Decomposable Graphical Models With a View Towards Outlier Detection and Sparse Tables

Lindskou, Mads

DOI (link to publication from Publisher):
10.54337/aau470863492

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

DECOMPOSABLE GRAPHICAL MODELS WITH A VIEW TOWARDS OUTLIER DETECTION AND SPARSE TABLES

BY
MADS LINDSKOU

DISSERTATION SUBMITTED 2022

AALBORG UNIVERSITY
DENMARK

Decomposable Graphical Models With a View Towards Outlier Detection and Sparse Tables

Ph.D. Dissertation

Mads Lindskou

Dissertation submitted February 27, 2022

Dissertation submitted:	February 27, 2022
PhD supervisor:	Associate Professor Torben Tvedebrink Aalborg University
Assistant PhD supervisors:	Associate Professor Poul Svante Eriksen Aalborg University Professor Niels Morling Aalborg University University of Copenhagen
PhD committee:	Honorary Professor Steffen Lauritzen (chair) University of Copenhagen, Denmark Professor Niels Richard Hansen University of Copenhagen, Denmark Professor Thore Egeland Norwegian University of Life Sciences, Norway
PhD Series:	Faculty of Engineering and Science, Aalborg University
Department:	Department of Mathematical Sciences

ISSN (online): 2446-1636
ISBN (online): 978-87-7573-937-0

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2022

Abstract

This thesis concerns graphical models, mainly with discrete variables, a class of probabilistic models that can be understood pictorially using an interaction graph. Some classical graphical models includes the naive Bayes classifier and the Ising model, where the interaction graph of the former has directed edges, and the latter has undirected edges. At the core of graphical models is the concept of conditional independence; X is conditionally independent of Y given variable Z if by knowing Z, Y is irrelevant for X. Having access to such knowledge typically simplifies the model and hence the interaction graph. For classification problems, the structure of the interaction graph may be of lesser importance, whereas the structure may be essential in problems where the goal is the understand the underlying phenomenon. A large number of different graphical models exists with special structures of the interaction; graphs with both directed and undirected edges, both discrete and continuous variables etc. Common to all these is that the graphical structure must be learned either from data, also referred to as structure learning, or specified through expert knowledge. Given an interaction graph, calculating posterior probabilities, also known as inference, is essential in graphical models. Although both structure learning and inference are, in general, NP-complete problems there exist efficient heuristics for both, and even exact methods for inference. Most notably is the junction tree algorithm.

The thesis consists of two parts, where the first serves as a brief introduction to graphical models and highlights some of the findings in the thesis papers by examples. The second part consists of the main papers A-D. Outlier detection in high-dimensional discrete data is a challenging task, which involves determining observations with small counts in the data. In paper A, a novel outlier detection method based on decomposable graphical models with discrete variables is presented. This method is directly extended in paper B to include continuous variables. Insights from the software complementing paper A revealed that sparsity in the involved tables can be exploited to save both time and memory during inference in graphical models as explained in detail in paper C. Finally, in the pursuit of optimizing memory and infer-

ence time a novel scheme, called unity propagation, is proposed in paper D along with a new smoothing technique, called unity smoothing, for handling inconsistencies between the model and new observations.

Resumé

Denne afhandling omhandler grafiske modeller, hovedsageligt for diskrete variable, hvilket er en klasse a probabilistiske modeller, som kan forstås ved hjælp af en dertilhørende interaktions graf. To klassiske modeller indenfor denne klasse tæller naive Bayes classifier og Ising modellen, hvor interaktions grafen for den førstnævnte indeholder orienterede kanter hvorimod interaktions grafen for den sidstnævne indeholder ikke-orienterede kanter. Kernen i grafiske modeller er konceptet om betinget uafhængighed; X er betinget uafhængig af Y givet Z, hvis informationen om Z gør Y irrelevant for X. Information om betinget uafhængighed simplificerer modellen og dermed interaktions grafen. For klassifikations problemer kan strukturen på interaktions grafen have en mindre betydning, hvorimod strukturen er essentiel i problemer hvor formålet er at forstå det underliggende fænomen som modelleres. Der eksisterer mange typer af grafiske modeller med forskellige interaktions grafer, dvs. grafer med både orienterede og ikke-orienterede kanter og både diskrete og kontinuerte variable osv. Fælles for disse er, at den grafiske struktur skal læres fra data, også kaldet strukturlæring, eller bestemmes ved hjælp af ekspertviden. Givet en interaktionsgraf, er det muligt at beregne posterior sandsynligheder, også kaldet inferens, hvilket er essentiel for grafiske modeller. Selvom både strukturlæring og inferens generelt er NP-komplette problemer, eksisterer der effektive heuristiker og eksakte metoder for inferens. Af særlig interesse er junction tree algoritmen.

Afhandlingen består af to dele, hvor den første fungerer som en kort introduktion til grafiske modeller og eksempler som belyser nogle af resultaterne i afhandlingens artikler. Anden del består af hoved artiklerne A-D Outlier detektion i højdimensionale diskrete data er et kompliceret problem, som involverer at bestemme observationer som er set få gange i data. Der gives en ny metode til outlier detektion i dekomposable grafiske modeller med diskrete variable i artikel A. Denne metode udvides direkte i artikel B for at kunne håndtere kontinuerte variable. Indsigt fra det komplementerende software for artikel A viste, at sparsiteten i de involverede tabeller kan udnyttes

for at spare både tid og hukommelse i forbindelse med inferens i grafiske modeller som forklaret i detaljer i artikel C. Slutteligt, i jagten på at optimere hukommelse og tidsforbrug ved inferens, forelås en ny metode, kaldet unity propagation, i artikel D sammen med en ny smoothing teknik, kaldet unity smoothing, til at håndtere inkonsistenser mellem modellen og nye observationer.

Contents

Preface

This thesis contains the scientific research conducted by me and my collaborators during my PhD study at the Department of Mathematical Sciences at Aalborg University. The work was partially funded by The Section of Forensic Genetics, Department of Forensic Medicine, Faculty of Health and Medical Sciences, University of Copenhagen, Denmark. The research concerns outlier detection and optimizing existing methods for inference in graphical models. Part I introduces graphical models and small examples to facilitate some of the findings in the papers. Part II is a collection of the four research papers:

A: Lindskou, M., Eriksen, P. S., and Tvedebrink, T. *Outlier detection in contingency tables using decomposable graphical models*. Scandinavian Journal of Statistics, 47(2), 347-360.

B: Lindskou, M., Tvedebrink, T., Eriksen, P. S. and Morling, N. *Detecting Outliers in High-dimensional Data with Mixed Variable Types using Conditional Gaussian Regression Models*. Submitted to *arXiv*.

C: Lindskou, M., Tvedebrink, T., Eriksen, P. S., Højsgaard, .S, and Morling, N. *The jti and sparta Packages: Junction Tree Inference using Sparse Tables with a View Towards High Dimensional Graphical Models*. Submitted to *Journal of Statistical Software*.

D: Lindskou, M., Tvedebrink, T., Eriksen, P. S., Højsgaard, .S, and Morling, N. *Unity Smoothing for Handling Inconsistent Evidence in Bayesian Networks and Unity Propagation for Faster Inference*. Submitted to *International Journal of Approximate Reasoning*.

Other work, by the PhD candidate, not included in the thesis counts

- Lindskou, M.. *molic: An R package for multivariate outlier detection in contingency tables*. Journal of Open Source Software, 4(42), 1665.
- Lindskou, M. *Outlier Detection in Categorical Data*. Symposium i Anvendt Statistik 2020.

I wish to thank my supervisor Torben Tvedebrink who have always taken the time to listen to my ideas and motivated me to keep going even in difficult periods. Torben and I have had countless pleasant conversations both regarding the thesis but also on a personal level. The same can be said about my co-supervisor Svante. I have always felt that I was welcome in his office, either for a nerdy talk or just everything and nothing. I have been part of a dream team working with you guys. I also wish to thank my co-supervisor Niels Morling, who has been invaluable in the entire process by always being critical and motivated about my work. Also, a big thank you to Søren Højsgaard, whom I had the pleasure to work with on paper C and D. During my stay a the Department of Computer Science, I had the privilege to work with Thomas Dyhre Nielsen and Kristian G. Olesen on a new method for triangulating non-decomposable graphs. The work is ongoing.

Finally, and most important, a special thanks to my beloved family; Mette, Viktor and Alberte. You guys are the very reason that I have completed my PhD study without severe stress and anxiety. The three of you are my everything ♡.

Mads Lindskou

Aalborg University, February 27, 2022

Part I

Background

Introduction

Graphical models (Lauritzen, 1996, Pearl, 2014, Edwards, 2012) is a family of probability distributions that can be interpreted using directed or undirected graphs, in which each node represents a random variable; an edge (or arc) between two variables indicates a relationship between these. Such a graphical representation is convenient, since, for high-dimensional problems the local interpretation of a subset of variables can be interpreted in the sub-graph induced by these and connected variables to these. In essence, a graphical model encodes for conditional independencies among the variables.

By exploiting the graph-theoretic representation, general algorithms can be used for computing marginal and conditional probabilities. For large enough problems, it may be impossible to compute such probabilities without exploiting the graphical representation. Of particular importance for calculations of probabilities in graphical models is the junction tree algorithm explained in more detail in both paper C and D. By far, the two most popular types of graphical models are Bayesian networks (BNs) and undirected graphical models (also known as Markov random fields, MRFs) which are two of three different types of graphical models considered in this thesis. In paper B, we also consider a type of graphical models called mixed graphical models that contains both discrete and continuous variables.

In the following, a brief exposition of undirected graphical models and BNs is presented in order to put the contributions of the thesis into perspective. The notation in the following may differ slightly from the notation used in the papers, and we strive to keep things simple and non-technical by using toy examples.

1 Undirected Graphs

An undirected graph $G = (V, E)$ is a pair consisting of a set of vertices, V, and a set of edges $E = \{\{u, v\} \mid u, v \in V, u \neq v\}$. A clique is a subset of V, for which all nodes are connected in the induced sub-graph of G (a complete graph). A maximal clique, with respect to inclusion, is a clique, which is not contained in any other cliques. From here, by clique we always mean a maximal clique.

Let A, S and B be disjoint subsets of the vertices in an undirected graph G. Then (A, S, B) forms a (weak) decomposition of G if $V = A \cup S \cup B$, S separates all vertices in A from all vertices in B, and S is complete. The decomposition (A, S, B) decomposes G into two sub-graphs, $G_{A \cup S}$ and $G_{B \cup S}$ defined over the vertices $A \cup S$ and $B \cup S$, respectively. The graph G is said to be decomposable if it can be successively decomposed into sub-graphs defined on its cliques.

