is lexicographically greater than the label of v_k, since v_j is connected to vertices following vertices of $C(v_j)$ and v_k is not connected to vertices following vertices of $C(v_j)$, contradicting that v_k should be ordered before v_j.

Theorem 3 Decomposing with respect to $C(v_j)$ as determined above is the same as decomposing with respect to $C'(v_i)$, where

$$C'(v_i) = \{w \in C(v_i) \mid w \text{ is adjacent to at least one vertex in } V(G) \setminus (a \cup C(v_i))\}. \tag{3.12}$$

Proof □

In Tarjan (1985) it is noted that decomposing with respect to $C'(v_i)$ will give smaller separators and fewer components, and improve the algorithm slightly, but not improve the worst-case complexity of the algorithm.

In Leimer (1993) it is proven that when decomposing with respect to minimal separators and using the vertex ordering of Lex M, then a subgraph eliminated at one stage in the elimination process cannot be a subgraph of subgraphs eliminated previously. Thus the graphs are precisely decomposed into the irreducible components.

Generators

Tarjan (1985) and Leimer (1993) give algorithms for ønding the vertex-sets of the irreducible components. Besides considering this problem, we will also consider here the problem of ønding the cliques of the irreducible components.

Let v_i be the vertex with the smallest order such that $C(v_i)$ is complete, i.e., $C(v_i) \subseteq c$ for some $c \in \mathcal{C}$. $\mathcal{C}$ are the cliques of the graph. Let a be the vertices of the connected component of $G_{V(G)\setminus C(v_i)}$ containing v_i, and let b be $V(G) \setminus (a \cup C(v_i))$. If $b \neq \emptyset$, then divide the generators $\mathcal{C}$ into the two classes $\mathcal{A}$ and $\mathcal{B}$: $\mathcal{A} = \{\mathrm{c} \in \mathcal{C} \mid \mathrm{c} \cap \mathrm{a} \neq \emptyset\}$ and $\mathcal{B} = \{\mathrm{c} \in \mathcal{C} \mid \mathrm{c} \cap \mathrm{a} = \emptyset\}$. Since the following lemmas and theorem are also true when decomposing hypergraphs, we write generators and not cliques. Also the following are true regardless of whether we decompose with respect to all the sets $C(v_i)$ such that $C(v_i) \subseteq c$ for some $c \in \mathcal{C}$ or only decompose with respect to the sets $C'(v_i) \subset c$ for some $c \in \mathcal{C}$ with $b \neq \emptyset$.

Lemma 5 If $C(v_i) \not\subseteq \partial v_i$, then the generators of $G_{a \cup C(v_i)}$ are $\mathcal{A} \cup \{\mathrm{C(v_i)}\}$ and the generators of $G_{b \cup C(v_i)}$ are $\mathcal{B}$.

Proof Clearly the generators of $G_{a \cup C(v_i)}$ are a subset of $\mathcal{A} \cup \{\mathrm{C(v_i)}\}$ and the generators of $G_{b \cup C(v_i)}$ are a subset of $\mathcal{B} \cup \{\mathrm{C(v_i)}\}$.

We prove that $C(v_i) \subseteq c$ for some $c \in \mathcal{B}$: $C(v_i)$ is complete, and thus $C(v_i) \subseteq c$ for at least one $c \in \mathcal{C}$. Such generators c must be allocated to $\mathcal{B}$: Assume $c \cap a \neq \emptyset$ for one of these sets c. If $v_i \in c \cap a$, then $\{v_i\} \cup C(v_i)$ is complete since $\{v_i\} \cup C(v_i) \subseteq c$, and thus $C(v_i) \subseteq \partial v_i$ in contradiction with the assumption of the lemma. Let v_j be the vertex with the highest ordering in $c \cap a$. If there is such a vertex $v_j \in c \cap a$ diœerent from v_i, then the ordering of v_j is lower than the ordering v_i, and $C(v_i) \subseteq \partial v_j$ since $\{v_j\} \cup C(v_i) \subseteq c$. Thus v_j is connected to all vertices of $C(v_i)$, but by the assumption of the lemma, v_i is not connected to all the vertices of $C(v_i)$, and thus v_j has a label lexicographically higher than the label of v_i, and thus should be ordered before v_i and with a higher ordering. Thus $c \cap a = \emptyset$ and $C(v_i) \subseteq c$ for some $c \in \mathcal{B}$.

The above proves that $C(v_i) \not\subseteq c$ for all $c \in \mathcal{A}$ and thus the generators of $G_{a \cup \setminus C(v_i)}$ are $\mathcal{A} \cup \{\mathrm{C(v_i)}\}$. □

Lemma 6 If $C(v_i) \subseteq \partial v_i$, then the only generator of $G_{a\cup\setminus C(v_i)}$ is $\{v_i\} \cup C(v_i)$ and the generators of $G_{b\cup C(v_i)}$ are $\mathcal{B}$, if $c \in \mathcal{B}$ exists such that $C(v_i) \subseteq c$, else the generators of $G_{b\cup C(v_i)}$ are $\mathcal{B} \cup \{\mathrm{C(v_i)}\}$.

Proof If v_i is the vertex with the smallest order such that $C(v_i)$ is complete, and $C(v_i) \subseteq \partial v_i$, then v_i is a perfect vertex in the graph, and the two generating classes $\{\{v_i\} \cup C(v_i)\}$ and $\mathcal{B}'$ are a decomposition of $\mathcal{C}$, where $\mathcal{B}'$ is $\mathcal{B}$, if $c \in \mathcal{B}$ exists such that $C(v_i) \subseteq c$, else $\mathcal{B} \cup \{\mathrm{C(v_i)}\}$. □

Let $a_{(i)}$ be the vertices of the connected component of $G_{V(G)\setminus(C(v_i)\cup A_{(i)})}$ containing v_i when decomposing with respect to a complete subset $C(v_i)$, where

$$A_{(i)} = \bigcup_{j<i \text{ and } C(v_j)\subseteq c \text{ for } c\in\mathcal{C}} a_{(j)}, \tag{3.13}$$

are the vertices removed from the graph before decomposing with respect to $C(v_i)$. The irreducible components with vertices $a_{(i)} \cup C(v_i)$ can then be ordered with respect to $A_{(i)}$ such that $a_{(i)} \cup C(v_i)$ precedes $a_{(j)} \cup C(v_j)$, if $A_{(i)} \subset A_{(j)}$. We write $a_{(i)}$ and $A_{(i)}$ and not $a(v_i)$ and $A(v_i)$ since the sets are not deøned for all vertices $v(i)$.

Divide the generators $\mathcal{C}$ into the classes $\mathcal{A}_\mathrm{i} = \{\mathrm{c} \in \mathcal{C} \mid \mathrm{c} \cap \mathrm{a_{(i)}} \neq \emptyset \text{ and } \mathrm{c} \cap \mathrm{A_{(i)}} = \emptyset\}$. Let $\mathcal{S}$ be the set of separators $C'(v_i)$ for eliminating complete subgraphs. Note that the same separator may occur more than once during the decomposition, i.e., there may exists vertices v_i and v_j, $v_i \neq v_j$, such that $C(v_i) = C(v_j)$ (or $C'(v_i) = C'(v_j)$) but $a_{(i)} \neq a_{(j)}$[1]. Divide the sets of $\mathcal{S}$ into the classes $\mathcal{S}_\mathrm{i} = \{\mathrm{c} \in \mathcal{S} \mid \mathrm{c} \cap \mathrm{a_{(i)}} \neq \emptyset \text{ and } \mathrm{c} \cap \mathrm{A_{(i)}} = \emptyset\}$.

Theorem 4 The generators of the non-decomposable component with vertices $a_{(i)} \cup C(v_i)$ are $\mathcal{A}_\mathrm{i} \cup \{\mathrm{C(v_i)}\} \cup \mathcal{S}'_\mathrm{i}$, where $\mathcal{S}'_\mathrm{i} = \{\mathrm{c} \in \mathcal{S}_\mathrm{i} \mid \forall\, \mathrm{a} \in \mathcal{S}_\mathrm{i} : \mathrm{c} \not\subset \mathrm{a} \text{ and } \forall\, \mathrm{a} \in \mathcal{A}_\mathrm{i} \cup \{\mathrm{C(v_i)}\} : \mathrm{c} \not\subseteq \mathrm{a}\}$. If $a_{(i)} \cup C(v_i)$ is a complete subgraph, then the only generator of the subgraph is $a_{(i)} \cup C(v_i)$.

Proof By induction on the number of separators. □

The following theorem can be used to reduce the computing time to determine whether a separator has already occurred during the decomposition of the graph. This is done by inserting the separators into a list ordered according to the order in which they are used and such that the last used separator is visited ørst.

Theorem 5 If an elimination ordering π of the vertices is found by Lex M and for two vertices v_i and v_j, $i < j$, we have $C(v_j) \subseteq \partial v_i$, then for $i < k < j$ we also have $C(v_j) \subseteq \partial v_k$.

Proof Pick a vertex v_k, $i < k < j$. Assume $C(v_j) \not\subseteq \partial v_k$. Then $C(v_j) \setminus \partial v_k \neq \emptyset$. Thus ∂v_k must contain at least one vertex not in $C(v_j)$ to ensure that v_k is ordered before v_i, since $C(v_j) \subseteq \partial v_i$. Such a vertex must have order greater than the vertex with

[1] If the separators $C'(v_i)$ are simply added to $\mathcal{S}$ during the decomposition, then $\mathcal{S}$ is actually not a set, but should be considered as a list of elements, possibly not all dioerent.

the greatest ordering in $C(v_j) \setminus \partial v_k$, else the label of v_k is lexicographically smaller than that of v_i, since $C(v_j) \subseteq \partial v_i$. (See the next section about the label and the lexicographical order of the label.) But then the label of v_k is lexicographically greater than the label of v_j and thus v_k should have been ordered before v_j and with a larger ordering. □

Corollary 1 When complete subgraphs are eliminated with respect to the same separator c more than once, then, if the graph is decomposed with respect to another separator a between decompositions with respect to the separator c, the separator a will contain the separator c.

Proof Let two vertices v_i and v_j, $i < j$, be eliminated with respect to the same separator $c = C(v_j) = C(v_i)$ and let v_k be eliminated between the two vertices v_i and v_j, $i < k < j$, with respect to the separator $a = C(v_k)$. If v_i is a perfect vertex in the subgraph induced by $\{v_i, \cdots, v_n\}$, then $C(v_i) \subseteq \partial v_i$ and then, since $C(v_j) = C(v_i)$, we also have $C(v_j) \subseteq \partial v_i$, and it follows directly of the above theorem that $C(v_j) \subseteq \partial v_k$. Since $c = C(v_j)$, $k < j$ and $C(v_j) \subseteq \{v_j, \cdots, v_n\}$ we have $c = c \cap \{v_k, \cdots, v_n\} \subseteq \partial v_k \cap \{v_k, \cdots, v_n\}$, and thus $c \subseteq a$ since $\partial v_k \cap \{v_k, \cdots, v_n\} \subseteq a$. □

This means that when a separator occurs more than once, then the graph is likely to be decomposed with respect to that separator in succession with no decompositions with respect to other separators in between.

Complexity

The minimal ordering of the vertices can be found by the algorithm Lex M of Rose et al. (1976) in $O(ne)$. In Lex M the vertices are numbered from n to 1 in decreasing order using a modiøed lexicographical search, where lexicographical search is deøned as follows. For each unnumbered vertex v, maintain a label, a list of the numbers of the numbered vertices adjacent to v, with the numbers in each list arranged in decreasing order. For the next vertex to number, select the vertex whose label is lexicographically the greatest, breaking ties arbitrarily. Although somewhat complicated, lexicographical search can be implemented to run in $O(e + n)$ time (Rose et al. 1976), and the modiøed version of lexicographical search used in Lex M to ønd the minimal ordering runs in $O(ne)$ time.

Computing the sets $C(v_i)$ is essentially a matter of computing the øll-in F_π which takes $O(e' + n)$ with the algorithm Fill In Computation of Tarjan & Yannakakis (1984). Here e' is the number of edges in the øll-in graph. Given an appropriate representation of the øll-in graph, e.g., adjacency-lists, where the vertices adjacent to a vertex are sorted according to a given order, then it can be checked in $O(e)$ whether $C(v_i)$ is complete. The separators $C(v_i)$ and the vertices for which $C(v_i)$ is complete are determined during ønding the vertex ordering by Lex M in Leimer (1993).

The vertices of the component containing v_i can be found in $O(e+n)$ by a function Find Connected Component(a, v_i, ∂), which from the vertex v_i will visit vertices only in a by a bfs, Breadth-ørst search, and with the adjacency lists ∂ as argument. Thus applying the decomposition step to a single vertex takes $O(e + n)$ time, and the total decomposition into irreducible components takes $O(ne + n^2)$. By this we have only

```
Index-Graphs(𝒞, Δ, ∂, C, π⁻¹)
    local a, b, d, 𝒜, ℬ, 𝒮, i, j, u;
    Expression ::= Null;
    d ::= Δ;
    b ::= d;
    i ::= 1;
    𝒮 ::= ∅;
    while b ≠ ∅
    {
        while ¬Complete(C(π⁻¹(i)))
            i ::= i + 1;
        u ::= π⁻¹(i);
        if C(u) ⊆ ∂u
        then {
            j ::= i + 1;
            while C(π⁻¹(j)) ∪ {π⁻¹(j)} ⊆ C(u) and j < |Δ|
                j ::= j + 1;
            a ::= {u} ∪ C(u);
            Update index(a, 1, Expression);
            Update index(∅, −1, Expression);
            i ::= j − 1;
            u ::= π⁻¹(i);
            a ::= a \ C(u);
            𝒜 ::= Return and delete generators(a, 𝒞);
            ℬ ::= Return and delete generators(a, 𝒮)
        } else {
            a ::= Find Connected Component(d \ C(u), u, ∂);
            𝒜 ::= Return and delete generators(a, 𝒞);
            ℬ ::= Return and delete generators(a, 𝒮);
            𝒜 ::= Reduce(𝒜 ∪ {C(u)} ∪ ℬ);
            Push Irreducible Component(a ∪ C(u), 𝒜, Expression) };
        𝒮 ::= {C(u)} ∪ 𝒮;
        if d ≠ a ∪ C(u)
        then {
            Update index(C(u), −1, Expression);
            Update index(∅, 1, Expression) }
        d ::= d \ a;
        b ::= d \ C(u);
        i ::= i + 1 }
return(Expression);
```

Figur 3.3: Index of Simple Undirected Graphs.

identiøed the vertex-sets of the irreducible components. The number e of edges is often greater than the number n of vertices, and $e \leq n(n-1)/2$ grows with n^2, so the complexity is often written as $O(ne)$.

The procedure Push Irreducible Component$(a, \mathcal{A}, \text{Expression})$ will update the data structure holding the result of the decomposition with an irreducible component with the generating class $\mathcal{A}$. If a is the only generator in $\mathcal{A}$, then the execution of Push Irreducible Component$(a, \mathcal{A}, \text{Expression})$ will update (by Update index) $\nu(a)$ to $\nu(a)+1$ and $\nu(\emptyset)$ to $\nu(\emptyset)-1$. The total time used in the procedure Update indexis $O(nw \log w)$, if a binary search tree is used.

We then consider the complexity of ønding the generators of the irreducible components. It is important to avoid ønding the cliques of the irreducible component solely from the vertex-set and the adjacency-matrix since this problem is NP-complete (Rose 1970).

In $O(n)$ time a generator can be returned and deleted from a data structure for a generating class $\mathcal{C}$, where for each vertex v we have a double linked list of double linked references to generators $c \in \mathcal{C}$ containing the vertex v. In this adjacency matrix for a hypergraph, an element for a generator in the double linked list of generators containing a vertex is also an element in a double linked list of vertices contained in the generator such that when a generator is deleted from the list of generators of one vertex, then the generator can also in linear time in the number of vertices in the generator be deleted from the lists of all other vertices of the generator.

Such a structure can be constructed in $O(nm)$ time, or more precisely $O(m')$ time. To determine the generating class $\mathcal{A}_{\mathrm{i}}$, we for each vertex $w \in a_{(i)}$ have to return (and delete) generators containing w, where we visit the sets $a_{(i)}$ ordered according to the decompositions. We only have to visit a total of n vertices. A total of m generators has to be returned, so the generating classes $\mathcal{A}_{\mathrm{i}}$ can be found in a total time of $O(nm)$. Analogously the sets $\mathcal{S}_{\mathrm{i}}$ can be found in a total time of $O(nw)$.

For the non-decomposable components we have to remove identical sets and sets subsets of other generators from $\mathcal{A}_{\mathrm{i}} \cup \{\mathrm{C}(\mathrm{v}_{\mathrm{i}})\} \cup \mathcal{S}_{\mathrm{i}}$. Examples can be constructed such that the number of generators in a generating class $\mathcal{A}_{\mathrm{i}}$ for a non-decomposable component is proportional to m and such that the number of sets in $\mathcal{S}_{\mathrm{i}}$ is also proportional to m (and w) Removing generators subsets of other generators simply by visiting all generators of $\mathcal{A}_{\mathrm{i}}$ for each separator in $\mathcal{S}_{\mathrm{i}}$ and checking for subset gives the complexity $O(nmw)$. But this situation is rare.

3.5 Hypergraphs

3.5.1 Identiøcation of Conformal Hypergraphs

Tarjan & Yannakakis (1984) give an algorithm for testing that a hypergraph is acyclic in $O(n+m')$. Thus we can decide from the generating class in linear time whether the hypergraph is (conformal and) decomposable. Ten conditions for a hypergraph to be acyclic are given in Beeri et al. (1983).

Checking whether a hypergraph is conformal is harder.

In Berge (1973) the following theorem is given: A hypergraph $\mathcal{C}$ is conformal, if and only if for each triple e_1, e_2 and e_3 of edges of $\mathcal{C}$ there is an edge f of $\mathcal{C}$ such that

$$(e_1 \cap e_2) \cup (e_1 \cap e_3) \cup (e_2 \cap e_3) \subseteq f. \tag{3.14}$$

This condition can be checked in $O(nm^4)$. But a worst case complexity of $O(n^2m^2)$ can be achieved by the following theorem:

Theorem 6 Consider the graph $G = (V(G), E(G))$ and the generating class $\mathcal{C}$, where $\{\alpha, \beta\} \in E(G)$, if $\{\alpha, \beta\} \subseteq c$ for some $c \in \mathcal{C}$. $\mathcal{C}$ are then the cliques of G, if and only if

$$\forall\ v \in V(G) \text{ and } a \in \mathcal{C} \ \ \exists\ \mathrm{b} \in \mathcal{C} : \ \ \{\mathrm{v}\} \cup (\partial \mathrm{v} \cap \mathrm{a}) \subseteq \mathrm{b}. \tag{3.15}$$

Proof Assume that the elements of $\mathcal{C}$ are the cliques of G. Then for any vertex v and any generator a the subset $\{v\} \cup (\partial v \cap a)$ is complete and thus contained in a clique b of G.

To show that the condition is suŒcient to ensure that the generating class is conformal, assume that

$$\forall\ v \in V(G) \text{ and } a \in \mathcal{C} \ \ \exists\ \mathrm{b} \in \mathcal{C} : \ \ \{\mathrm{v}\} \cup (\partial \mathrm{v} \cap \mathrm{a}) \subseteq \mathrm{b}. \tag{3.16}$$

Let $a = \{\alpha_1, \alpha_2, \cdots, \alpha_l\}$ be a clique of G. We then show that there is a $c \in \mathcal{C}$ such that $a \subseteq c$. Then $a = c$, and $\mathcal{C}$ must then be the cliques of G. Let $b_0 = \emptyset$ and for $i = 1, 2, \cdots, l$ let $b_i = \{\alpha_i\} \cup b_{i-1}$. Assume that there is a c_{i-1} such that $b_{i-1} \subseteq c_{i-1}$. We then have

$$b_i = \{\alpha_i\} \cup b_{i-1} = \{\alpha_i\} \cup (\partial\alpha_i \cap b_{i-1}) \subseteq \{\alpha_i\} \cup (\partial\alpha_i \cap c_{i-1}), \tag{3.17}$$

since $b_{i-1} \subseteq \partial\alpha_i$.

Since there is a $c_i \in \mathcal{C}$ such that $\{\alpha_i\} \cup (\partial\alpha_i \cap c_{i-1}) \subseteq c_i$, and since there is a $c_0 \in \mathcal{C}$ such that $b_0 \subseteq c_0$, we have for $i = 1, \cdots, l$ that there is a $c_i \in \mathcal{C}$ such that $b_i \subseteq c_i$. Since $b_l = a$ there is a $c_l \in \mathcal{C}$ such that $a \subseteq c_l$. This theorem is also given in Badsberg (1986). □

The worst case complexity $O(n^2m^2)$ is frustrating since the decomposition of both graphical and hierarchical can be performed in $O(ne + n^2m)$, i.e., only a third degree polynomium. But the average complexity of the procedure is much better. The procedure can be implemented as follows. For each vertex v, divide $\mathcal{C}$ into three classes: A class $\mathcal{A}$ with generators containing vertices from ∂v but not the vertex v, $\mathcal{B}$ with generators containing v, and ønally a class $\mathcal{D}$ containing all other generators. For generators in $\mathcal{B}$ and $\mathcal{D}$ there is nothing to check, and for generators a in $\mathcal{A}$, the generator b such that $\{v\} \cup (\partial v \cap a) \subseteq b$ must be found in $\mathcal{B}$. If for each vertex and each generator to check we only have to visit a very few generators of $\mathcal{B}$ before ønding one fulølling the condition, then we might expect a complexity of $O(n^2m)$, which is probably close to the average complexity for conformal hypergraphs, and which have been veriøed by empirical studies. If the condition is not fulølled after only visiting a very few vertices, i.e., the hypergraph is not conformal, then we may expect a complexity as low as $\Omega(nm)$, also veriøed by empirical studies.

(If all the classes $\mathcal{A}$(v) and $\mathcal{B}$(v) are found before checking is started, then they can be found in a total time of $O(nm)$ by visiting the generators one by one, and then update $\mathcal{B}$(v) or $\mathcal{A}$(v) according to whether v is in the generator or in the boundary of the generator respectively.)

3.5.2 Decomposition of the Non-Conformal Hypergraphs

The algorithm of Tarjan (1985) can be modiøed to decompose non-conformal hypergraphs. We form the 2-section graph of the hypergraph and then apply the algorithm of Tarjan (1985) on the 2-section graph. By the following lemma we know that the separators of the hypergraph $\mathcal{C} = (\mathrm{V(C)} \subseteq \Delta, \mathcal{C})$ is a subset of the separators of the 2-section graph $G^{\mathcal{C}}$ of the hypergraph:

Lemma 7 If a hypergraph $\mathcal{C}$ is decomposable with respect to a subset d of an edge of the hypergraph then the 2-section graph of the hypergraph is decomposable with respect to the set d.