2 Graphical Log-Linear Models

An undirected graphical model with discrete variables is, in its essence, a log-linear model with the additional property that conditional independencies can be inferred from an undirected graph called the interaction graph. Consider three binary variables x, y and z, defined over I_x, I_y and I_z respectively where each equals $\{0, 1\}$. Denote by p a generic probability mass function. If it holds that

$$p(x, y, z) = \frac{p(x, z)p(y, z)}{p(z)}, \tag{1}$$

we say that x and y are conditionally independent given z, written $x \perp\!\!\!\perp y \mid z$. It can be shown that if $x \perp\!\!\!\perp y \mid z$, the joint density factorizes as

$$p(x, y, z) = h(x, z)g(y, z) \tag{2}$$

for some functions h and g. This result is also known as the factorization criterion. Using this, the conditional independencies in (1) can be encoded via a hierarchical log-linear model as

$$\log p(x, y, z) = \mu + \mu_x + \mu_y + \mu_z + \mu_{x,z} + \mu_{y,z}, \tag{3}$$

where μ is a constant, and all other terms are parameters that are set to zero if one of the corresponding variables is zero. The term hierarchical means that if a parameter in (3) is set to zero, all its other higher-order relatives are also set to zero. If, for example, $\mu_y = 0$, we must also set $\mu_{y,z} = 0$. In a log-linear model, two variables are conditionally independent of all other variables if they do not appear in any interaction term together. In what follows, we use the model in (3) as a running example and show the connection to decomposable graphical models.

The global Markov property states that for any decomposition (A, S, B) it holds that $A \perp\!\!\!\perp B \mid S$, meaning that all variables in A are conditionally independent of all variables in B given S. Consider now the interaction graph, G, in Figure 1 with cliques $C_1 = \{x,z\}$ and $C_2 = \{y,z\}$ and separator $S = \{z\}$. The triple (C_1,S,C_2) forms a decomposition of G and hence encodes for the same conditional independencies as the log-linear model in (3). In fact, G is decomposable and the model is called a decomposable graphical model. If, however, the edge $\{x,y\}$ is added to G and $\mu_{x,y}$ is added to (3), the model is no longer graphical since the three-way interaction term $\mu_{x,y,z}$ must also be present. That is, not all log-linear models are graphical models. Furthermore, not all graphical models are decomposable. Finally, the cliques are called generators of the hierarchical log-linear model in (3) since they uniquely specify the terms needed in the model.

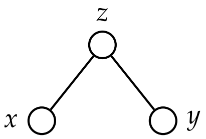


Fig. 1: The interaction graph G.

	z_0		z_1	
	y_0	y_1	y_0	y_1
x_0	125	87	17	5
x_1	67	40	30	9

Table 1: A contingency table of the variables x,y and z.

Consider a sample of $N = 380$ observations from p and denote by $n(x,y,z)$ the number of observations for a specific value of x,y and z. The sample can then be summarized in the contingency table shown in Table 1, where x_0 is an abbreviation for $x = 0$ etc. From the table, we see for example that $n(x_1,y_1,z_1) = 9$. Given z, the rows in Table 1 are approximately proportional and it seems reasonable to state that $x \perp\!\!\!\perp y \mid z$ corresponding to the interaction graph G.

We could just as well have adopted the multinomial scheme, and the likelihood function then takes the form

$$L(p) \propto \prod_{(x,y,z)\in\mathcal{I}} p(x,y,z)^{n(x,y,z)}, \tag{4}$$

where $\mathcal{I} = I_x \times I_y \times I_z$. The maximum likelihood estimates (MLEs) under the constraint that $x \perp\!\!\!\perp y \mid z$ is given by

$$\hat{p}(x,y,z) = N^{-1}\frac{n(x,z)n(y,z)}{n(z)}, \tag{5}$$

where e.g., $n(x,z)$ is the marginal table over x and z. In general, the maximum likelihood estimates is given by a product over the clique tables divided by a product over the separator tables. Let $q(x,y,z)$ be the saturated model, where the edge $\{x,y\}$ is added. That is, q corresponds to a log-linear model where all interaction terms are included and no restrictions are imposed other than $\sum_{(x,y,z)\in\mathcal{I}} q(x,y,z) = 1$. The MLEs are then given by

$$\hat{q}(x,y,z) = N^{-1}n(x,y,z). \tag{6}$$

For high-dimensional contingency tables, possibly with many complex interaction terms, it is difficult a priori to have very precise ideas about the relevant models and where one initially should look for possible conditional independence among the variables (Darroch et al., 1980). As such, the graph representation is important when searching for models. A cornerstone in the thesis is the **R** library **ess** (Lindskou, 2021) that fits decomposable undirected graphical models according to the method described in Deshpande et al. (2013). It is a forward procedure that starts with the independence graph, $(V,\emptyset)$, (which is indeed decomposable) and successively adds the "best edge" in each step. The edges that are allowed to be inserted are completely characterized since the graph is forced to be decomposable and hence the search space is considerably narrowed. Let $G = (V,E)$ be a decomposable graph and assume $\{x,y\} \notin E$. Given two cliques, C_1 and C_2 with separator S, the edge $\{x,y\}$ can be added if, and only if, $x \in C_1$, $y \in C_2$ and both x and y are connected to all vertices in S.

Recall that the entropy of p is given by

$$H(p) = -\sum_{(x,y,z)\in\mathcal{I}} p(x,y,z)\log p(x,y,z).$$

It can be shown that minimizing the Kullback-Leibler (KL) divergence of a given model, p, from the saturated model, $\hat{q}$, is equivalent to maximum likelihood estimation. For decomposable graphical models, Malvestuto (1991) showed that

$$\begin{aligned}\mathrm{KL}(p,\hat{q}) &= H(p(x,y,z)) - H(\hat{q}(x,y,z)) \\ &= H(p(x,z)) + H(p(y,z)) - H(p(z)) - H(\hat{q}(x,y,z)).\end{aligned}$$

Furthermore, this is the same as minimizing the entropy of the given model, p. In each step of **ess**, one seeks to add the edge that maximizes the difference in entropy, which in turn minimizes the entropy of p. One property of **ess** is that it can reuse the entropies in subsequent calculations saving a lot of computational time and memory. However, the difference in entropy does not give us any idea as to when we should stop adding edges. Consider Akaikes Information Criterion defined as

$$\mathrm{AIC} = -2\log\left(\hat{L}\right) + 2k,$$

where $\hat{L}$ is the maximum value of the likelihood function in (4), and k is the number of estimated parameters in the model. Let AIC and AIC$^{'}$ denote Akaikes information criteria for the models $\hat{p}$ and $\hat{q}$, respectively. We strive for AIC $-$ AIC$'>0$. That is

$$\Delta\mathrm{AIC} = 2\log\left(\hat{L}'/\hat{L}\right) + 2(k-k') > 0.$$

The first term of ΔAIC is called the deviance, and it follows that

$$\begin{aligned} D^2 &= 2\sum_{(x,y,z)\in\mathcal{I}} n(x,y,z)\log\frac{n(x,z)n(y,z)}{n(x,y,z)n(z)} \\ &= 2N\left\{H(\hat{p}(x,y,z)) - H(\hat{q}(x,y,z))\right\}. \end{aligned}$$

The difference in AIC then reduces to

$$\Delta\mathrm{AIC} = 2N\left\{H(\hat{p}(x,y,z)) - H(\hat{q}(x,y,z))\right\} + 2(k-k').$$

Finally, this gives us a stopping criteria while preserving the savings in computational time of the entropies. In the running example, it follows that

$$k-k' = |I_z|(|I_x|-1)(|I_y|-1) = |\mathcal{I}| - |\mathcal{I}_{xz}| - |\mathcal{I}_{yz}| + |I_z| = 2,$$

where $\mathcal{I}_{xz} = \mathcal{I}_{yz} = I \times I$. This corresponds to adding the terms $\mu_{x,y}$ and $\mu_{x,y,z}$ to (3) as discussed above. Define by $\mathcal{J} = \{i \in \mathcal{I} \mid i \text{ is observed in data}\}$ the sparse statespace of $\mathcal{I}$ (and similarly for $\mathcal{I}_{xz}$ etc.). In **ess**, the difference in parameters can be calculated based on either $\mathcal{J}$ or $\mathcal{I}$. In the former case, more edges are added to accommodate for associations that would not otherwise be included due to sparsity.

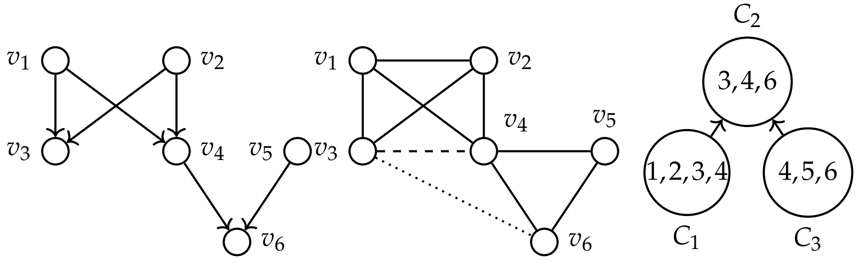
Log-linear regression models are less suited for complex models with many nodes and edges whereas the graphical models enjoy the Markov property enabling easy verification of conditional independence via the interaction graph. Moreover, when interest is in calculating posterior probabilities, the decomposable graphical models are a necessity and many different algorithms have been developed for this very purpose. One algorithm that stands out is the Junction Tree Algorithm explained broadly using a flow-chart in Section 4. This algorithm is also the topic of paper C and D.

Paper A deals with the notion of outliers in decomposable undirected graphical models, which only makes sense when the number of variables and their statespaces have a certain size. However, in the toy example with observations given in Table 1, we may declare the observation (x_0, y_1, z_1) as an outlier since it was only observed five times. The method exploits the decomposition to assign local scores on cliques and separators. Paper B directly extends the work in paper A to allow for a mix of discrete and continuous variables.

3 Bayesian Networks and the Junction Tree Algorithm

Bayesian networks (BNs), also called belief networks, are graphical models where the interaction graph is a directed acyclic graph (DAG). The model is easier to define, but, in general, it requires more effort to determine conditional independencies from a DAG. Consider the DAG G in Figure 2(a) where the set of nodes is given by $\{v_1, v_2, v_3, v_4, v_5, v_6\}$. The joint pmf is then given by

$$p(v_1, v_2, v_3, v_4, v_5, v_6) = p(v_1)p(v_2)p(v_3 \mid v_1, v_2)p(v_4 \mid v_1, v_2)p(v_5 \mid v_4, v_6)p(p_6). \tag{7}$$



(a) (b) (c)

Fig. 2: TBA

That is, the pmf factorizes into conditional probability tables (CPTs) of the nodes given their parents. Computing marginals and other conditionals from (7) can be done simply by marginalization. However, the order in which the variables are marginalized out is crucial. Choosing a particular order may result in a failed attempt to marginalize due to lack of computer memory. This is where the junction tree algorithm comes to rescue. It starts by moralizing, i.e., add an edge between nodes that have a common child and remove the directions. If the resulting graph is not decomposable, cf. Section 1, it must be made so by triangulation, i.e., add fill-in edges until no cycles of length greater than three are present. The moralized graph of G is already decomposable, but assume we are interested in the joint pmf of v_3 and v_6. Because they are not situated in the same clique of the moralized graph, we must extend the moralized graph with the edge $\{v_3, v_6\}$ depicted with a dotted line. This in turn makes the graph non-decomposable and we must add the dashed fill-in edge $\{v_3, v_4\}$. The maximal cliques, $C_1 = \{v_1, v_2, v_3, v_4\}, C_2 = \{v_3, v_4, v_6\}$ and $C_3 = \{v_4, v_5, v_6\}$, can now be located e.g., using maximum cardinality search and used to construct a secondary structure called the junction tree, see Figure 2(c). We choose C_2 as the root of the junction tree, meaning that we send messages towards C_2 such that we can query the joint pmf of v_3 and v_6. The CPTs can now be allocated to the cliques to form the clique potential tables i.e., those table objects that are manipulated via multiplication, division and marginalization during message passing in the junction tree. In paper C we introduce a new object, called sparta, that is efficient when the tables are sparse. A CPT can be allocated to a clique if the child node and all its parent nodes belong to the clique. Notice, that no CPTs belong to clique C_2; we call such cliques for unity cliques and these cliques can be exploited to speed up the message passing, which is the topic of paper D.

When the message passing has finished, each clique potential table encodes for the joint pmf of the variables involved in the clique. Hence, the joint pmf of (v_3, v_6) is easily obtained by marginalizing out v_4 in the potential associated with C_2.

Imagine now that G is a pedigree. That is, v_1 and v_2 are indeed the parents of v_3 and v_4, etc. The genetic information passed on from parent to child is deterministic if we assume no artifacts like mutations, etc. We consider a single DNA marker with the possible alleles a, b and c. Assume that both v_1 and v_2 have the genotype (a, b). Since v_1 and v_2 passes on one of their alleles to v_3 with equal probability, it follows that

$$p(v_3 = x \mid v_1 = (a,b), v_2 = (a,b)) = \begin{cases} 1/4 & \text{if } x = (a,a) \\ 1/4 & \text{if } x = (b,b) \\ 1/2 & \text{if } x = (a,b) \\ 0 & \text{else} \end{cases} \tag{8}$$

irrespectively of the allele frequencies p_a, p_b and p_c. However, to encode the entire table $p(v_3 \mid v_1, v_2)$, the entire statespace must be present. If we make the encoding $1 \mapsto 4$, $1/2 \mapsto 2$, $1/4 \mapsto 1$ (where the number to the left of the arrow is a probability and the number to the right of the arrow is the encoding), and we symbolize a zero with a dot, then $p(v_3 \mid v_1, v_2)$ can be represented as shown in Table 2. The conditional (8) is highlighted with a dashed rectangle. The interesting thing is, that the table is very sparse. If we let n_l be the number of alleles, then the number of heterozygotes is $n_h = \binom{n_l}{2}$, and the number of genotypes is $n_g = n_l + n_h$. Given the parents genotypes, there are either one, two, three or four possible genotypes that the child can inherit. If for example, both parents are homozygous with the same allele, the child must also be homozygous with the same allele. Investigating these four cases, it can be shown that the number of non-zero cells in $p(v_3 \mid v_1, v_2)$

is given by

$$n_z = \begin{cases} n_l^2 + 4n_l n_h & \text{if } n_l = 2 \\ n_l^2 + 4n_l n_h + 3n_h + 4n_h(n_h - 1) & \text{if } n_l \geq 3. \end{cases} \tag{9}$$

The total number of cells is given by $n_t = n_g^3$. Using (9), the total number of cells in Table 2 is 216 of which only 78 non-zero. In Figure 3, a plot of n_l and n_t is shown for 4,5,...,30 alleles, indicating that there is a huge benefit by leveraging sparsity. Let $\phi_{C_1} = p(v_1)p(v_2)p(v_3 \mid v_1, v_2)p(v_4 \mid v_1, v_2)$ be the clique potential for clique C_1. In the following, we analyze the memory needed to store ϕ_{C_1} with dense tables and using the new sparse tables, called `sparta`, proposed in paper C. First, the product $\phi_{23} = p(v_3 \mid v_1, v_2)p(v_4 \mid v_1, v_2)$ contains

$$n_l^2 + 2 \cdot 4n_l n_h + 3 \cdot 3n_h + 4 \cdot 4n_h(n_h - 1) \tag{10}$$

non-zero cells.

	v_3	*aa*	*bb*	*cc*	*ab*	*ac*	*bc*		v_3	*aa*	*bb*	*cc*	*ab*	*ac*	*bc*
v_2	v_1							v_2	v_1						
aa	*aa*	4	.	.	2	2	.	*ab*	*aa*	2	.	.	1	1	.
	bb	.	.	.	.	.	.		*bb*	.	2	.	1	.	1
	cc	.	.	.	.	.	.		*cc*	.	.	.	.	.	.
	ab	.	4	.	2	.	2		*ab*	2	2	.	2	1	1
	ac	.	.	4	.	2	2		*ac*	.	.	2	.	1	1
	bc	.	.	.	.	.	.		*bc*	.	.	2	.	1	1
bb	*aa*	.	.	.	.	.	.	*ac*	*aa*	2	.	.	1	1	.
	bb	.	4	.	2	.	2		*bb*	.	.	.	.	.	.
	cc	.	.	.	.	.	.		*cc*	.	.	2	.	1	1
	ab	4	.	.	2	2	.		*ab*	.	2	.	1	.	1
	ac	.	.	.	.	.	.		*ac*	2	.	2	1	2	1
	bc	.	.	4	.	2	2		*bc*	.	2	.	1	.	1
cc	*aa*	.	.	.	.	.	.	*bc*	*aa*	.	.	.	.	.	.
	bb	.	.	.	.	.	.		*bb*	.	2	.	1	.	1
	cc	.	.	4	.	2	2		*cc*	.	.	2	.	1	1
	ab	.	.	.	.	.	.		*ab*	2	.	.	1	1	.
	ac	4	.	.	2	2	.		*ac*	2	.	.	1	1	.
	bc	.	4	.	2	.	2		*bc*	.	2	2	1	1	2

Table 2: A pedigree CPT with the encoding 1 ↦ 4, 1/2 ↦ 2, 1/4 ↦ 1, where the number to the left of the arrow is a probability and the number to the right of the arrow is the encoding.

Multiplying ϕ_{23} with $p(v_1)$ and $p(v_2)$ will leave the number of non-zero cells unchanged. Assume now that $n_l = 30$, then ϕ_{C_1} has 3,129,855 non-zero cells compared to the number of total cells, 46,753,250,625, resulting in a fraction of non-zero cells given by 0.000067. For large enough tables, each element can be assumed to take up 8 bytes in **R**, and the memory needed to store the dense clique potential is therefore 374 gigabytes. A `sparta` object takes up $y(4k+8)$ bytes (see Paper C) where y is the number of non-zero elements, and k is the number of variables. Hence, the sparse table takes up 0.07 gigabytes which easily fits into memory on any standard laptop.

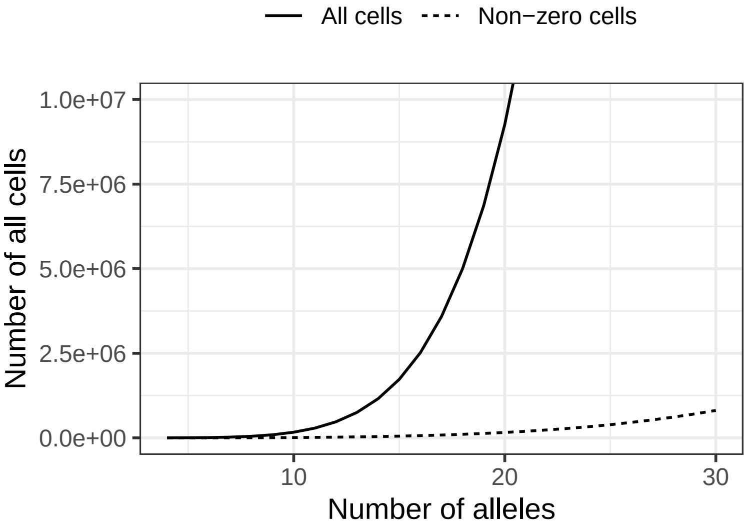


Fig. 3: Number of all cells and non-zero cells for at pedigree CPT with three alleles.

Denote by $N(m)$ the number of observed alleles for the pedigree members in the index vector m. The **R** package `pednoa`[1] was designed to determine the distribution $p(N(m))$ (possibly over several different DNA markers) and to show the benefit of using `sparta` tables as backend in the `jti` package. It is a direct extension of the work in Tvedebrink (2014), where they provide closed form solutions to calculate $p(N(m))$ for m unrelated contributors over L different DNA markers.

3.1 The Road to the Junction Tree Algorithm

The junction tree algorithm was briefly described in the previous section. In the following, JTA is described using a flow-chart in Figure 4 in order to give

[1] https://github.com/mlindsk/pednoa

an insight into the complexity and numerous sub-routines that together form the algorithm in its entirety. The solid boxes are the meta routines that must be conducted, whereas a dashed box is an example of concrete algorithm that can be used for the routine that it points towards. All solid boxes were described in the previous section, but it should be clear that JTA is not a trivial algorithm to implement when the flowchart is augmented with the dashed boxes. It takes quite some time to implement any of the dashed boxes, and in particular sparta and unity propagation described in paper C and D.