Proof This is a straightforward checking of deønitions. Since $d \subseteq c \in \mathcal{C}$, d is a subset of an edge of the hypergraph, d is a complete subgraph in the 2-section graph $G^{\mathcal{C}}$. By the deønition of decomposition of a hypergraph the set d separates the hypergraph into two subgraphs both containing at least one vertex not in d. Thus d is a complete separator of the 2-section graph. □

For each separator found of the 2-section graph we test whether this separator also decomposes the hypergraph. If it does so, then we decompose the hypergraph. A hypergraph $\mathcal{C} = (\mathrm{V}(\mathcal{C}), \mathcal{C})$ with 2-section graph $G^{\mathcal{C}} = (V(G^{\mathcal{C}}), E(G^{\mathcal{C}}))$ can be decomposed into primes, irreducible subgraphs by the following algorithm:

- Find a minimal ordering π of $G^{\mathcal{C}}$ and compute $C(v_i)$ for each vertex v_i.
- Repeat the following step for each vertex v_i in increasing order with respect to π: Let A be the vertex set of the connected component of $G^{\mathcal{C}}_{V(G^{\mathcal{C}})\setminus C(v_i)}$ containing v_i, and let $B = V(G^{\mathcal{C}}) \setminus (C(v_i) \cup A)$. If $C(v_i)$ is contained in a generator of $\mathcal{C}$, and thus complete in $G^{\mathcal{C}}$, and $B \neq \emptyset$, decompose $G^{\mathcal{C}}$ into $G' = G^{\mathcal{C}}_{A\cup C(v_i)}$ and $G'' = G^{\mathcal{C}}_{B\cup C(v_i)}$, separated by $C(v_i)$. Discard G' as a prime, and replace $G^{\mathcal{C}}$ by G''.

We clearly only perform valid decompositions, but have to verify that the components found cannot be decomposed further:

Lemma 8 If the hypergraph $\mathcal{C}$ contains a clique separator, some decomposition step will be successful.

Proof If the hypergraph has a separator according to which the hypergraph is decomposable, then the separator can be chosen to be minimal. By the above lemma, the separator will also be a complete separator of the 2-section graph of the hypergraph, and the separator will be a minimal separator in the 2-section graph. Lemma (1) (lemma 2 of Tarjan (1985)) states that if the 2-section graph has a minimal separator, then the algorithm of Tarjan (1985) will visit the separator, and thus a minimal

```
Index-Hypergraph(𝒞, Δ, ∂, C, π⁻¹)
    local a, b, d, 𝒜, i, u;
    Expression ::= Null;
    d ::= Δ; b ::= d; i ::= 1;
    while b ≠ ∅
    {
        while ¬Complete(C(π⁻¹(i)))
            i ::= i + 1;
        u ::= π⁻¹(i);
        if i = |Δ| or π⁻¹(|Δ|) ∈ C(u)
        then {
            Push Irreducible Component(d, 𝒞, Expression);
            b ::= ∅
        } else {
            if Subset of a generator(C(u), 𝒞)
            then {
                if Subset of a generator(C(u) ∪ {u}, 𝒞)
                then {
                    j ::= i + 1;
                    while C(π⁻¹(j)) ∪ {π⁻¹(j)} ⊆ C(u) and j < |Δ|
                        j ::= j + 1;
                    a ::= {u} ∪ C(u) \ C(π⁻¹(j − 1));
                    i ::= j − 1;
                    u ::= π⁻¹(i);
                    Update index(a ∪ C(u), 1, Expression);
                    Update index(∅, −1, Expression)
                } else {
                    a ::= Find Connected Component(d \ C(u), u, ∂);
                    𝒜 ::= Find Restricted GC(a ∪ C(u), C(u), 𝒞);
                    Push Irreducible Component(a ∪ C(u), 𝒜, Expression) };
                Update index(C(u), −1, Expression);
                Update index(∅, 1, Expression);
                d ::= d \ a;
                𝒞 ::= Find Restricted GC(d, C(u), 𝒞);
                b ::= d \ C(u) };
            i ::= i + 1 } }
return(Expression);
```

Figur 3.4: Index of Hypergraphs.

separator of the hypergraph can be found by applying the algorithm of Tarjan (1985) on the 2-section graph of the hypergraph. □

Lemma 9 The components resulting from applying the algorithm cannot be decomposed further.

Proof Lemma (2) (theorem 2 of Tarjan (1985)) states that the parts, into which the algorithm is decomposing the 2-section graph, cannot be decomposed further. When we decompose the hypergraph, we can get components where the 2-section graphs of the components can be decomposed further. In the decomposition of the hypergraph we have skipped separators of the 2-section graph that are not valid separators of the hypergraph. These separators are skipped because they are not subsets of edges of the hypergraph. In the components resulting of decomposing the hypergraph these separators will neither be a subset of edges of the hypergraphs of the components, and thus the components cannot be decomposed further. □

Theorem 7 The decomposition algorithm for hypergraphs is correct.

Proof By lemma (8) the algorithm will visit all minimal separators of the hypergraph, and by lemma (9) the components resulting of applying the algorithm cannot be decomposed further. □

This theorem is also given in Badsberg (1986).

In this section we store the generating class in a linked list, a simpler structure than used in the previous section. We then get a worse average complexity, but the same worst case complexity for ønding the generators of irreducible components. The cliques of an irreducible component can be found by the function Find Restricted GC $(d, C(v_i), \mathcal{C})$ in $O(nm)$ time from the generating class of the hypergraph to decompose: If $\mathcal{C}$ is decomposable with respect to $C(v_i)$ and d is the vertex-set of one of the components resulting from the decomposition, then the generating class restricted to d can be found as follows. We visit all generators of $\mathcal{C}$, and if the visited generator is a subset of d, then it is a generator of the restricted generating class. If not the resulting generator only contains the generator d, then the separator $C(v_i)$ is also a generator of the irreducible component. No removal of any generator that is a subset of another is necessary.

Since by construction $a \cup C(u)$ and $d \backslash a$ are separated by $C(u)$, then $a \cup C(u)$ and $d \backslash a$ form a decomposition of $\mathcal{C}$ with respect to $C(u)$, if $C(u)$ is a subset of some generator of $\mathcal{C}$. This can be checked in $O(nm)$ time by the function Subset of a generator$(C(u), \mathcal{C})$. (If the adjacency matrix of the previous section is used, we only have to visit generators containing one of the vertices of $C(u)$ to perform this test.)

If the hypergraph is decomposable with respect to $C(u)$, then in $O(nm)$ time the generating classes $\mathcal{A}$ and $\mathcal{C}$ of the two components are found by the function Find Restricted GC.

We thus get a total complexity $O(ne + n^2 + nm \min(n, m))$ of the algorithm for decomposing a hypergraph.

The algorithm for conformal graphs can easily be modiøed to handle also non-conformal graphs: The lines 13 to 38 are only performed, if $C(u)$ is a subset of some generator of $\mathcal{C}$. Line 13, if $C(u) \subseteq \partial u$, is replaced by if (Conformal and $C(u) \subseteq \partial u$) or ($\neg$ Conformal and Subset of a generator($C(u) \cup \{u\}, \mathcal{C}$)).

If v_i is a perfect vertex, then decomposition with respect to a minimal separator can be achieved by the following theorem:

Theorem 8 Let v_i be the vertex with the smallest order such that $C(v_i)$ is complete. If $C(v_i) \cup \{v_i\} \subseteq c$, $c \in \mathcal{C}$, then v_i is a perfect vertex in the subgraph of $\mathcal{C}$ induced by $\{v_i, \cdots, v_n\}$.

Proof The proof is analogous to that of Theorem 3. □

3.6 Empirical studies

The following tables shows the computing times (in seconds) for decomposing the models (i.e., generating classes) of table 3.5 by respectively, (A), the algorithm HiModel of Geng (1989) and, (B), by the algorithms presented in this chapter.

Each computing time, *time*, is found as the mean of the computing times of 48 decompositions of the relevant model with a given size n, the dimension of the model. The 48 decompositions are performed in 4 runs on 4 identical machines (SPARC-machines, Sun 4-25 with 20 MB RAM) with 12 decompositions performed in each run. The 12 decompositions of the same model of size n in the same run are not performed in succession, but with the decompositions of the other models in between. Only the computing time for decomposing the model is measured, i.e., the reported computing time does not include computing time for generating models, for writing results and for reducing the expression. The algorithm of this chapter also decides whether the non-acyclic hypergraphs also are non-conformal.

In the upper half of the tables the number n of vertices, the dimensions of the models, are ranging from 16 to 128[2]. In the lower half of the tables the number n of vertices are ranging from 64 to 2048, and only the algorithms of the current chapter are applied. Because of the wider range a vertex cannot be stored in an unsigned char, but has to be stored as an integer, and these computing times of the lower half of the table are thus not directly comparable with those of the upper half of the table.

Because of the use of øxed sized arrays in HiModel, the use of HiModel is limited to the upper half of the table. The available memory did not allow for more than 64 resulting irreducible components with a maximum of 128 generators in each of the 64 generating classes for the resulting irreducible components. Thus some computing times are not available for HiModel in the upper half of the table.

The set-operations are for set with fewer than 32 vertices performed in constant time. Also for many sets of size 128 many operations are performed in the same time, but in sets with several hundreds vertices the size of the sets must be taken into

[2] The scale of the dimension is exponential. In this printing of the chapter the 1., 3., 5., 7. and 9. lines from the original data are omitted to make each table øt onto one page, and thus the regression coeŒcients may not be reproduced exactly.

Model $\mathcal{D}_1$: $[\ [1]\ [2]\cdots[n]\ ]$,

Model $\mathcal{D}_2$: $[\ [1,2,3,\cdots,n-1,n]\ ]$,

Model $\mathcal{D}_3$: $[\ [1,2,3]\ [2,3,4]\cdots[n-2,n-1,n]\ ]$,

Model $\mathcal{D}_4$: $[\ [1,2,3,\cdots,n-2,n-1][2,3,4,\cdots,n-1,n]\ ]$,

Model $\mathcal{G}_1$: $[\ [1,2,3]\ [2,3,4]\ \cdots[n-4,n-3,n-2]\ [n-3,n-1][n-2,n][n-1,n]\ ]$,

Model $\mathcal{G}_2$: $[\ [1,2]\ [2,3][1,4][3,4]\ [4,5][3,6][5,6]\cdots[n-2,n-1][n-3,n][n-1,n]\ ]$,

Model $\mathcal{G}_3$: $[\ [1,2][2,3][3,4][1,4]\ [4,5][5,6][6,7][4,7]\cdots$
$[n-5,n-4][n-4,n-3][n-3,n-2][n-5,n-2]$
$[n-2,n-1][n-1,n][n,1][n-2,1]\ ]$,

Model $\mathcal{G}_4$: $[\ [2,3,4,5][1,6][5,7][6,7,10][7,8][8,9][9,10]$
$[10,11,12,13][1,14][13,15][10,13,14][15,18][15,16][16,17][17,18]$
$[18,19,20,21][1,22][21,23][22,23,26][23,24][24,25][25,26]$
$[26,27,28,29][1,30][29,31][26,29,30][31,34][31,32][32,33][33,34]\cdots$
$[n-23,n-22,n-21,n-20][1,n-19][n-20,n-18][n-19,n-18,n-15]$
$[n-18,n-17][n-17,n-16][n-16,n-15]$
$[n-15,n-14,n-13,n-12][1,n-11][n-12,n-10][n-15,n-12,n-11]$
$[n-10,n-7][n-10,n-9][n-9,n-8][n-8,n-7]$
$[n-7,n-6,n-5,n-4][1,n-3][n-4,n-2][n-3,n-2,2][n-2,n-1][n-1,n][n,2]\ ]$,

Model $\mathcal{H}_1$: $[\ [1,2]\ [1,3][2,3]\ [2,4][3,4]\cdots[n-2,n][n-1,n]\ ]$,

Model $\mathcal{H}_2$: $[\ [1,2,3]\ [2,3,4]\ \cdots[n-3,n-2,n-1]\ [n-2,n][n-1,n]\ ]$,

Model $\mathcal{H}_3$: $[\ [1,2][1,3]\ [2,3,4]\ [3,4,5]\cdots[n-2,n-1,n]\ ]$,

Model $\mathcal{H}_4$: $[\ [1,2,3][1,4][2,5][3,6][4,5][5,6][4,6][4,7][5,8][6,9]$
$[7,8,9][7,10][8,11][9,12][10,11][11,12][10,12][10,13][11,14][12,15]\cdots$
$[n-8,n-7,n-6][n-8,n-5][n-7,n-4][n-6,n-3][n-5,n-4]$
$[n-4,n-3][n-5,n-3][n-5,n-2][n-4,n-1][n-3,n]\ \ [n-2,n-1,n]\ ]$.