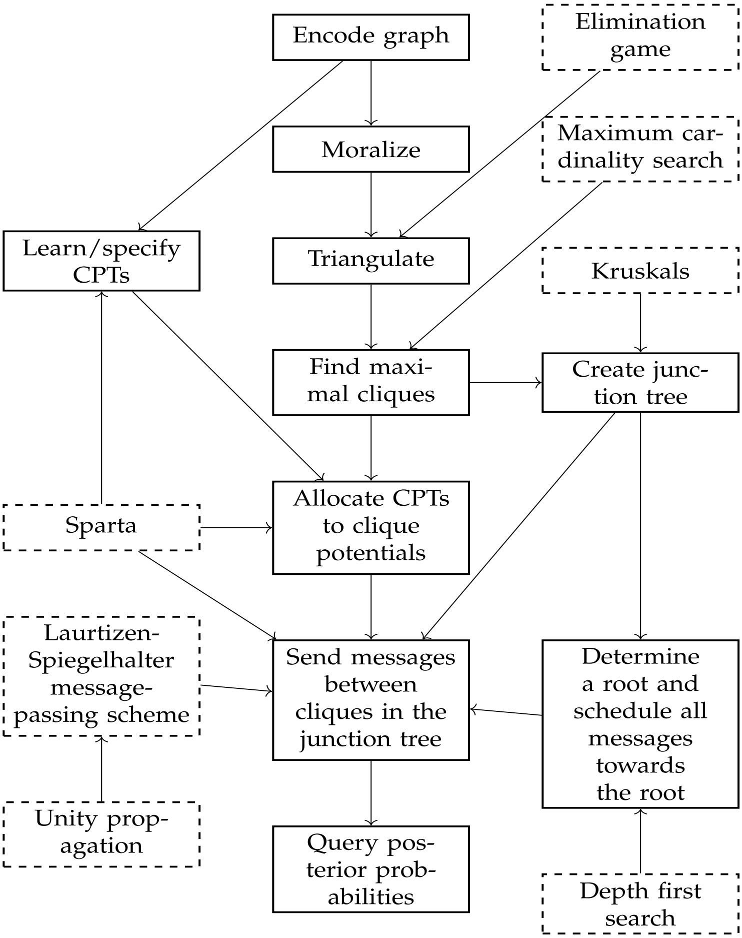

Encode graph
Elimination game
Moralize
Maximum cardinality search
Learn/specify CPTs
Triangulate
Kruskals
Find maximal cliques
Create junction tree
Sparta
Allocate CPTs to clique potentials
Laurtizen-Spiegelhalter message-passing scheme
Send messages between cliques in the junction tree
Determine a root and schedule all messages towards the root
Unity propagation
Query posterior probabilities
Depth first search

Fig. 4: The road to junction tree inference.

References

Darroch, J. N., Lauritzen, S. L., and Speed, T. P. (1980). Markov fields and log-linear interaction models for contingency tables. *The Annals of Statistics*, pages 522–539.

Deshpande, A., Garofalakis, M., and Jordan, M. I. (2013). Efficient stepwise selection in decomposable models. *arXiv preprint arXiv:1301.2267*.

Edwards, D. (2012). *Introduction to graphical modelling*. Springer Science & Business Media.

Lauritzen, S. L. (1996). *Graphical models*, volume 17 of *Oxford Statistical Science Series*. Clarendon Press.

Lindskou, M. (2021). ***ess**: Efficient Stewise Selection in Decomposable Markov Random Fields*. R package version 1.1.2.

Malvestuto, F. M. (1991). Approximating discrete probability distributions with decomposable models. *IEEE Transactions on systems, man, and cybernetics*, 21(5):1287–1294.

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

Tvedebrink, T. (2014). On the exact distribution of the numbers of alleles in dna mixtures. *International journal of legal medicine*, 128(3):427–437.

Part II

Papers

Paper A

Outlier Detection in Contingency Tables Using Decomposable Graphical Models

Mads Lindskou, Poul Svante Eriksen and Torben Tvedebrink

The paper has been published in the
Scandinavian Journal of Statistics, 47(2), 347-360

The layout has been revised.

Abstract

For high-dimensional data, it is a tedious task to determine anomalies such as outliers. We present a novel outlier detection method for high-dimensional contingency tables. We use the class of decomposable graphical models to model the relationship among the variables of interest, which can be depicted by an undirected graph called the interaction graph.

Given an interaction graph, we derive a closed form expression of the likelihood ratio test (LRT) statistic and an exact distribution for efficient simulation of the test statistic. An observation is declared an outlier if it deviates significantly from the approximated distribution of the test statistic under the null hypothesis.

We demonstrate the use of the LRT outlier detection framework on genetic data modeled by Chow-Liu trees.

1 Introduction

An outlier is a case-specific unit since it may be interpreted as natural extreme noise in some applications, whereas in other applications it may be the most interesting observation. A universal definition of an outlier is given by "Hawkins (1980): “an observation which deviates so much from the other observations in the data-set as to arouse suspicions that it was generated by a different mechanism”. It will be clear that our method adapts this definition by specifying a hypothesis of an outlier being distributed differently than all other observations in a given contingency table. This definition, however, is extremely vague and cannot be used for outlier detection on an operational level. In Section 4, we define an outlier in a contingency table by means of the likelihood ratio principle. This definition captures the perception of the definition suggested by "Hawkins (1980).

Although the literature on outlier detection is vast, the problem of detecting outliers in contingency tables has mainly been focused on two-way tables (Kuhnt, 2004, Kuhnt et al., 2014). Kuhnt (2004) took a probabilistic approach using a log-linear model and gave a formal definition of outliers based on this model. They defined an outlier region, and declared an observation as an outlier if it fell within this region. No example was given for tables with dimensions larger than three. However, for many real-world applications it is not unusual to have more than 100 dimensions, and the method of Kuhnt (2004) may fail due to the curse of dimensionality. In this paper, we focus on high-dimensional tables and the application in forensic genetics. The method is general and applies to any outlier detection problem in contingency tables including sparse and/or high-dimensional tables.

A more recent approach for outlier detection in contingency tables was proposed by Kuhnt et al. (2014) using what they called minimal patterns in log-linear models. However, it is difficult to find the minimal patterns, and they focused solely on two-dimensional tables. Kuhnt et al. (2014) commented on the need of exploring high-dimensional sparse tables, a challenge that we address in this paper.

We use a decomposable graphical model (DGM), which is a log-linear model with further restrictions imposed on the probability mass function. The advantage of using DGMs is that the probability mass function can be expressed in a closed form (Lauritzen, 1996). For DGMs, the probability mass function can be associated with an interaction graph, from which conditional independences among the variables can be inferred. This gives a way to investigate the underlying nature of outliers.

Our method relies on the assumption, that the table is adequately described by a given DGM. That is, we do not focus on learning the interaction graph. Therefore, we can not directly compare efficiency and running time with other state-of-the-art methods. However, it is also advantageous to separate the learning procedure from the outlier test since one can exploit expert knowledge and use the most recent algorithms of DGM learning in order to form the interaction graph.

The paper is organized as follows. In Section 2, we briefly introduce DGMs with some notation used for contingency tables and preliminary results introduced in Section 3. In Section 4, we introduce our novel outlier detection procedure for contingency tables using DGMs. In Section 5, we show the procedure using genetic data with a simple interaction structure between the variables. Finally, in Section 6, we discuss the work and summarize the novelties presented in the paper.

2 Undirected graphs

A brief outline of some important properties for undirected graphs are explained and visualized by a running example using the graph in Figure 1. For more details, see for example Lauritzen (1996).

A graph $G = (\Delta, E)$ is a structure, in which Δ is a finite set of vertices, and $E \subseteq \Delta^{(2)}$ is a finite set of edges, in which $\Delta^{(2)} = \{\{a,b\} \mid a,b \in \Delta,\ a \neq b\}$. The graph is weighted if a weight is assigned to each edge. The subgraph G_A consists of vertices $A \subseteq \Delta$ from G and the corresponding edges $E_A = E \cap A^{(2)}$ between them. The union of two graphs, say $G = G_1 \cup G_2$, is the graph with vertex set $E = E_1 \cup E_2$ and edge set $\Delta = \Delta_1 \cup \Delta_2$. A graph is complete if there is an edge between all pairs of vertices, and a complete subgraph is called a clique if it is not contained in any other complete subgraph. A subset of Δ is complete if it induces a complete subgraph. The cliques of the graph in Figure 1 are $C_1 = \{a,b\}, C_2 = \{b,c,d\}, C_3 = \{c,d,e\}, C_4 = \{d,e,f\}$, and $C_5 = \{f,g\}$.

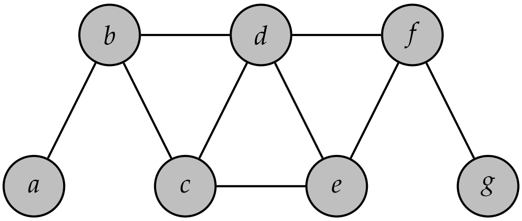


Fig. 1: Example of a decomposable graph with cliques $C_1 = \{a,b\}, C_2 = \{b,c,d\}, C_3 = \{c,d,e\}, C_4 = \{d,e,f\}$, and $C_5 = \{f,g\}$.

A sequence of vertices $v_0, v_1, \ldots, v_n$ such that $\{v_i, v_{i+1}\} \in E$ is called a path of length n. Two sets, $A, B \subseteq \Delta$, are separated by a third set $C \subseteq \Delta$ if all paths between vertices in A and B intersect C. A triple (A,B,C) of disjoint subsets of Δ forms a decomposition of G if

1. $\Delta = A \cup B \cup C$,
2. C separates A from B, and
3. C is a complete subset of Δ.

The triplet (A,B,C) is said to decompose G into $G_{A \cup C}$ and $G_{B \cup C}$. A decomposable graph is one that can be successively decomposed into its cliques.

A triangulated graph is an undirected graph with no cycle (i.e. a path starting and ending at the same vertex) of length $n \geq 4$ without chords, i.e. edges that are not part of the cycle but connects two vertices of the cycle. It can be shown that whenever a graph is triangulated, it is also decomposable and vice versa. The graph in Figure 1 is triangulated and therefore decomposable. Triangulatedness can be verified algorithmically using the Maximum Cardinality Search algorithm (Yannakakis, 1981).

Let $C_1, C_2, \ldots, C_K$ be a sequence of the cliques in an undirected graph G and define the history and separators, respectively, as

$$H_j = C_1 \cup C_2 \cup \cdots \cup C_j \quad \text{and} \quad S_j = H_{j-1} \cap C_j$$

for $j = 2, \ldots, K$ with $H_1 = C_1$. The sequence is said to obey the running intersection property (RIP) if $S_i \subseteq C_j$ for some $j < i$ for $i = 2, 3, \ldots, K$. The cliques of a decomposable graph can be numbered to have RIP ordering; the cliques C_1, C_2, C_3, C_4, and C_5 obtained from the graph in Figure 1, in that order, is a RIP ordering with separators $S_2 = \{b\}, S_3 = \{c,d\}, S_4 = \{d,e\}$, and $S_5 = \{f\}$. Notice that the RIP ordering is not necessarily unique.

A graph is connected if there is a path between any two nodes. If not, the graph is called disconnected and the subgraphs, for which the graph is connected, are called the components of the graph. A tree is a connected graph without cycles, in which any path between two vertices is unique. A minimum spanning tree is a subgraph that is a tree with minimum possible total edge weight in an undirected weighted graph. A graph, in which each component is a tree, is called a forest. The graph in Figure 1 is connected but not a tree. It follows that all forests are decomposable.

Finally, a probability measure can be associated with an *interaction graph*; an undirected graph in which each vertex is a random variable and two vertices are neighbors if, and only if, interaction is permitted between the variables.

For decomposable graphs, the density can be factorized in terms of the cliques and separators. Collectively, models for which the interaction graph is decomposable, are called decomposable graphical models (DGMs). A crucial property is that we can read the conditional probabilities from the interaction graph using the RIP ordering. More precisely, the random variables in the j'th clique are conditionally independent of the variables in the i'th clique given the j'th separator for all $i < j$.

3 Notation and preliminaries for contingency tables

We use the notation introduced by Lauritzen (1996). Let Δ denote a finite set of discrete variables, in which each variable, $\delta \in \Delta$, takes a value in the *level set* I_δ. An outcome, $i = (i_\delta)_{\delta \in \Delta}$, is a cell of the contingency table, where

$$i \in I = \times_{\delta \in \Delta} I_\delta.$$

The entire contingency table of counts is the set $n = \{n(i)\}_{i \in I}$, where $n(i)$ is the number of observations that falls in cell i and $|n| = \sum_{i \in I} n(i)$ is the total number of observations. The probability that an observation belongs to cell i is denoted as $p(i)$.

When interest is only on some subset of Δ, say a, we can form the a-marginal with marginal cells

$$i_a \in I_a = \times_{\delta \in a} I_\delta.$$

The number of observations in cell i_a in the a-marginal is given by

$$n_a(i_a) = \sum_{j \in I : j_a = i_a} n(j) = \sum_{j_{\Delta \setminus a} \in I_{\Delta \setminus a}} n(i_a, j_{\Delta \setminus a}),$$

where the last expression is convenient for handling marginal tables. Similarly, p_a denotes the a-marginal density of p.

The decomposable graphical model can then be written (Lauritzen, 1996) as

$$p(i) = p_{C_1}(i_{C_1}) \prod_{k=2}^{K} \frac{p_{C_k}(i_{C_k})}{p_{S_k}(i_{S_k})}, \tag{1}$$

where $C_1, C_2, \ldots, C_K$ and $S_2, S_3, \ldots, S_K$ are the RIP ordered cliques and separators in a decomposable graph.

Let the sample $y_1, y_2, \ldots, y_M$ with $M = |n|$ be drawn independently from the density in (1) and define the contingency table of zeroes except for cell $i = y_\alpha$, which contains a one:

$$x^{(\alpha)}(i) = \begin{cases} 1 \text{ if } i = y_\alpha, \\ 0 \text{ otherwise} \end{cases}$$

for $\alpha = 1, 2, \ldots, M$. The cell counts in the observed table is then given by

$$n(i) = \sum_{\alpha=1}^{M} x^{(\alpha)}(i), \quad i \in I.$$

Assume the cell counts to be random, but the total number of counts to be fixed, and denote by N_a the stochastic counterpart of n_a. Then, the joint density of the marginal tables N_a, $a \subseteq \Delta$, follows the multinomial distribution

$$P(N_a = n_a) = \binom{M}{n_a} \prod_{i \in I_a} p_a(i)^{n_a(i)}, \tag{2}$$

where

$$\binom{M}{n_a} := \frac{M!}{\prod_{i \in I_a} n_a(i)!}$$

is a multinomial coefficient. Since the model of interest is decomposable, it follows that the set of clique marginals, $N^* = \{N_C\}_{C \in \mathcal{C}}$, is sufficient (although not minimally sufficient) for $p^* = \{p_C\}_{C \in \mathcal{C}}$, where $\mathcal{C}$ is the set of cliques. When the cell probabilities are not restricted in any way, except for the constraints of being positive and summing to one, the model is said to be *saturated*. In the family of log-linear models, the logarithms of the cell probabilities are typically constrained to follow an ANOVA-like factorial expansion (Højsgaard et al., 2012). Graphical models are constructed from saturated models that are saturated on each clique. For saturated models, the estimated cell probabilities are given as the cell counts divided by the total number of observations (Lauritzen, 1996). Hence, we obtain the set of maximum likelihood estimates

$$\hat{p}^* = \left\{ \frac{n_C}{M} \right\}_{C \in \mathcal{C}},$$

and the estimate of (1) is thus

$$\hat{p}(i) = \left(\frac{n_{C_1}(i_{C_1})}{M} \right) \prod_{k=2}^{K} \frac{n_{C_k}(i_{C_k})}{n_{S_k}(i_{S_k})}. \tag{3}$$

Let n^* denote a realized value of N^*. Hence, the maximum of the likelihood function over the clique marginals is

$$\begin{aligned}
L(n^*) &= \prod_{i \in I} \hat{p}(i)^{n(i)} \\
&= \left\{ \prod_{i \in I} \left(\frac{n_{C_1}(i_{C_1})}{M} \right)^{n(i)} \right\} \left\{ \prod_{k=2}^{K} \frac{\prod_{i \in I} n_{C_k}(i_{C_k})^{n(i)}}{\prod_{i \in I} n_{S_k}(i_{S_k})^{n(i)}} \right\} \\
&= \left\{ \prod_{i_{C_1} \in I_{C_1}} \left(\frac{n_{C_1}(i_{C_1})}{M} \right)^{\sum_{i_{\Delta \setminus C_1}} n(i_{C_1}, i_{\Delta \setminus C_1})} \right\} \times \\
&\quad \left\{ \prod_{k=2}^{K} \frac{\prod_{i_{C_k} \in I_{C_k}} n_{C_k}(i_{C_k})^{\sum_{i_{\Delta \setminus C_k}} n(i_{C_k}, i_{\Delta \setminus C_k})}}{\prod_{i_{S_k} \in I_{S_k}} n_{S_k}(i_{S_k})^{\sum_{i_{\Delta \setminus C_k}} n(i_{S_k}, i_{\Delta \setminus S_k})}} \right\} \\
&= \prod_{i_{C_1} \in I_{C_1}} \left(\frac{n(i_{C_1})}{M} \right)^{n(i_{C_1})} \prod_{k=2}^{K} \frac{\prod_{i_{C_k} \in I_{C_k}} n(i_{C_k})^{n(i_{C_k})}}{\prod_{i_{S_k} \in I_{S_k}} n(i_{S_k})^{n(i_{S_k})}}.
\end{aligned} \tag{4}$$

4 Test for outliers in contingency tables

In order to test if an observation, say $y := y_M$, is an outlier, we assume that y_M is an observation sampled from a distribution different from that of $y_1, y_2, \ldots, y_{M-1}$. Let $X^{(\alpha)}$ be the stochastic counterpart of $x^{(\alpha)}$. The distribution of Y_M can then be described by the expected table $\mathrm{E}\left[X^{(M)}\right]$, where $\mathrm{E}\left[X^{(M)}(y_M)\right] = P(Y_M = y_M)$. Assuming that

$$\mathrm{E}\left[X^{(\alpha)}\right] = p, \quad \alpha = 1, 2, \ldots, M-1$$

and

$$\mathrm{E}\left[X^{(M)}\right] = q,$$

where p and q are specified through (1), the null hypothesis is

$$H_0 : q = p.$$

Hence, if H_0 is false, y_M is considered an outlier in the table defined by $y_\alpha, \alpha < M$. Denote by $n^* - x^*$ the subtraction of the two sets of tables n^* and x^* as

$$n^* - x^* := \{n_C - x_C\}_{C \in \mathcal{C}},$$

where $n_C - x_C$ is the element-wise subtraction in the C-marginals of n and x. We then define the likelihood ratio

$$LR = \frac{L(n^*)}{L(x^*)L(n^* - x^*)}, \tag{5}$$

where $x^* := (x^{(M)})^*$ and where the numerator corresponds to the likelihood (4) under H_0. Small values of LR will be critical to H_0. It is clear from (4) that $L(x^*) = 1$ since we obtain factors of either 0^0 or 1^1. Consider the remaining fraction of LR:

$$\frac{L(n^*)}{L(n^* - x^*)}. \tag{6}$$

The factors involving M can be written as

$$Z(M) = \prod_{i_{C_1} \in I_{C_1}} \frac{\left(\dfrac{1}{M}\right)^{n_{C_1}(i_{C_1})}}{\left(\dfrac{1}{M-1}\right)^{n_{C_1}(i_{C_1}) - x_{C_1}(i_{C_1})}} = M^{-M}(M-1)^{M-1}.$$

The remaining factors are of the form

$$Q(a) = \prod_{i_a \in I_a} \frac{n_a(i_a)^{n_a(i_a)}}{(n_a(i_a) - x_a(i_a))^{(n_a(i_a) - x_a(i_a))}} = \frac{n_a(y_a)^{n_a(y_a)}}{(n_a(y_a) - 1)^{(n_a(y_a) - 1)}},$$

where the ratios are simplified since $x_a(i_a) = 0$ unless $i_a = y_a$. Combining it all, we end up with

$$LR = Z(M) \prod_{k=1}^{K} Q(C_k) \prod_{k=2}^{K} Q(S_k)^{-1}.$$

Define $H(x) := G(x-1) - G(x)$, where

$$G(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x \log(x) & \text{otherwise .} \end{cases}$$

The deviance statistic $D(y) = -2\log(LR)$ then becomes

$$D(y) = 2\left(\sum_{k=1}^{K} H(n_{C_k}(y_{C_k})) - \sum_{k=2}^{K} H(n_{S_k}(y_{S_k})) - H(M)\right),$$

where large values are critical to the null hypothesis.