Figur 3.5: Models for Empirical study.

	Model $\mathcal{D}_1$		Model $\mathcal{D}_2$		Model $\mathcal{D}_3$		Model $\mathcal{D}_4$	
Dimension	A	B	A	B	A	B	A	B
16	0.0054	0.0054	0.0015	0.0017	0.0096	0.0075	0.0017	0.0025
20	0.0071	0.0073	0.0008	0.0017	0.0173	0.0096	0.0010	0.0033
24	0.0067	0.0104	0.0015	0.0029	0.0269	0.0133	0.0013	0.0033
28	0.0100	0.0123	0.0008	0.0035	0.0429	0.0152	0.0010	0.0052
36	0.0150	0.0187	0.0013	0.0035	0.0910	0.0223	0.0025	0.0056
40	0.0183	0.0225	0.0010	0.0029	0.1227	0.0271	0.0017	0.0069
44	0.0217	0.0244	0.0012	0.0042	0.1656	0.0315	0.0013	0.0077
48	0.0256	0.0285	0.0019	0.0035	0.2144	0.0354	0.0021	0.0083
52	0.0292	0.0321	0.0010	0.0046	0.2731	0.0404	0.0008	0.0092
60	0.0385	0.0402	0.0010	0.0040	0.4248	0.0521	0.0021	0.0090
64	0.0437	0.0446	0.0010	0.0052	0.5196	0.0581	0.0013	0.0104
68	0.0483	0.0500	0.0013	0.0044	0.6281	0.0635	0.0025	0.0110
76	0.0615	0.0604	0.0006	0.0081	0.8867	0.0771	0.0019	0.0110
84	0.0733	0.0721	0.0021	0.0050	1.2083	0.0904	0.0021	0.0129
92	0.0885	0.0837	0.0015	0.0079	1.6050	0.1054	0.0010	0.0144
100	0.1052	0.0971	0.0019	0.0075	2.0804	0.1235	0.0002	0.0165
108	0.1223	0.1104	0.0023	0.0083	2.6481	0.1415	0.0010	0.0175
116	0.1406	0.1271	0.0015	0.0079	3.3094	0.1608	0.0023	0.0175
128	0.1723	0.1504	0.0015	0.0085	4.5121	0.1942	0.0027	0.0196
β	1.7316	1.5869	0.3502	0.8754	2.9857	1.5711	-0.0642	0.9694
SE.	0.0445	0.0147	0.1325	0.0637	0.0183	0.0186	0.1796	0.0421
Degree	2	2	1	1	4	2	0	1
64		0.0515		0.0046		0.0621		0.0104
92		0.0965		0.0065		0.1198		0.0127
128		0.1779		0.0083		0.2158		0.0190
180		0.3383		0.0121		0.4121		0.0277
256		0.6610		0.0165		0.8031		0.0390
364		1.3340		0.0260		1.6133		0.0560
512		2.6460		0.0373		3.1829		0.0840
724		5.3485		0.0525		6.3173		0.1202
1024		10.6352		0.0771		12.7210		0.1804
1448		21.0246		0.1013		25.2065		0.2340
2048		42.4327		0.1494		50.4946		0.3788
β		1.9515		1.0208		1.9431		1.0528
SE.		0.0128		0.0127		0.0121		0.0156
Degree		2		1		2		1

Figur 3.6: Computing times for Decomposable Graphs.

	Model $\mathcal{G}_1$		Model $\mathcal{G}_2$			Model $\mathcal{G}_3$			Model $\mathcal{G}_4$	
Dimension	A	B	A	B	n	A	B	n	A	B
16	0.018	0.024	0.025	0.022	15	0.014	0.021	9	0.005	0.009
20	0.033	0.031	0.040	0.034	18	0.023	0.027	17	0.014	0.023
24	0.055	0.044	0.063	0.043	21	0.031	0.032	17	0.015	0.023
28	0.088	0.054	0.091	0.057	27	0.055	0.047	25	0.036	0.039
36	0.186	0.080	0.174	0.084	33	0.093	0.063	33	0.069	0.060
40	0.254	0.095	0.229	0.099	39	0.145	0.083	33	0.070	0.059
44	0.344	0.112	0.297	0.116	42	0.179	0.092	41	0.128	0.086
48	0.451	0.131	0.374	0.136	45	0.214	0.103	41	0.127	0.083
52	0.577	0.149	0.465	0.155	51	0.303	0.127	49	0.212	0.113
60	0.903	0.189	0.689	0.199	57	0.415	0.153	57	0.333	0.146
64	1.099	0.210	0.823	0.224	63	0.550	0.181	57	0.333	0.147
68	1.318	0.236	0.975	0.247	66	0.628	0.198	65	0.489	0.182
76	1.860	0.288	1.332	0.304	75	0.907	0.246	73	0.708	0.221
84	2.550	0.343	1.768	0.365	81	1.130	0.283	81	0.989	0.268
92	3.409	0.407	-	0.431	90	1.534	0.341	89	1.359	0.316
100	4.405	0.473	-	0.504	99	-	0.404	97	1.803	0.370
108	5.607	0.545	-	0.581	105	-	0.449	105	2.357	0.425
116	7.043	0.620	-	0.666	114	-	0.524	113	3.026	0.488
128	9.652	0.746	-	0.803	126	-	0.632	121	3.862	0.552
β	3.042	1.668	2.590	1.696		2.600	1.601		2.650	1.602
SE.	0.012	0.014	0.023	0.015		0.028	0.018		0.059	0.014
Degree	4	2	3	2		3	2		4	2
64		0.223		0.238	63		0.195	57		0.154
92		0.429		0.458	90		0.370	89		0.344
128		0.794		0.851	126		0.691	121		0.600
180		1.540		1.679	177		1.339	177		1.255
256		3.066		3.428	255		2.795	249		2.455
364		6.313		7.089	363		5.809	361		5.269
512		13.061		14.775	510		12.045	505		10.848
724		27.883		32.017	723		26.021	721		24.101
1024		60.796		69.922	1023		57.434	1017		52.524
1448		132.513		156.452	1446		129.428	1441		119.029
2048		293.934		359.220	2046		305.671	2041		281.590
β		2.078		2.115			2.112			2.099
SE.		0.028		0.029			0.033			0.035
Degree		3		3			3			3

Figur 3.7: Computing times for Graphical Graphs, Conformal Hypergraphs.

	Model $\mathcal{H}_1$		Model $\mathcal{H}_2$		Model $\mathcal{H}_3$			Model $\mathcal{H}_4$	
Dimension	A	B	A	B	A	B	n	A	B
16	0.107	0.016	0.017	0.018	0.018	0.015	15	0.019	0.021
20	0.208	0.025	0.033	0.022	0.033	0.020	21	0.040	0.034
24	0.363	0.029	0.056	0.029	0.055	0.025	27	0.066	0.049
28	0.580	0.037	0.088	0.035	0.089	0.030	27	0.066	0.049
36	1.254	0.051	0.185	0.052	0.185	0.042	39	0.150	0.088
40	1.742	0.060	0.256	0.059	0.254	0.049	39	0.150	0.088
44	2.339	0.068	0.342	0.068	0.337	0.055	45	0.210	0.111
48	3.069	0.077	0.450	0.076	0.435	0.061	51	0.284	0.136
52	3.929	0.087	0.577	0.086	0.556	0.069	51	0.283	0.137
60	6.126	0.107	0.904	0.108	0.858	0.084	63	0.476	0.196
64	7.483	0.120	1.099	0.119	1.041	0.094	63	0.475	0.195
68	-	0.131	1.319	0.130	1.253	0.101	69	0.600	0.228
76	-	0.155	1.863	0.156	1.756	0.120	75	0.744	0.263
84	-	0.182	2.556	0.184	2.381	0.139	87	-	0.342
92	-	0.209	3.414	0.213	3.145	0.160	93	-	0.384
100	-	0.240	4.411	0.245	4.061	0.181	99	-	0.429
108	-	0.274	5.614	0.280	5.144	0.206	111	-	0.526
116	-	0.306	7.056	0.316	6.412	0.230	117	-	0.579
128	-	0.362	9.666	0.378	8.688	0.268	129	-	0.693
β	3.066	1.449	3.048	1.472	2.980	1.378		2.269	1.626
SE.	0.006	0.014	0.010	0.016	0.010	0.013		0.020	0.014
Degree	3	2	4	2	4	2		3	2
64		0.127		0.121		0.095	63		0.212
92		0.227		0.220		0.168	93		0.408
128		0.392		0.395		0.288	129		0.741
180		0.710		0.747		0.519	183		1.422
256		1.343		1.491		0.969	255		2.675
364		2.574		3.157		1.848	363		5.195
512		4.925		6.679		3.503	513		10.336
724		9.589		14.658		6.841	723		20.347
1024		18.972		33.121		13.333	1023		41.524
1448		37.434		77.331		26.144	1449		82.245
2048		74.319		186.706		51.601	2049		160.768
β		1.850		2.125		1.829			1.922
SE.		0.020		0.046		0.021			0.015
Degree		2		3		2			2

Figur 3.8: Computing times for Non-Conformal Hypergraphs.

consideration. Thus if the tables are used to estimate the complexity of the algorithms, then the complexity found for HiModel should be multiplied by a factor n. This is not necessary for the algorithms of this chapter, since the range of n used to estimate the complexity is wide enough to reveal that set operations are not performed in constant time. (Also the algorithm of this chapter explicit visits the vertices, and is not implemented by the use of only intersections and tests for subsets as HiModel.)

In the tables with computing times the number β is the regression coeŒcient with standard error in the regression of $\log(time)$ onto $\log(n)$. In plots of $\log(time)$ against $\log(n)$ a progressive relation, increasing slope, was revealed for most of the 15 types of models. This can be explained by the fact that the computing time is a polynomium in the dimension of the model, where also lower degree terms are present.

The line marked *degree* shows the necessary degree of a polynomium øtting the data, i.e., if the computing time is described by a polynomium of degree *degree*, then the coeŒcient of time raised to the highest degree is signiøcant, but if the computing time is described by a polynomium of degree $degree + 1$, then the coeŒcient of time raised to the highest degree is not signiøcant.

First observe, that except for the very simple models $\mathcal{D}_2$ and $\mathcal{D}_4$, models with only a very few generators, the algorithms of this current chapter are superior to the algorithm HiModel.

ıGraphical modelsȷ, the algorithm of this chapter: A 2. degree polynomium is not suŒcient to describe variation. In regression analyses with weights the reciprocal the square of the computing time the coeŒcients of a cubic term is signiøcant. For two of the models the coeŒcient of a quartic term also is signiøcant, but the regression coeŒcient is negative.

When comparing the computing times of the models $\mathcal{G}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$ one may wonder why the later models are must easier to handle. At model $\mathcal{H}_3$ the check of conformality fails at the very ørst vertex visit, whereas the check ørst fails at the last visited vertex in model $\mathcal{H}_2$, and never fails at model $\mathcal{G}_1$. This explains why a 3. degree polynomium is needed at models $\mathcal{G}_1$ and $\mathcal{H}_2$. In model $\mathcal{H}_2$ and $\mathcal{H}_3$ the 2-section graphs are decomposable, and the minimal ordering can then be found by Maximum Cardinality Search, whereas the slower algorithm Lex M has to be used in model $\mathcal{G}_1$.

Final remarks

If the answers to the following questions are positive, then non-decomposable graphs and non-conformal hypergraphs can be decomposed in $O(ne + nm)$ time:

- Can the index, i.e., the list of vertex sets of complete subgraphs and separators, be reduced faster than $O(nw \log w)$? The algorithm Restricted Maximum Cardinality Search on Hypergraphs cannot be used, since several diœerent sets may be exhausted in the same step.

- Can the list of separators intersecting a non-decomposable component be reduced faster than $O(nwm)$, a total time of $O(nwm)$ for all the non-decomposable components, or can the graph be decomposed such that this reduction is not necessary? The algorithm Restricted Maximum Cardinality Search on Hypergraphs cannot be used, since several diœerent sets may be exhausted in the same step.

- Can it be checked faster than $O(nm)$, a total time of $O(nwm)$ for all separators, whether a separator of the 2-section graph is also a separator of the hypergraph? Adjacency matrices give a fast implementation, but maintain a worst case complexity of $O(nwm)$.

Kapitel 4

Algorithms for Collapsing Log-Linear Models onto the Smallest Set Containing a Given Set

Abstract: Asmussen & Edwards (1983) deøned necessary and suŒcient conditions for collapsibility of hierarchical log linear models for a multidimensional contingency table. In this paper two algorithms are presented. For graphical models and hierarchical models respectively we give an algorithm to ønd the smallest set containing a given set and onto which the model is collapsible.

Key Words: Collapsibility, Decomposable, Graphical and Hierarchical Log-Linear Models.

4.1 Introduction

The problem we address in this chapter is as follows: given a graphical or a hierarchical log-linear model $\mathcal{M}$ and a subset of the factors of the model, what is the minimal set containing the given subset, such that $\mathcal{M}$ is collapsible onto the set?

This set can be found by the algorithm HiModel of Geng (1989). This algorithm has a complexity of at least $O(nm^3)$, with m the number of generators of the model and n the number of variables.

In Madigan & Mosurski (1990) an algorithm solving the problem in $O(m')$ time for decomposable models only is considered. m' is the total size of the generating class of the model. They show that an algorithm for selectively reducing an acyclic hypergraph given by Tarjan & Yannakakis (1984), in context of answering queries in relation databases, can be used to solve the problem here considered for decomposable models.

Here we present an algorithm for solving the problem for graphical models in $O(ne)$ time. By a simple modiøcation the algorithm can also handle non-graphical models in $O(ne + nmw)$. Here e is the number of edges in the 2-section graph of the model and w is the number of decompositions with respect to minimal separators.

Madigan & Mosurski (1990) write that their algorithm has applications in expert systems: In the context of expert systems, by reducing a probability inÆuence network onto only the relevant nodes, the algorithms reduce the required computation and simplify interpretation. This chapter gives eŒcient algorithms for an extended class of problems. The algorithm of this chapter is also crucial in computation of an adjustment of the number of degrees of freedom in tests between two log-linear models, see Badsberg (1991).

The problem solved in this chapter can be given a pure discrete mathematical formulation: Given a graph (or hypergraph) and a subset of the set of vertices of the graph, what is the smallest set A containing a given set Δ such that the boundary of each connected component of the graph induced by the complement of the set A is complete (contained in an edge of the hypergraph)?

4.2 Notation and Preliminaries

Contingency Tables and Log-Linear Models

See the corresponding subsection of chapter 2, but read Δ as $\Delta \cup \Gamma$.

Graphs

See the corresponding subsection of chapter 2, but read Δ as $\Delta \cup \Gamma$:

We shall consider graphs with two types of vertices. $\Delta \subseteq V(G)$ is the set of vertices, we shall call *marked*, and $\Gamma = V(G) \setminus \Delta$ contains the *unmarked* vertices.

The decomposition formed by a and b is strong, if at least one of the three conditions $a \cap b \subseteq \Delta$, $a \setminus b \subseteq \Gamma$, $b \setminus a \subseteq \Gamma$ holds.

Hypergraphs

See the corresponding subsection of chapter 2, but read Δ as $\Delta \cup \Gamma$.

The Fill-In Graph

Add the following to the corresponding subsection of chapter 2:

The set $\partial v_i \cap \{v_{i+1}, \cdots, v_n\}$ of vertices following v_i and adjacent to v_i is called the monotone adjacency set. By $C(v_i)$ we will denote the monotone adjacency set in the øll-in graph:

$$C(v_i) = \partial_\pi v_i \cap \{v_{i+1}, \cdots, v_n\}. \tag{4.1}$$

Through this chapter m will denote the number of generators in the generating class $\mathcal{M}$, m' will denote the total size of the generating class, i.e., the sum $m' = \sum_{i=1,\cdots,m} |c_i|$, where $\mathcal{M} = \{\mathrm{c_i, i = 1, \cdots, m}\}$, n the number of vertices in the 2-section graph $G^{\mathcal{H}} = (V(G^{\mathcal{H}}), E(G^{\mathcal{H}}))$ for the hypergraph $\mathcal{H} = (\mathrm{V}(\mathcal{H}), \mathcal{M})$, e the number of edges in the 2-section graph, and e' will denote the number of edges in a øll-in graph of the 2-section graph. w will denote the number of decompositions with respect to minimal separators.

Collapsibility and Decompositions of Models

See the corresponding subsection of chapter 2, but read Δ as $\Delta \cup \Gamma$. The two following corollaries follows from Theorem 1:

Corollary 2 If a and b form a decomposition of a hypergraph $\mathcal{H} = (\mathrm{V}(\mathcal{H}), \mathcal{M})$ relative to hierarchical log-linear model $\mathcal{M}$ (or a decomposition of the 2-section graph $G = (V(G), E(G))$ relative to a graphical log-linear model $\mathcal{M}$), then $\mathcal{M}$ can be collapsed onto a.

Proof This is a part of Theorem 2.1 of Asmussen & Edwards (1983). □

Lemma 10 A hierarchical log-linear model $\mathcal{M}$ is collapsible onto A, if and only if the boundary of every connected component of A^c is contained in a generator of $\mathcal{M}$.

Proof This is Theorem 2.3 of Asmussen & Edwards (1983). □

4.3 Collapsing Graphical Models

The problem we address in this section is as follows: given a graphical log-linear model $\mathcal{M}$ with interaction graph $G = (V(G), E(G))$ and the set Δ, a subset of $V(G) = \Gamma \cup \Delta$, what is the minimal set A, $\Delta \subseteq A \subseteq \Gamma \cup \Delta$, such that $\mathcal{M}$ is collapsed onto A?

Madigan & Mosurski (1990) show that the algorithm for selectively reducing an acyclic hypergraph given in Tarjan & Yannakakis (1984) can be used to ønd such sets for decomposable models. The algorithm runs in $O(m')$ time. Here we give an $O(ne')$ time algorithm for solving the problem also for non-decomposable graphical models. The algorithm we are going to present is based on the following algorithm for decomposing a graph $G = (V(G), E(G))$ into primes, irreducible subgraphs:

- Find a minimal ordering π of G and compute $C(v_i)$ for each vertex v_i.
- Repeat the following step for each vertex v_i in increasing order with respect to π: Let A be the vertex set of the connected component of $G_{V(G)\setminus C(v_i)}$ containing v_i, and let $B = V(G) \setminus (C(v_i) \cup A)$. If $C(v_i)$ is complete in G and $B \neq \emptyset$, decompose G into $G' = G_{A\cup C(v_i)}$ and $G'' = G_{B\cup C(v_i)}$, separated by $C(v_i)$. Replace G by G'', and discard G' as a prime.

Lemma 11 If G contains a clique separator, some decomposition step will be successful.

Proof This is Lemma 2 of Tarjan (1985). □

Lemma 12 The decomposition algorithm is correct.

Proof This is Theorem 2 of Tarjan (1985). □

We modify the algorithm of Tarjan (1985) to perform decompositions eliminating as many vertices of $V(G) \setminus \Delta$ as possible.

With the graph $G = (V, E)$, $V = \Gamma \cup \Delta$, we for some vertex ω not contained in V associate a graph G^ω,

$$G^\omega = (V^\omega, E^\omega) = (V \cup \{\omega\}, E \cup \{\{\delta, \omega\} |\ \delta \in \Delta\}),$$

with, for each vertex v_i, the monotone adjacency set $C^\omega(v_i)$,

$$C^\omega(v_i) = \partial^\omega_\pi v_i \cap \{v_{i+1}, \cdots, v_n, \omega\}, \tag{4.2}$$

in the øll-in graph F^ω of G^ω for the vertex elimination ordering π. The boundary of a vertex v in the øll-in graph F^ω is denoted by ∂^ω_π.

Now consider the following algorithm:

i) For the graph $G = (V, E)$, $V = \Gamma \cup \Delta$, create the graph G^ω.

ii) Find by the algorithm Lex M of Rose et al. (1976) a minimal vertex elimination ordering $\pi = [v_1, \cdots, v_{n+1}]$ for G^ω with the extra vertex $\omega = v_{n+1}$ given the highest ordering and such that $\Delta = \{v_k, \cdots, v_n\}$. This is possible by Lemma (13). Determine also the sets $C^\omega(v_i)$, which by Lemma (14) for vertices $v_i \in V \setminus \Delta$ are equal to the sets $C(v_i)$ of the graph G.

iii) According to the minimal vertex elimination ordering, for each vertex v_i such that $C(v_i)$ is complete and $i < k$, eliminate the connected component of the remaining graph containing v_i.

Step iii) corresponds to the algorithm of Tarjan (1985), but here terminated before any decomposition removing vertices of Δ. Modiøed versions of the algorithm of Tarjan (1985) are investigated in Leimer (1993) and chapter 3. The algorithm in these two papers is optimized in the sense that the graph is decomposed into precisely the irreducible components and only decompositions with respect to minimal separators are performed.

Lemma 13 If vertex $\omega = v_{n+1}$ is given the highest ordering in the algorithm Lex M of Rose et al. (1976), then all the vertices of Δ are ordered before any other vertex by Lex M.

Proof Any vertex can be ordered ørst with the highest ordering $n + 1$. If the vertex $\omega = v_{n+1}$ is ordered ørst in Lex M, then $\omega = v_{n+1}$ is inserted in the labels of precisely the vertices of Δ, and thus these vertices will be ordered before any vertex not in Δ. □

Lemma 14 No separator $C^{\omega}(v_i)$ will contain ω for any vertex $v_i \in V \setminus \Delta$, and $C^{\omega}(v_i) = C(v_i)$ for these vertices.