In order to perform an exact test of H_0, the conditional distribution of $D(X^*) := D(Y)$ given N^* is needed. Below, we derive a closed form of the probability

$$\begin{aligned} P(X^* = x^* \mid N^* = n^*) &= \frac{P(X^* = x^*, N^* = n^*)}{P(N^* = n^*)} \\ &= \frac{P(X^* = x^*)P(N^* - X^* = n^* - x^*)}{P(N^* = n^*)}. \end{aligned} \tag{7}$$

We show how to approximate the distribution of $D(Y)$ by simulating from (7). From the proof of Theorem 4.22 in Lauritzen (1996), we infer that

$$P(N^* = n^*) = P(N_{C_1} = n_{C_1}) \prod_{k=2}^{K} \frac{P(N_{C_k} = n_{C_k})}{P(N_{S_k} = n_{S_k})}. \tag{8}$$

Using (2) together with (8), the contribution from the kth clique to (7) is reduced to one since

$$\prod_{i_{C_k} \in I_{C_k}} \frac{p_{C_k}(i_{C_k})^{x_{C_k}(i_{C_k})} p_{C_k}(i_{C_k})^{n_{C_k}(i_{C_k}) - x_{C_k}(i_{C_k})}}{p_{C_k}(i_{C_k})^{n_{C_k}(i_{C_k})}} = 1,$$

and thus all the parameters, $p_{C_k}(i_{C_k})$, cancel in (7); and similarly for the separators. Also, it is clear from (2) that all remaining contributions from $P(X^* = x^*)$ are reduced to 1. Putting it all together, we have

$$\begin{aligned} &P(X^* = x^* \mid N^* = n^*) = \\ &\quad \left\{ \prod_{i_{C_1} \in I_{C_1}} \frac{\binom{M-1}{n_{C_1}(i_{C_1}) - x_{C_1}(i_{C_1})}}{\binom{M}{n_{C_1}(i_{C_1})}} \right\} \times \\ &\quad \left\{ \prod_{k=2}^{K} \prod_{i_{C_k} \in I_{C_k}} \frac{\binom{M-1}{n_{C_k}(i_{C_k}) - x_{C_k}(i_{C_k})}}{\binom{M}{n_{C_k}(i_{C_k})}} \prod_{i_{S_k} \in I_{S_k}} \frac{\binom{M}{n_{S_k}(i_{S_k})}}{\binom{M-1}{n_{S_k}(i_{S_k}) - x_{S_k}(i_{S_k})}} \right\}, \end{aligned}$$

which, in terms of y, can be written as:

$$P(Y = y \mid N^* = n^*) = \frac{n_{C_1}(y_{C_1})}{M} \prod_{k=2}^{K} \frac{n_{C_k}(y_{C_k})}{n_{S_k}(y_{S_k})}, \tag{9}$$

where $0/0 := 0$; i.e. if $n_{S_k}(y_{S_k}) = 0$ for some $k > 1$, then $n_{C_k}(y_{C_k}) = 0$ and hence $P(Y = y \mid N^* = n^*) = 0$. Notice that the density in (9) is the same as in (3). This is not surprising since our best estimate of the probability of observing $Y = y$ is exactly the cell probability in (3) based on all observations including y.

In order to calculate the conditional mean of $D(Y)$, define the transformation of $D(Y)$ discarding the constant $H(M)$ as

$$T(Y) = \sum_{k=1}^{K} T_k(Y), \tag{10}$$

where

- $T_1(Y) := H(n_{C_1}(Y_{C_1}))$ and
- $T_k(Y) := H(n_{C_k}(Y_{C_k})) - H(n_{S_k}(Y_{S_k}))$ for $k = 2,3,\dots,K$.

Notice that T_k is a function depending only on Y through Y_{C_k} for $k = 1,2,\dots,K$.

Lemma 1. *The conditional distribution of Y_{C_k} given N^* is*

$$P(Y_{C_k} = y_{C_k} \mid N^* = n^*) = \frac{n_{C_k}(y_{C_k})}{M}. \tag{11}$$

Proof. We prove the lemma by induction on the number of cliques. If $K = 1$, the result is trivial and we assume the result hold for $K > 1$. In this proof, we use the more compact notation $P(y_a) := P(Y_a = y_a)$. Suppose now that the number of cliques is $K + 1$. Then by conditional independence induced by the separator S_{K+1}, it follows that

$$\begin{aligned} &P(y_{C_1},\dots,y_{C_{K+1}},y_{S_{K+1}} \mid n^*) \\ =&P(y_{C_{K+1}} \mid y_{S_{K+1}},y_{C_1},\dots,y_{C_K},n^*)P(y_{C_1},\dots,y_{C_K} \mid n^*) \\ =&P(y_{C_{K+1}} \mid y_{S_{K+1}},n^*)P(y_{C_1},\dots,y_{C_K} \mid n^*), \end{aligned} \tag{12}$$

implying by (9) that

$$P(y_{C_{K+1}} \mid y_{S_{K+1}},n^*) = \frac{n_{C_{K+1}}(y_{C_{K+1}})}{n_{S_{K+1}}(y_{S_{K+1}})}. \tag{13}$$

Furthermore, by the induction hypothesis, it follows that

$$P(y_{S_{K+1}} \mid n^*) = \sum_{y_{C_j \setminus S_{K+1}}} \frac{n_{C_j}(y_{C_j \setminus S_{K+1}}, y_{S_{K+1}})}{M} = \frac{n_{S_{K+1}}(y_{S_{K+1}})}{M}, \tag{14}$$

since $S_{K+1} \subseteq C_j$ for some $j < K+1$. Multiplication of (13) and (14) gives the result stated in (11). □

Using Lemma 1, the conditional expectation takes the form

$$\mathrm{E}[T(Y) \mid N^* = n^*] = \sum_{k=1}^{K} \sum_{i_{C_k} \in I_{C_k}} T_k(i_{C_k}) \cdot \frac{n_{C_k}(i_{C_k})}{M}. \tag{15}$$

The conditional variance can be calculated from moments over pairs of cliques, which can be done efficiently by using the RIP ordering.

4.1 Simulation

Using (12), it follows that

$$P(Y = y \mid N^* = n^*) = P(Y_{C_1} = y_{C_1} \mid N^* = n^*) \prod_{k=2}^{K} P(Y_{C_k} = y_{C_k} \mid Y_{S_k} = y_{S_k}, N^* = n^*).$$

Hence, we can simulate an observation $Y = y$ from the table conditionally on $N^* = n^*$ by

- simulating $Y_{C_1} = y_{C_1}$ from the distribution

$$P(Y_{C_1} = y_{C_1} \mid N^* = n^*) = \frac{n_{C_1}(y_{C_1})}{M}, \tag{16}$$

- for $k = 2, 3, \ldots, K$, simulate $Y_{C_k} = y_{C_k}$ conditionally on $Y_{S_k} = y_{S_k}$ from the distribution

$$P(Y_{C_k} = y_{C_k} \mid Y_{S_k} = y_{S_k}, N^*) = \frac{n_{C_k}(y_{C_k \setminus S_k}, y_{S_k})}{n_{S_k}(y_{S_k})}. \tag{17}$$

For $k = 1$, it is straightforward to simulate using (16). Suppose that we have simulations of Y_{C_j} for $1 \leq j < k$. Since $S_k \subset C_j$ for some $j < k$, we know the value of Y_{S_k} and we can simulate $Y_{C_k \setminus S_k} = y_{C_k \setminus S_k}$ using (17) to obtain $y_{C_k} = (y_{C_k \setminus S_k}, y_{S_k})$.

5 An Example in Forensic Genetics

Ancestry Informative Markers (AIMs) are genetic markers selected for their informativeness concerning genogeographic origin. The term genogeographic is used rather than geographic since it emphasizes the genetic component of the phenomena and also stresses that populations in this context are defined more by their shared genetic ancestry than by geography. Often, genetics and geography is strongly associated. However, culture, ethnicity, and language are often stronger associated with populations than with regional origin (Harrison, 1977, Cavalli-Sforza et al., 1994).

The prevailing AIMs are single nucleotide polymorphisms (SNPs), which are specific locations in the genome with nucleotide variation in the population. Bi-allelic SNPs are most common, but tri- and tetra-allelic SNPs exist. SNPs used for ancestry assessment are also referred to as ancestry informative SNPs (AISNPs), and there exists large population tables with allele frequencies (Kidd et al., 2018).

The standard approaches to infer the population of origin of a DNA profile of interest include analysis with the STRUCTURE program (Pritchard et al., 2000, Phillips, 2015), maximum likelihood assignment (Phillips et al., 2007, Kidd et al., 2018), and principal components analysis (Phillips, 2015). However, none of these methods directly allow all populations in a reference table to be rejected as potential populations of origin. That a given population is the most likely one among a number of populations does not imply that the population is relevant for a given profile of interest.

Tvedebrink et al. (2018) developed an LRT based on the principles of Fisher's exact test. Each population in the reference table is tested as a possible population of origin. In fact, this is an outlier detection test.

Recently, advances in DNA sequencing has made it possible to sequence short segments of DNA (<200 basepairs) including two or more SNPs. These are called *microhaplotypes* (Kidd et al., 2014, Kidd and Speed, 2015, Oldoni et al., 2019) (or microhaps for short). They have been demonstrated to be well suited for ancestry assessment. The short distance between SNPs within a microhap implies that recombination among them rarely occurs. Hence, the methodology of Tvedebrink et al. (2018) can not be used as this assumes mutually independence of the SNPs within a population. However, the caveats of not having a relevant population in the reference table still exists, which makes the methodology derived above relevant to use for microhaps.

Let L denote the total number of microhaps of interest in a given population. Each microhap, $\mathcal{H}_l = (s_{l1}, \dots, s_{lm_l})$, $l = 1, \dots, L$, consists of m_l SNPs, where s_{lk} is the k'th SNP in the l'th microhaplotype.

A given profile, z, of interest is written

$$z = (z^{(1)}, z^{(2)}),$$

where

$$z^{(j)} \in I = \times_{\delta \in \Delta} I_\delta, \quad j = 1,2$$

is the j'th *partial profile* of z, one for each chromosome. Here, Δ is the collection of all SNPs in consideration with each variable $\delta \in \Delta$ being binary (bi-allelic SNPs). By Mendelian segregation, we can assume the two partial profiles for each profile to be independent.

Below, we demonstrate how to exploit the dependency structure of microhaps by using the likelihood ratio test derived in Section 4 in order to test whether a DNA profile is an outlier in a given table.

5.1 Summary of 1000 Genome Data

We analyzed data from the 1000 Genomes Project Consortium et al. (2015) consisting of 5,008 partial DNA profiles and 130 microhaps from five different continental regions, see Table 1. We excluded 21 microhaps from the original data because all SNPs were not available. We found that some microhaps at a specific chromosome were not independent (data not shown). We disregarded these and assumed all remaining microhaps to be independent within and between chromosomes. The total number of analyzed microhaps considered was $L = 97$ consisting of a total of 276 bi-allelic SNP markers. The number of SNPs, m_ℓ, within microhaps ranges from two to five. However, it should be noticed that the derived methodology handles any value of m_ℓ.

Marginal tables restricted to profiles in a given continental region will be referred to as continental specific tables. A table is then expressed as the sum of the continental specific tables as

$$n = \sum_{d \in CR} n^d,$$

where $CR = \{AFR, AMR, EAS, EUR, SAS\}$ is the set of continental regions described in Table 1. Hence,

$$|n| = \sum_{d \in CR} |n^d|.$$

As an example, the genotype of a selected profile, z, from Europe on $\mathcal{H}_1$ is seen in Table 2. The s_{12}-marginals are

$$z^{(1)}_{s_{12}} = T \quad \text{and} \quad z^{(2)}_{s_{12}} = C.$$

Continental region	Number of partial profiles: $\lvert n^d \rvert$
Africa (*AFR*)	1,322
America (*AMR*)	694
East Asia (*EAS*)	1,008
Europe (*EUR*)	1,006
South Asia (*SAS*)	978

Table 1: Number of partial profiles for each continental region in the 1000 Genomes Project data.