Proof There are no øll-in edges to the vertex ω given the highest ordering: Assume there is no edge between v and ω, but there is a øll-in edge between v and ω. Then there is a path $\omega = v_{(1)}, v_{(2)}, \cdots, v_{(k)} = v$ such that $\pi(v_{(i)}) < \min\{\pi(\omega), \pi(v)\}$ for $i = 2, \cdots, k-1$. The vertex $v_{(2)}$ adjacent to ω, $\{v_{(2)}, \omega\} \in E(G^{\omega})$, has then a smaller ordering than v, $\pi(v_{(2)}) < \pi(v)$, contradicting Lemma (13).

By the deønition of the øll-in it is obvious that there is a øll-in edge between two vertices of G if and only if there is a øll-in edge between the two same vertices of G^{ω}.

No vertex $v_i \in V \setminus \Delta$ is connected to ω, and thus

$$C^{\omega}(v_i) = \partial_{\pi}^{\omega} v_i \cap \{v_{i+1}, \cdots, v_n, \omega\} = \partial_{\pi} v_i \cap \{v_{i+1}, \cdots, v_n\} = C(v_i) \tag{4.3}$$

for $v_i \in V \setminus \Delta$. □

We need two additional lemmas to prove that the result of the algorithm cannot be reduced further.

Lemma 15

i) Let a and b form a decomposition of G^{ω}. We can assume that $\omega \in a$ without the loss of generality. If $a \setminus b \neq \{\omega\}$, then $a \setminus \{\omega\}$ and $b \setminus \{\omega\}$ will form a strong decomposition of G.

ii) Let a and b form a strong decomposition of G; then

   1) $a \cup \{\omega\}$ and b will then form a decomposition of G^{ω} if $b \setminus a \subseteq \Gamma$ and

   2) $a \cup \{\omega\}$ and $b \cup \{\omega\}$ will then form a decomposition of G^{ω} if $a \cap b \subseteq \Delta$.

Proof This is Lemma 1 of Leimer (1989). □

Lemma 16 If a graphical log-linear model $\mathcal{M}$ with interaction graph $G = (V, E)$ can be collapsed onto A_1 and A_2, where $A_1, A_2 \subseteq V$, then $\mathcal{M}$ can be collapsed onto $A_1 \cap A_2$.

Proof This is Lemma 3.4 of Madigan & Mosurski (1990). □

Theorem 9 Given a graphical log-linear model $\mathcal{M}$ with interaction graph $G = (V, E)$ and a subset Δ of V, then the above algorithm gives the minimal set A, for $\Delta \subseteq A \subseteq V$, such that $\mathcal{M}$ can be collapsed onto A. Furthermore, for any set C with $\Delta \subseteq C \subseteq V$, where $\mathcal{M}$ can be collapsed onto C, we have $A \subseteq C$.

Proof By Lemma (15) we have a decomposition formed by a and b of G with $b \setminus a \subseteq V \setminus \Delta$, if and only if we have a decomposition formed by $a \cup \{\omega\}$ and b of G^{ω}. Thus we might as well run the elimination process on G. By Lemma (14) no separator $C^{\omega}(v_i)$ will contain ω for any vertex $v_i \in V \setminus \Delta$.

When the algorithm terminates, we have a subgraph G_A of G such that G by decompositions can be reduced to G_A and such that $\Delta \subseteq A$. From Corollary (2) we then have that $\mathcal{M}$ can be collapsed onto the vertex set of the subgraph produced in each elimination step of the algorithm, and thus also onto the ønal set A produced by the algorithm.

Lemma (12) states that the algorithm of the Tarjan (1985) is correct, i.e., when our algorithm terminates there is no decomposition formed by a and b of the remaining graph such that $b \setminus a \subseteq V \setminus \Delta$. One may then wonder if a minimal vertex elimination ordering of G without the restrictions imposed by ω would reveal other decompositions eliminating only vertices in $V \setminus \Delta$. But such a decomposition would also eliminate only vertices in $V \setminus \Delta$ from G^ω, and thus the decomposition would have been found by the vertex elimination on G^ω.

Now let us assume the set A is not minimal, that is, there exists a set $A' \subset A$ such that $\mathcal{M}$ is collapsible onto A'. If $\mathcal{M}$ is collapsible onto A', then the boundary of each connected component B_l of the graph induced by $A \setminus A'$ is complete by Lemma (10). Then each connected component B_l can be eliminated by a decomposition formed by $A \setminus B_l$ and $B_l \cup \partial B_l$ such that $B_l \subseteq A \setminus \Delta$ contrary to the above observation.

For the ønal assertion, we have from Lemma (16) that $\mathcal{M}$ is collapsible onto $A \cap C$. But we have just shown that $\mathcal{M}$ is not collapsible onto any set $A' \subset A$ and so, $A \cap C = A$. □

4.4 Collapsing Hierarchical Models

A hypergraph $\mathcal{H} = (\mathrm{V}, \mathcal{M})$ with 2-section graph $G^{\mathcal{H}} = (V(G^{\mathcal{H}}), E(G^{\mathcal{H}}))$ can be decomposed into primes, irreducible subgraphs by the following algorithm:

- Denote the 2-section graph $G^{\mathcal{H}}$ of $\mathcal{H}$ by $G = (V(G), E(G))$, and ønd a minimal ordering π of G and compute $C(v_i)$ for each vertex v_i.

- Repeat the following step for each vertex v_i in increasing order with respect to π: Let A be the vertex set of the connected component of $G_{V \setminus C(v_i)}$ containing v_i, and let $B = V \setminus (C(v_i) \cup A)$. If $C(v_i)$ is contained in a generator of $\mathcal{H}$, and thus complete in G, and $B \neq \emptyset$, decompose G into $G' = G_{A \cup C(v_i)}$ and $G'' = G_{B \cup C(v_i)}$, separated by $C(v_i)$. Discard G' as a prime, and replace G by G''.

Lemma 17 The above decomposition algorithm for hypergraphs is correct.

Proof This is Theorem 5 of chapter 3. □

The algorithm of the previous section for collapsing graphical models is easily modiøed to handle also non-graphical hierarchical models. We form the 2-section graph $G^{\mathcal{H}} = (V(G^{\mathcal{H}}), E(G^{\mathcal{H}}))$ of the hypergraph $\mathcal{H} = (\mathrm{V}, \mathcal{M})$ for the model hierarchical $\mathcal{M}$, and in the third step of the algorithm also control that the separator is contained in a generator of the model (no separator will contain ω):

iii) According to the minimal vertex elimination ordering, for each vertex v_i such that $C(v_i) = C^{\omega}(v_i)$ is contained in a generator of $\mathcal{M}$, and thus is complete, and $i < k$, eliminate the connected component of the remaining graph containing v_i.

Since we for each separator have to check that the separator is contained in a generator of the model, the algorithm runs in $O(ne' + nwm)$ time. For correctness of the decomposition algorithm and details of the complexity of the algorithm, see chapter 3.

Theorem 10 Given a hierarchical log-linear model $\mathcal{M}$ with hypergraph $\mathcal{H} = (\mathrm{V}, \mathcal{M})$ and 2-section graph $G^{\mathcal{H}} = (V, E)$ and a subset Δ of V, then the above algorithm gives the minimal set A, for $\Delta \subseteq A \subseteq V$, such that $\mathcal{M}$ can be collapsed onto A. Furthermore, for any set C with $\Delta \subseteq C \subseteq V$, where $\mathcal{M}$ can be collapsed onto C, we have $A \subseteq C$.

Proof The proof is analogously to that of Theorem 1. Statements like ı$C(v_i)$ is completeȷ have to be replaced by ı$C(v_i)$ is complete and contained in a generator of $\mathcal{M}$ȷ. We have to use Theorem 5 of chapter 3 instead of Theorem 2 of Tarjan (1985). Details are left to the reader. $\square$

Del II

The program CoCo

Kapitel 5

Model Search in Contingency Tables by CoCo

Abstract

CoCo[1], a highly advanced program for analysis of complete and incomplete contingency tables, is presented.

In the paper a short presentation of CoCo is given. Incremental search by backward elimination and forward selection and the global search procedure from Edwards & Havrnek (1985) is considered.

By incremental search a single minimal acceptable model is identiøed. By the principles of weak acceptance and weak rejection the class of minimal acceptable models are found in the global search procedure. In CoCo each of the model searches can be done by a single command, or CoCo can be guided through the search in a highly user controlled model selection.

5.1 CoCo

CoCo works especially eœectively on graphical models, and some of the commands in CoCo are designed to handle graphical models. Graphical models are for contingency tables log-linear interaction models that can be represented by a simple undirected graph with as many vertices as the table has dimension (Darroch et al. 1980). Further more all these models can be given an interpretation in terms of conditional independence and the interpretation can be read directly oœ the graph in the form of a Markov property. The class of graphical models is a proper subclass of the hierarchical models, but the class strictly contains the decomposable models.

CoCo is a program designed to estimate and test in large contingency tables. By using graph-theoretical results (Rose et al. 1976) (Tarjan & Yannakakis 1984) and

[1] For the time being, CoCo runs on Macintosh and on workstation and PC under respectively Unix and DOS. CoCo can be obtained free of charge by anonymous ftp over internet from ftp.iesd.auc.dk (130.225.48.4). Or CoCo is available on disks (3.5j or 5.25j High density).

(Tarjan 1985) the hierarchical log-linear interaction models are decomposed. For non-decomposable models the IPS-algorithm is not used on the full table, but only on the non-decomposable atoms. If one model is tested against another and the two models have common decompositions, the test can be partitioned in tests on smaller tables (Goodman 1971). Collapsibility (Asmussen & Edwards 1983) of tests is used.

Estimation in and tests between models with an unlimited number of factors (80 factors on a PC) can then be performed, if the largest non-decomposable atom in associated interaction graphs has no more than 14 binary vertices (24 on workstations).

In sparse tables an adjusted number of degrees of freedom is computed. Besides incomplete tables and tables with incomplete observations and latent variables can be handled and exact tests between any two nested decomposable models can be computed by Monte Carlo methods, see the Guide to CoCo, (Badsberg 1991). For ordinal variables the Goodman and Kruskal's Gamma coeŒcient can be computed.

To make the use a little easier some data selection is possible. Observed counts, estimated counts and probabilities and residual (observed minus estimated counts, adjusted residuals, standardized residuals, Freeman-Tukey, etc.) can be listed, printed in tables, described and plotted.

CoCo can be loaded into New S (Becker et al. 1988) and XLISP-STAT (Tierney 1991). CoCo is linked truly together with the other system. All CoCo commands can in XLISP-STAT be performed as calls to LISP functions. Tables and models can be send to the CoCo-object in New S or XLISP-STAT and functions for returning statistics and residuals from the CoCo-object for, e.g., high-resolution plotting are provided. In XLISP-STAT a model search is possible by interaction with pictures of interaction graphs for models.

5.2 Incremental search

The incremental search in CoCo is performed by forward inclusion (Dempster 1972) and backward elimination (Wermuth 1976b) of edges in independence graphs for models. By incremental search a single minimal acceptable model (or few models) is identiøed.

The independence graph is a simple undirected graph with as many vertices as the table has dimension. A vertex for each variable. Two vertices are adjacent in the independence graph for the model unless the two variables are conditionally independent given the other variables.

In the backward elimination all pairs of vertices adjacent in a speciøc model are tested conditionally independent. The non-signiøcant edge with the largest P-value (or smallest value of the Akaike Information Criterion, AIC, (Akaike 1974) is then removed. This stepwise edge elimination process can be performed recursively until no more edges can be removed, i.e., all edges in the resulting model are signiøcant with respect to the selected level of signiøcance. Or at each step based on a list of edges sorted according to, e.g., P-values a set of edges can be eliminated.

The principle of coherence can be applied: if a model is rejected, then all its submodels are also rejected. In the backward elimination this means that if elimination of an edge is rejected at one step, the edge is not eligible for elimination at subsequent steps.

In the selection, the decision of acceptance of a model can be based on the deviance, Pearson's chi-square χ^2 or the power divergence $2nI^\lambda$ (Read & Cressie 1988). The number of degrees of freedom can be adjusted for non-estimable parameters or simulated exact tests can be applied. In the incremental search the AIC values or a the Bayesian Information Criterion (Schwarz 1978) can be used, and for ordinal variables the Goodman and Kruskal's gamma coeŒcient can be applied.

In the backward elimination the tests can either be made locally, i.e. nested tests: the model is tested against the previously accepted model, or the model can be tested against a øxed base model.

In the forward selection in turn, each edge not present in the independence graph for the current model is added to the model and the current model is then tested against the resulting model. Or the resulting model is tested against a speciøc base model. All edges found to be signiøcant are added to the current model and the resulting model is inserted in the list of models in CoCo, see belove.

In CoCo the backward elimination and the forward selection can be started from any initial model. Steps of the backward elimination and the forward selection can be mixed together in any order, and between each step any set of edges or interaction terms can be added or eliminated.

The backward elimination from the saturated model has the advantage that the tests are not performed under models later to be found rejected. In the forward selection from the uniform model the initial tests will be performed under models likely to be rejected. But in the forward selection the initial tests often will be collapsible to small tables, and much larger tables can be handled.

The incremental search is done either among graphical models or the search can be restricted to decomposable models. Diœerent edge elimination orderings might result in the same model, and since a sequence of decomposable models diœering with one edge always exists between any two nested decomposable models and when restricting to decomposable models, the IPS-algorithm is not used, restricting to decomposable models might speed up the model search. If exact tests are applied then restricting to decomposable models is also useful since the exact tests are only available in CoCo for decomposable models.

Also speciøc edges can be øxed so they are not removed in the backward elimination or added in the forward selection.

Headlong backward. CoCo can be linked together with XLISP-STAT (Tierney 1991). The model-search can then be done by graphical interaction with a plot of the independence graph. In this environment a recursive backward elimination strategy, where the ørst edge found to be non-signiøcant is eliminated, is implemented in the LISP language. In each step of this backward elimination, edges with P-value less than a given limit (e.g. 1% or 5%) is rejected, and is, when the principle of weak rejection is applied, not considered in sub-sequential steps. The ørst edge found in each step with P-value greater than a second limit (e.g. 20%) is removed. Visited edges in the current step with a found P-value between the two limits and all not visited eligible edges in the current step are eligible in the next step. This gives a faster elimination than a backward elimination where all eligible edges in each step have to be visited to ønd the least signiøcant edge.

Not to favour the removal of edges with, e.g., a low lexicographical order, the edges in this headlong backward elimination are visited in a random order. Hence several calls to the procedure might give diœerent results, and a sample of models with all edges signiøcant to a selected level of signiøcance can be generated.

The list of models. The resulting models from each cycle of the backward elimination and the resulting model from the forward selection are added to a linked list of models in CoCo. This list of models in CoCo has been found to be very useful. Besides models generated by the incremental search procedures the model list also contains all models read into CoCo. The list of models helps keeping track of the search. The models in the list can be edited: edges or interaction-terms added or removed, the models tested against each other, øtted values in models tabulated, plotted, etc.

The forward selection and the backward elimination can also be used to add or remove 1st order interaction terms from non-graphical log-linear interaction models.

The program BIFROST by Greve et. all is built on top of CoCo for model search by backward elimination in block recursive models (Kreiner 1989).

5.3 Global search

By the principles of weak acceptance and weak rejection, coherence, the search space of all hierarchical models on the contingency table are divided into the two sets of models: the class of minimal acceptable models and the class of maximal rejected models. Hence after using the 'Fast Procedure for Models Search' by (Edwards & Havrnek 1985, Edwards & Havrnek 1987) any model can be labelled as accepted or rejected.

The models (or sub-models of a speciøc base model) are classiøed into two regions $\mathcal{A}$ and $\mathcal{R}$: weakly accepted models and weakly rejected models. Weakly accepted models are models that include an accepted model, and weakly rejected models are models that are included in a rejected model. In turn, a boundary below the weakly accepted models and a boundary above the weakly rejected models, the R-dual $D_R(\mathcal{A})$ and the A-dual $D_A(\mathcal{R})$ respectively, are øtted.

In CoCo the global search can be started with the accepted class containing the saturated model and the class of rejected models containing the model with only main-eœects, or any models can be entered into the two classes. The search stops when either all models in $D_A(\mathcal{R}) \setminus \mathcal{A}$ are accepted or all models in $D_R(\mathcal{A}) \setminus \mathcal{R}$ are rejected.

Graphical search. In the graphical search, for each step either all not classiøed models in the graphical A-dual are øtted or all not classiøed models in the graphical R-dual are øtted.

$D^g_A(\mathcal{R})$: The graphical A-dual $D^g_A(\mathcal{R})$ of the rejected models is the minimal models in the class of models that adds at least one edge to each rejected model.

$D^g_R(\mathcal{A})$: The graphical R-dual $D^g_R(\mathcal{A})$ of the accepted models is the maximal models in the class of models that omits at least one edge from each accepted model.

Restricting graphical search to decomposable models. By a naive approach the selection can be restricted to decomposable models: non-decomposable models in the duals to øt are simply ignored. The search is then performed among only decomposable models. After a search among only decomposable models a search among graphical models with the result of the decomposable search as a starting-point can be performed. Since non-decomposable models in the graphical dual are just ignored in the decomposable search, there might still be graphical and decomposable models to øt, when both the set $D^g_A(\mathcal{R}) \setminus \mathcal{A}$ of not classiøed models in the graphical A-dual and the set $D^g_R(\mathcal{A}) \setminus \mathcal{R}$ of not classiøed models in the graphical R-dual contains no more decomposable models. So the decomposable search should be followed by a graphical search. One might consider adding edges to or removing edges from the non-decomposable models in the graphical duals in the models selection among decomposable models. Since a lot of computing time in the search module is used to ønd duals, and not, as it might be expected, in computing the tests and øtting the models by the IPS-algorithm, the usefulness of the decomposable search is doubtful (unless exact tests are to be used).

Hierarchical Search. In the hierarchical search we for each step either øt all not classiøed models in the hierarchical A-dual or all not classiøed models in the hierarchical R-dual.

$D_A(\mathcal{R})$: The hierarchical A-dual $D_A(\mathcal{R})$ of the rejected models is the minimal models in the class of models that adds at least one interaction term to each rejected model.

$D_R(\mathcal{A})$: The hierarchical R-dual $D_R(\mathcal{A})$ of the accepted models is the maximal models in the class of models that sets at least one interaction term in accepted models to zero.

If the hierarchical search follows a graphical search, the regions found in the graphical search are used as a starting point in the hierarchical search, else the search is started with the saturated model accepted and the model with only main eœects rejected, or any models can be entered into the two classes.

Since the determination of the duals can involve a lot of computation, various strategies have been considered. Which of the following three strategies is the fastest depends on the situation. They need not give the same result since, e.g., weakly rejected models may be accepted. The strategies can be applied in both the graphical search and the hierarchical search.

Fit smallest dual: For each step (classiøcation of all not already classiøed models in a dual into the accepted and rejected regions) in the search, both duals are found, and the dual with fewest not classiøed models is selected for the next step. This will often minimize the number of øtted models, but sometimes a lot of computing time is used for ønding large duals that are not used.

Use rough dual sizes: With this strategy rough estimates of the dual sizes are found after each step, and the dual with the smallest rough estimate of dual size is selected for the next step. The rough estimate of dual size is the product of the number of edges (or interaction-terms) in the model (or the dual representation of the model) over all models in the class $\mathcal{A}$ or $\mathcal{R}$. Since it sometimes avoids ønding large duals that are not used, this can be eœective. By this strategy the complete search in (Edwards & Havrnek 1985) on a 6 dimensional table is done by CoCo in less than 900 milliseconds on a SPARCstation.

Alternating øtting: The R-dual and A-dual can in turn be classiøed. In a few examples, where alternating øtting the R-dual and the A-dual is optimal, this is the fastest. But often the strategy will start to select the largest dual, and then the strategy will be the slowest.

Restricted search. In CoCo the global search among graphical models can, besides being restricted to decomposable models, also be restricted to models containing a speciøc model and/or to models contained in a speciøc model. Some edges are øxed in the model and some are edges øxed out. Also the hierarchical search can be restricted to models containing a speciøc model and/or to sub-models of a speciøc model.

L i t t e r a t u r

Akaike, H. (1974). A new look at the statistical model identiøcation, IEEE Transactions on Automatic Control 19: 716˘723.

Andersen, A. H. (1974). Multidimensional contingency tables, Scandinavian Journal of Statistics 1: 115˘127.

Anglin, D. G. & Oldford, R. W. (1993). Modelling response models in software, Preliminary Papers of the Fourth International Workshop on Artiøcial Intelligence and Statistics, pp. 483˘494.

Asmussen, S. & Edwards, D. (1983). Collapsibility and response variables in contingency tables, Biometrika 70: 567˘578.

Badsberg, J. H. (1986). Kontingenstabeller ˘ implementation og kompleksitet af algoritmer for estimation og test i store kontingenstabeller, Masters thesis, Aalborg University.

Badsberg, J. H. (1991). A guide to CoCo, Research Report R 91˘43, Department of Mathematics and Computer Science, Aalborg University, Denmark.

Badsberg, J. H. (1992). Model search in contingency table by CoCo, in Y. Dodge & J. Whittaker (eds), Computational Statistics, Vol. 1, Physica Verlag: Heidelberg, pp. 251˘256. COMPSTAT 1992, Nechatel.

Becker, R. A., Chambers, J. M. & Wilks, A. R. (1988). The New S Language, A Programming Environment for Data Analysis and Graphics, Wadsworth & Brooks/Cole Advanced Books & Software, Paciøc Grove, California.

Beeri, C., Fagin, R., Maier, D. & Yannakakis, M. (1983). On the desirability of acyclic database schemes, J. Assoc. Comput. Mach. 30: 479˘513.

Berge, C. (1973). Graphs and Hypergraphs, North-Holland, Amsterdam.

Chambers, J. M. & Hastie, T. J. (1991). Statistical models in S, Wadsworth & Brooks/Cole Advanced Books & Software, Paciøc Grove, California.

Christensen, R. (1990). Log-Linear Models, Springer-Verlag.

Darroch, J. N. & Ratcliœ, D. (1972). Generalized iterative scaling for log-linear models, Annals of Mathematical Statistics 43: 1470˘1480.

Darroch, J. N., Lauritzen, S. L. & Speed, T. P. (1980). Markov øelds and log-linear interaction models for contingency tables, Annals of Statistics 8: 522˘539.

Dawid, A. P. & Skene, A. M. (1979). Maximum likelihood estimation of observer error-rates using the em algorithm, Journal of the Royal Statistical Society, Series C 28(1): 20˘28.

Deming, W. E. & Stephan, F. F. (1940). On a least squares adjustment of a sampled frequency table when the expected marginal totals are known, Annals of Mathematical Statistics 11: 427˘444.