In order to exploit the framework presented in Section 4, we need to determine an interaction graph. The framework of Section 4 works for all DGMs. In the following section, we demonstrate the outlier framework using Chow-Liu trees (Chow and Liu, 1968). In order to learn a Chow-Liu tree, one must compute the mutual information for all pairs of variables and assign these weights to the corresponding edges. A Chow-Liu tree is then formed as a minimum spanning tree of the weighted graph. Chow and Liu (1968) showed that among all minimum spanning trees Chow-Liu trees are optimal in terms of maximizing the likelihood function. We denote Chow-Liu trees as G^T. For the saturated model, the interaction graph is denoted G^S i.e. the complete graph. Finally, the graph with no edges, implying mutual independence among all SNPs, is denoted as $G^{\varnothing}$.

SNP	s_{11}	s_{12}	s_{11}
$z^{(1)}_{\mathcal{H}_1}$	T	T	C
$z^{(2)}_{\mathcal{H}_1}$	T	C	C

Table 2: The genotype of haplotype $\mathcal{H}_1$ for a selected profile in the European table n^{EUR}.

5.2 Testing For z Being an Outlier

Consider again an arbitrary profile $z = (z^{(1)}, z^{(2)})$ and suppose that we are interested in whether or not z belongs to a given table n^d. Define $n^{d,j}, j = 1,2$ as the table n^d with the partial profile $z^{(j)}$ included. Next, construct the interaction graphs for $n^{d,1}$ and $n^{d,2}$ and denote by $G^{j,\ell}$ the ℓ'th component of the interaction graph for $n^{d,j}$ with the set of cliques $\mathcal{C}(G^{j,\ell})$. Here, G is arbitrary. Below, we will consider the interaction graphs already mentioned: G^T, G^S and $G^{\varnothing}$.

Assume that $Z^{(1)} \sim p_1$, $Z^{(2)} \sim p_2$ and all partial profiles in n^d are distributed according to p. The hypothesis then takes the form

$$H_0 : p_1 = p_2 = p.$$

For $j = 1,2$, define the two test statistics $T(Z^{(j)})$ using (10), where we aggregate over all microhaps

$$T(Z^{(j)}) = \sum_{\ell=1}^{L} \sum_{C \in \mathcal{C}(G^{j,\ell})} T_C(Z^{(j)}).$$

To evaluate H_0 that both partial profiles originate from the same population, we define

$$T(Z) := T(Z^{(1)}) + T(Z^{(2)})$$

as the convolution of $T(Z^{(1)})$ and $T(Z^{(1)})$. If the p-value $P(T(Z) \geq T(z))$, where P is the probability density function of $T(Z)$, is below a chosen level of significance α, the null hypothesis, H_0, is rejected. Here, we arbitrarily, choose $\alpha = 0.05$.

5.3 Results

Now, for each combination $(d',d) \in CR \times CR$ and for each interaction graph G^T, G^S and $G^{\varnothing}$, we calculate the proportion of profiles in $n^{d'}$ that are outliers in n^d as follows: For all profiles $z = (z^{(1)}, z^{(2)})$ in $n^{d'}$, we form $n^{d,j}, j = 1,2$. If the observed value $T(z)$ falls below the $\alpha-$quartile in the distribution of $T(Z)$, we regard z as an outlier in n^d. Note, that d' corresponds to the true origin, while d is the hypothesized origin. For cells with $d' \neq d$, we aim for numbers close to one that reflect that the model predicts the true origin in all cases. For cells with $d' = d$, the aim is to obtain numbers as close to α as possible.

The results are shown in Table 3, where rows indicate d' and columns indicate d. As an example, the proportion of profiles from *SAS* that are outliers in *EUR* according to the model, is 0.922 for G^T, 0.928 for G^S and 0.863 for $G^{\varnothing}$. In general, both G^T and G^S outperform $G^{\varnothing}$, especially when the hypothesized origin is *SAS*. For cells with $d' = d$, the performance is equally good for the three types of interaction graphs. However, for cells with $d \neq d$, G^S in general performs slightly better than G^T. It is striking that the proportion of outliers detected for $(d',d) = (EUR, SAS)$ is more than doubled if we use either G^T or G^S rather than $G^{\varnothing}$. This underlines that SNPs within microhaps cannot be assumed to be mutually independent.

The excess of outliers (proportions greater than $\alpha = 0.05$) for *AFR* and *AMR* when H_0 is true, $d = d'$, may be caused by heterogeneity in the continental regions. That is, genotyping errors or genetic deviations among the profiles may result in higher rejecting rates than the significance level.

d d'	*EUR*	*EAS*	*AMR*	*SAS*	*AFR*
EUR	0.054	1	0.191	0.509	1
	0.036	1	0.175	0.495	1
	0.046	1	0.145	0.231	1
EAS	1	0.054	0.994	0.966	1
	1	0.044	0.994	0.986	1
	1	0.063	0.994	0.980	1
AMR	0.778	1	0.095	0.769	1
	0.790	1	0.089	0.847	1
	0.697	1	0.049	0.565	1
SAS	0.922	1	0.710	0.037	1
	0.928	1	0.699	0.025	1
	0.863	1	0.620	0.047	1
AFR	1	1	0.997	1	0.101
	1	1	0.998	1	0.098
	0.998	1	0.918	0.834	0.106

▓ G^T ░ G^S □ $G^\emptyset$

Table 3: Performance matrix of outlier tests using G^T, G^S and $G^\emptyset$ (▓, ░, □) as interaction graphs. Rows (d') correspond to the true origin, while columns (d) correspond to the model under the null hypothesis. The numbers are the proportions of profiles in d' that were declared outliers in d according to the model.

6 Discussion

We propose a new method based on decomposable graphical models to detect outliers and anomalies in contingency tables of any dimension and sparsity.

We provide a closed form expression of the deviance and demonstrate how to approximate the distribution of the deviance using simulation. A simulation study showed the performance of the method for different types of associations between SNPs within microhaps.

The performance of the presented method is not affected by sparsity due to the fact that the model aggregates local information via cliques in the interaction graph.

Especially for sparse tables, however, it is crucial that the hypothesized outlier is appended to the table as described in Section 5.2. An increase from zero to one in a cell count in a sparse table can affect the result dramatically. However, this may also lead a large increase in computational time since the approximated distribution must be recalculated each time an observation is being tested as an outlier. Furthermore, the structure of the interaction graph must be refitted. Hence, if the model is to be deployed, it is important how the model is implemented and, in particular, what kind of interaction graph is used.

We have implemented the relevant method in the programming languages `R` and `C++`. The user can choose between the three graphs G^T,G^S and $G^{\emptyset}$ or specify any decomposable interaction graph. The program can be shared on request.

In the forensic genetic example, the model based on G^T was slightly more inaccurate in detecting outliers than the model based on G^S. There may be several reasons for this. The sample size is large compared to the dimension of the tables, and it may be adequate to use the naïve estimates obtained using G^S. In addition, more than half of the microhaps included only two SNPs implying that G^T and G^S coincide. Furthermore, it may be too restrictive to model the dependencies between SNPs using only trees.

Although decomposable models are a subclass of undirected graphical models, finding an optimal decomposable model is known to be intractable (Deshpande et al., 2001). However, there exists interesting heuristic algorithms that handles more complex structures than the tree-based method that we have used (Pérez et al., 2016, Deshpande et al., 2001, Bukszár and Prékopa, 2001, Altmueller and Haralick, 2004).

Acknowledgments

The authors would like to thank Niels Morling and Helle Smidt Mogensen from Section of Forensic Genetics, Department of Forensic Medicine, Faculty of Health and Medical Sciences, University of Copenhagen, Denmark for providing data and valuable comments on the manuscript.

Furthermore, anonymous reviewers increased the level of clarity and pointed to previous work on similar topics.

References

Altmueller, S. M. and Haralick, R. M. (2004). Practical aspects of efficient forward selection in decomposable graphical models. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 710–715. IEEE.

Bukszár, J. and Prékopa, A. (2001). Probability bounds with cherry trees. *Mathematics of Operations Research*, 26(1):174–192.

Cavalli-Sforza, L. L., Menozzi, P., and Piazza, A. (1994). *The History and Geography of Human Genes*. Princeton Universily Press, 41 William Street, Princeton, New Jersey 08540.

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

Consortium, . G. P. et al. (2015). A global reference for human genetic variation. *Nature*, 526(7571):68.

Deshpande, A., Garofalakis, M., and Jordan, M. I. (2001). Efficient stepwise selection in decomposable models. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 128–135. Morgan Kaufmann Publishers Inc.

Harrison, G. A. (1977). *Population structure and human variation*. International Biological Programme. Cambridge University Press.

"Hawkins, D. M. ("1980"). *"Identification of outliers"*, volume "11". "Springer".

Højsgaard, S., Edwards, D., and Lauritzen, S. (2012). *Graphical models with R*. Springer Science & Business Media.

Kidd, K. K., Pakstis, A. J., Speed, W. C., Lagacé, R., Chang, J., Wootton, S., Haigh, E., and Kidd, J. R. (2014). Current sequencing technology makes microhaplotypes a powerful new type of genetic marker for forensics. *Forensic Science International: Genetics*, 12:215 – 224.

Kidd, K. K., Soundararajan, U., Rajeevan, H., Pakstis, A. J., Moore, K. N., and Ropero-Miller, J. D. (2018). The redesigned forensic research/reference on genetics-knowledge base, frog-kb. *Forensic Science International: Genetics*, 33:33 – 37.

Kidd, K. K. and Speed, W. C. (2015). Criteria for selecting microhaplotypes: mixture detection and deconvolution. *Investigative Genetics*, 6(1):1–10.

Kuhnt, S. (2004). Outlier identification procedures for contingency tables using maximum likelihood and l1 estimates. *Scandinavian journal of statistics*, 31(3):431–442.

Kuhnt, S., Rapallo, F., and Rehage, A. (2014). Outlier detection in contingency tables based on minimal patterns. *Statistics and Computing*, 24(3):481–491.

Lauritzen, S. L. (1996). *Graphical models*, volume 17 of *Oxford Statistical Science Series*. Clarendon Press.

Oldoni, F., Kidd, K. K., and Podini, D. (2019). Microhaplotypes in forensic genetics. *Forensic Science International: Genetics*, 38:54 – 69.