Dempster, A. P. (1972). Covariance selection, Biometrika 28: 157˘175.

Edwards, D. (1989). A guide to MIM version 1.7, Research Report 12, Statistical Research Unit, University of Copenhagen. A Guide to MIM version 2.0, 1991.

Edwards, D. (1990). Hierarchical interaction models (with discussion), Journal of the Royal Statistical Society, Series B 52: 3˘20 and 51˘72.

Edwards, D. (1993). Some computational aspects of graphical model selection, in J. Antoch (ed.), Computational Aspects of Model Choice, Springer-Verlag, pp. 187˘210.

Edwards, D. & Havrnek, T. (1985). A fast procedure for model search in multidimensional contingency tables, Biometrika 72: 339˘351.

Edwards, D. & Havrnek, T. (1987). A fast model selection procedure for large families of models, Journal of the American Statistical Association 82: 205˘211.

Fienberg, S. E. (1970). An iterative procedure for estimation in contingency tables, Annals of Mathematical Statistics 41: 907˘917.

Frydenberg, M. (1989). The chain graph Markov property, Scandinavian Journal of Statistics 17: 333˘353.

Frydenberg, M. (1990). Marginalization and collapsibility in graphical interaction models, Annals of Statistics 18: 790˘805.

Frydenberg, M. & Lauritzen, S. L. (1989). Decomposition of maximum likelihood in mixed interaction models, Biometrika 76: 539˘555.

Fuchs, C. (1982). Maximum likelihood estimation and model selection in contingency tables with missing data, Journal of the American Statistical Association 77(378): 270˘278.

Geng, Z. (1989). Decomposability and collapsibility for log-linear models, Algorithm AS 244, Appl. Statist. 38(1): 189˘197.

Geng, Z., Asano, C., Ichimura, M. & Kimura, H. (1993). Decomposability and collapsibility for contingency tables with missing data, Algorithm AS 294, Appl. Statist. 42(1): 548˘554.

Geng, Z. & Asano, C. (1988). Recursive procedures for hierarchical loglinear models on high-dimensional contingency tables, J. Japanese Soc. Comp. Statist. 1(1): 17˘26.

Gilchrist, R. & Scallan, A. (1988). Funigirls: A prototype functional programming language for the analysis of generalized linear models, in D. Edwards & N. E. Raun (eds), COMPSTAT, Vol. 1, Physica Verlag: Heidelberg, pp. 207˘212. COMPSTAT 1988, Copenhagen.

Goodman, L. A. (1971). Partitioning of chi-square, analysis of marginal contingency tables, and estimation of expected frequencies in multidimensional contingency tables, Journal of the American Statistical Association 66: 339˘344.

Haberman, S. J. (1972). Log-linear øt for contingency tables, Algorithm AS 51, Appl. Statist. 21: 218˘227.

Haberman, S. J. (1974). Log-linear Models For Frequency Data, Univ. of Chicago Press, Chicago, Illinois.

Hitz, M. & Hudec, M. (1994). Applying the object oriented paradigm to statistical computing, in R. Dutter & W. Grossmann (eds), Computational Statistics, Physica Verlag: Heidelberg, pp. 389˘394. COMPSTAT 1994, Vienna.

Huber, P. J. (1994). ıHugeȷ data sets, in R. Dutter & W. Grossmann (eds), Computational Statistics, Physica Verlag: Heidelberg, pp. 3˘13. COMPSTAT 1994, Vienna.

Jensen, F. V. (1988). Junction trees and decomposable hypergraphs, Research report, JUDEX Ltd., Aalborg.

Jensen, F. V. & Jensen, F. (1994). Optimal junction trees, Proceedings of the tenth conference on Uncertainty in Artiøcial Intelligence, Seattle, Washington.

Jensen, F. V., Lauritzen, S. L. & Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computations, Computational Statistics Quarterly 4: 269˘282.

Jirouek, R. (1991). Solution of the marginal problem and decomposable distributions, Kybernetike 27(5): 403˘412.

Jirouek, R. (1994). On the eœective implementation of the iterative proportional øtting procedure, To appear in Computational Statistics and Data Analysis.

Kjrulœ, U. (1990). Graph triangulation ¯ algorithms giving small total state space, Technical Report R 90-09, University of Aalborg, Denmark.

Kjrulœ, U. (1992). Optimal decomposition of probabilistic networks by simulated anneling, Statistics and Computing 2: 7˘17.

Kreiner, S. (1987). Analysis of multidimensional contingency tables by exact conditional tests: Techniques and strategies, Scandinavian Journal of Statistics 14: 97˘112.

Kreiner, S. (1989). User's guide to DIGRAM¯ a program for discrete graphical modelling, Technical Report 89˘10, Statistical Research Unit, University of Copenhagen.

Lauritzen, S. L. (1982). Lectures on Contingency Tables, (3rd edition 1989), Aalborg: University of Aalborg Press, Denmark.

Lauritzen, S. L. (1989). Mixed graphical association models (with discussion), Scandinavian Journal of Statistics 16: 273˘306.

Lauritzen, S. L. (1991). The EM algorithm for graphical association models with missing data, Technical Report R 91˘05, Department of Mathematics and Computer Science, Aalborg University, Denmark.

Lauritzen, S. L. (1993). Graphical association models, DRAFT, Technical Report IR 93˘2001, Department of Mathematics and Computer Science, Aalborg University, Denmark.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N. & Leimer, H.-G. (1990). Independence properties of directed Markov øelds, Networks 20: 491˘505.

Lauritzen, S. L. & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion), Journal of the Royal Statistical Society, Series B 50: 157˘224.

Lauritzen, S. L. & Wermuth, N. (1984). Mixed interaction models, Research report R˘84˘8, Department of Mathematics and Computer Science, Aalborg University, Denmark.

Lauritzen, S. L. & Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative, Annals of Statistics 17: 31˘57.

Lauritzen, S. L., Speed, T. P. & Vijayan, K. (1984). Decomposable graphs and hypergraphs, J. Austral. Math. Soc. (Series A) 36: 12˘29.

Lauritzen, S. L., Thiesson, B. & Spiegelhalter, D. J. (1992). Diagnostic systems created by model selection methods - a case study, Technical report, The university of Aalborg, Institute for Electronic Systems, Department of Mathematics and Computer Science.

Leimer, H.-G. (1989). Triangulated graphs with marked vertices, in L. D. Andersen (ed.), Graph Theory in Memory of G.A. Dirac, Vol. 41, Ann. Discrete Math., pp. 311˘324.

Leimer, H.-G. (1993). Optimal decomposition by clique separators, Discrete Mathematics 113: 90˘123.

Madigan, D. & Mosurski, K. (1990). An extension of the results of Asmussen and Edwards on collapsibility in contingency tables, Biometrika 77: 315˘319.

Malvestuto, F. M. (1989). Computation the maximum-entropy extension of given discrete probability distributions, Computational Statistics and Data Analysis 8: 229˘311.

McDonald, J. A. (1986). Antelope: data analysis with object oriented programming and constraints, Proceeding of 1985 Joint Statistical Meetings, Statistical Computing Section, pp. 1˘10.

McDonald, J. A. & Pedersen, J. (1988). Computing environments for data analysis III: Programming environments, SIAM J. Sci. Stat. Comput. 9(2): 380˘400.

Mehta, C. R. & Patel, N. R. (1983). A network algorithm for performing øsher's exact test in $r \times c$ contingency tables, Journal of the American Statistical Association 78(382): 427˘434.

Morgan, M. & Blumenstein, B. A. (1991). Exact conditional tests for hierarchical models in multidimensional contingency tables, Appl. Statist. 40(3): 435˘442.

Oldford, R. W. (1988). Object-oriented software representations for statistical data, J. of Econometrics 38(3): 227˘246.

Oldford, R. W. & Peters, S. C. (1986). Object-oriented data representations for statistical data, COMPSTAT, Physica Verlag: Heidelberg. COMPSTAT 1986.

Oldford, R. W. & Peters, S. C. (1988). DINDE: Towards more sophisticated software environments for statistics, SIAM Journal on Computing 9(1): 191˘211.

Pateøeld, W. M. (1981). An eŒcient method of generating random R*C tables with given row and column totals, Algorithm AS 159, Appl. Statist. 30: 91˘97.

Read, T. R. C. & Cressie, N. A. C. (1988). Goodness-of øt Statistics for Discrete Multivariate Data, Springer-Verlag, New York.

Rose, D. J. (1970). Triangulated graphs and the elimination process, J. Math. Anal. Appl. 32: 597˘609.

Rose, D. J. (1973). A graph-theoretic study of the numerical solution of sparge positive deønite systems of linear equations, in R. C. Read (ed.), Graph Theory and Computing, Academic Press, New York, pp. 183˘217.

Rose, D. J., Tarjan, R. E. & Lueker, G. S. (1976). Algorithmic aspects of vertex elimination on graphs, SIAM Journal on Computing 5: 266˘283.

Sakamoto, Y. & Akaike, H. (1978). Analysis of cross classiøed data by aic, Ann. Inst. Statist. Math. 30: 185˘197.

Schwarz, G. (1978). Estimating the dimension of a model, Annals of Statistics 6, 2: 461˘464.

Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L. & Cowell, R. G. (1993). Bayesian analysis in expert systems, Statistical Science 8: 219˘247.

Stuetzle, D. G. (1987). Plot windows, Journal of the American Statistical Association 82: 466˘475.

Sundberg, R. (1975). Some results about decomposable (or markov-type) models for multidimensional contingency tables: Distribution of marginals and partitioning of tests, Scandinavian Journal of Statistics 2: 771˘779.

Tarjan, R. E. (1985). Decomposition by clique separators, Discrete Mathematics 55: 221˘232.

Tarjan, R. E. & Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, SIAM Journal on Computing 13: 566˘579.

Tierney, L. (1990). LISP-STAT ¯ An Object-Oriented Environment for Statistical Computing and Dynamic Graphics, Wiley, New York.

Tierney, L. (1991). Generalized linear models in lisp stat, Technical Report 557, School of Statistics, University of Minnesota.

Wedelin, D. (1993). EŒcient algorithms for probabilistic inference, combinatorial optimization and the discovery of causal structure from data, Ph.d. thesis, Department of Computer Sciences, Chalmers University of Technology, University of Gteborg, Gteborg, Sweden.

Wen, W. (1990). Optimal decomposition of belief networks, Proceedings of the Sixth Workshop on Uncertainty in Artiøcial Intelligence, Cambridge, MA, pp. 245˘256.

Wermuth, N. (1976a). Analogies between multiplicative models in contingency tables and covariance selection, Biometrics 32: 95˘108.

Wermuth, N. (1976b). Model search among multiplicative models, Biometrics 32: 253˘264.

Wermuth, N. (1988). Block-recursive linear regressions, Research report 88-1, Berichte zur Stochastik und verwandte Gebeite, University of Mainz.

Wermuth, N. & Lauritzen, S. L. (1983). Graphical and recursive models for contingency tables, Biometrika 70: 537˘552.

Wermuth, N. & Lauritzen, S. L. (1990). On substantive research hypotheses, conditional independence graphs and graphical chain models (with discussion), Journal of the Royal Statistical Society, Series B 52: 21˘72.

Whittaker, J. (1990). Graphical Models in Applied Multivariate Statistics, Wiley, Chichester.

Wilkinson, G. N. & Rogers, C. E. (1979). Symbolic description of factorial models for analysis of variance, Journal of the Royal Statistical Society, Series C 22(3): 392˘399.