Pérez, A., Inza, I., and Lozano, J. A. (2016). Efficient approximation of probability distributions with k -order decomposable models. *International Journal of Approximate Reasoning*, 74(nil):58–87.

Phillips, C. (2015). Forensic genetic analysis of bio-geographical ancestry. *Forensic Sci Int Genet*, 18:49–65.

Phillips, C., Salas, A., Sánchez, J. J., Fondevila, M., Gómez-Tato, A., Álvarez-Dios, J., Calaza, M., de Cal, M. C., Ballard, D., Lareu, M. V., and Carracedo, Á. (2007). SNPforID Consortium. Inferring ancestral origin using a single multiplex assay of ancestry-informative marker SNPs. *Forensic Sci Int Genet*, 1:272–280.

Pritchard, J. K., Stephens, M., and Donnelly, P. J. (2000). Inference of population structure using multilocus genotype data. *Genetics*, 155:945–959.

Tvedebrink, T., Eriksen, P. S., Mogensen, H. S., and Morling, N. (2018). Weight of the Evidence of Genetic Investigations of Ancestry Informative Markers. *Theoretical Population Biology*, 120:1–10.

Yannakakis, M. (1981). Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79.

Paper B

Detecting Outliers in High-dimensional Data with Mixed Variable Types using Conditional Gaussian Regression Models

Mads Lindskou, Torben Tvedebrink, Poul Svante Eriksen and Niels Morling

The paper has been submitted to
arXiv

The layout has been revised.

Abstract

Outlier detection has gained increasing interest in recent years, due to newly emerging technologies and the huge amount of high-dimensional data that are now available. Outlier detection can help practitioners to identify unwanted noise and/or locate interesting abnormal observations. To address this, we developed a novel method for outlier detection for use in, possibly high-dimensional, datasets with both discrete and continuous variables. We exploit the family of decomposable graphical models in order to model the relationship between the variables and use this to form an exact likelihood ratio test for an observation that is considered an outlier. We show that our method outperforms the state-of-the-art Isolation Forest algorithm on a real data example.

1 Introduction

Outlier detection is an important learning paradigm and has drawn significant attention within the research community, as shown by the increasing number of publications in this field. An outlier in a data set is an observation that, for some reason, does not share the same characteristics as the majority (there may be more than one outlier) of all other observations. An outlier may be the most interesting observation in some situations. In other situations, it may be regarded as extreme noise, and it may be appropriate to remove it from the data. There is no clear mathematical definition of an outlier in the literature. "Hawkins (1980) gave the following definition: “an observation which deviates so much from the other observations in the dataset as to arouse suspicions that it was generated by a different mechanism”. In Lindskou et al. (2019), this definition, was adapted by specifying a statistical hypothesis of an outlier being distributed differently than all other observations for discrete data sets. In this paper, we extend this definition to capture outliers in data sets with variables of mixed types, i.e. both discrete and continuous variables.

Most research on outlier detection has been focused on the two pure cases where all variables are either discrete or continuous, while little research has been done in the mixed case for high-dimensional data sets (Garchery and Granitzer, 2018). State-of-the art algorithms include the Isolation Forest (iForest) algorithm (Liu et al., 2008) which has gained a lot of attention in recent years. Many software packages implement iForest. In particular, the procedure is implemented in the major data science languages, such as R and Python, making it readily available to practitioners. Although iForest was designed for outlier detection in the pure continuous case, it is frequently used in the mixed case where the discrete variables are transformed into continuous variables. However, the transformation may induce an unwanted ordering of or distance between the levels of the discrete variables. Hence, the information content of the data may be altered by the transformation, but nonetheless iForest is used in many papers on outlier detection with mixed data, and the performance is most often excellent, see e.g. Eiras-Franco et al. (2019), Xu et al. (2019), Aryal et al. (2016), Garchery and Granitzer (2018) and Aryal et al. (2019). In the review papers by Domingues et al. (2018) and Emmott et al. (2015), iForest is recommended as the best overall outlier detection procedure and is recommended in production environments. In Section 8.2, we show that our method outperforms iForest as an outlier detection method in the mixed case, when data includes a large number of discrete variables. This is in contradiction the findings in the aforementioned papers.

We propose a novel probabilistic outlier detection method that relies on the family of decomposable mixed graphical models, see e.g. Lauritzen (1996). Graphical models is a family of statistical models, for which the dependencies between variables can be depicted or read off from a graph called the interaction graph. They are composed of a set of random variables (possibly both continuous and discrete) and an interaction graph, for which each of the vertices represents one of the random variables. In essence, a graphical model encodes the conditional independencies between the random variables. By imposing the graphs to be decomposable, the likelihood function is ensured to be on closed form, which enables us to express what is meant by an outlier using an exact likelihood ratio test (LRT). Furthermore, the components of the proposed LRT are composed of local information from the graph in terms of cliques yielding a way to explore which variables have the largest impact of the declaration of an observation as an outlier.

The likelihood function corresponds to a multivariate multiple regression model over the continuous variables, however due to local independencies, it is possible to restrict attention to simple multiple linear regressions with both continuous and discrete explanatory variables. In the pure discrete case, the method coincide with the one given in Lindskou et al. (2019) which will be apparent in Section 5. In addition, the method also handles the pure continuous case which is shown to be equivalent to a sum of studentized residuals over local structures in the graph.

The LRT relies on simulating observations from the model, as in Lindskou et al. (2019). However, in the mixed case we show that it is redundant to simulate the continuous counterpart of a simulated discrete observation. It turns out that the continuous contribution to the likelihood ratio can be drawn from a beta distribution once the discrete counterpart is known.

The rest of the paper is organised as follows. Section 2 reviews the definitions of graphical models needed together with a useful proposition. Section 4 is devoted to the notation and likelihood function needed to arrive at the exact likelihood ratio test given in Section 5. Section 7 summaries the proposed outlier detection method, ODMGM, in a pseudo algorithm providing a very detailed explanation of the steps to carry out. In Section 8, a real data set with mixed variable types is analysed, and the results are compared to those obtained with iForest.

2 Decomposable Mixed Graphs

We focus on undirected graphs, i.e. graphical models for which the edges in the interaction graph are not directed. Such models are also known as Markov random networks. In the following, we introduce notation and concepts that can be found in e.g. Lauritzen (1996). An undirected mixed graph, $G = (V, E)$, is a pair consisting of a set of vertices V and a set of edges $E = \{\{u, v\} \mid u, v \in V, u \neq v\}$. Furthermore, V is the union of Δ, the discrete variables, and Γ, the continuous variables, where $|\Delta| = s, |\Gamma| = r$ and $m = |V| = s + r$. In the figures, we use circles to represent continuous variables and dots (discs) to represent discrete variables.

A mixed graph is triangulated if it has no cycle of length ≥ 4 without a chord. Mixed graphs are said to be decomposable if they are triangulated and do not contain any path between two non-adjacent discrete vertices passing through continuous variables only. Such paths are also called forbidden paths. For a decomposable mixed graph, G, with vertex set V, it holds that the sub-graph $G_A = (A, \{\{u, v\} \mid u, v \in A\, u \neq v\})$ is also decomposable for all subsets $A \subseteq V$. The subset A is called a clique if G_A is a complete graph, i.e. any two vertices are adjacent. Define the star graph of $G = (V, E)$ as

$$G^\star = (V^\star, E^\star) = \big(V \cup \{\star\}, E \cup \{\{d, \star\} \mid d \in \Delta\}\big).$$

That is, $G^\star$ is the graph, in which V is extended with the $\star-$node, and all discrete vertices are connected to this. Leimer (1988) showed, that a graph is decomposable if and only if the corresponding star graph is triangulated. It can, therefore, be checked if a graph G is decomposable by using the maximum cardinality search (MCS) algorithm (Yannakakis, 1981) on $G^\star$.

Let $C_1, C_2, \dots, C_k$ be a sequence of the cliques in an undirected graph, and define for $j = 1, 2 \dots, k$

- $H_j = C_1 \cup \cdots \cup C_j$,
- $S_j = C_j \cap H_{j-1}$ and
- $R_j = C_j \setminus H_{j-1}$,

where we define H_0 as the empty set. These sets are also referred to as the histories, separators and residuals, respectively. The sequence is then said to be perfect if the following conditions hold:

(a) for all $j > 1$, there exist an index $i < j$ such that $S_j \subseteq C_i$,

(b) all separators are complete, and

(c) either $R_j \subseteq \Gamma$ or $S_j \subseteq \Delta$ for all $j > 1$.

Condition (a) is known as the running intersection property, and condition (c) ensures that no forbidden path exists. Denote by $\mathrm{ne}(v) = \{u \mid \{u,v\} \in E\}$ and $\mathrm{cl}(v) = \mathrm{ne}(v) \cup \{v\}$ the neighbours and closure of the vertex $v \in V$, respectively, and define

$$B(v_j) = \mathrm{cl}(v_j) \cap \{v_1, v_2, \ldots, v_j\}, \quad j > 1. \tag{1}$$

If the sets in (1) form a perfect sequence of sets, the sequence of vertices, $v_1, v_2, \ldots, v_m$, is said to be a perfect numbering of the vertices. A graph is decomposable if and only if, the vertices admit a perfect numbering and/or the cliques of the graph can be perfectly numbered to form a perfect sequence (Lauritzen, 1996). The cliques are then said to have the running intersection property (RIP).

We use the notation $\mathrm{pa}(v) := B(v) \setminus \{v\}$ to denote the parents of v defined as the preceding numbered vertices of v that are also neighbours of v. The following result appears in Lauritzen (1996, p. 18) as a remark. However, this result is vital for the model assumptions in Section 4, and we therefore give a concise formal proof.

Proposition 3. *For decomposable mixed graphs, a perfect numbering of the vertices can be chosen such that the discrete variables are numbered before the continuous ones.*

Proof. Let δ be a discrete vertex numbered after the continuous vertex γ. By definition, $\gamma \notin \mathrm{pa}(\delta)$ and $\delta \notin \mathrm{pa}(\gamma)$. Hence, interchanging δ and γ in the perfect sequence would leave $B(\gamma)$ and $B(\delta)$ unchanged and, thus, complete and satisfy $B(\delta) \subseteq \Delta$. After a suitable number of such interchanges, all continuous vertices will be preceded by the discrete vertices. □

Consider the mixed graph in Figure 1 (left), where $V = \{a,b,c,d,e,f\}$, and

$$E = \big\{\{a,b\},\{b,c\},\{b,d\},\{c,d\},\{c,e\},\{d,e\}\big\},$$

with $\Delta = \{b,c,f\}$ and $\Gamma = \{a,d,e\}$. The corresponding star graph is depicted in Figure 1 (right), where it can be seen that the graph is triangulated and hence decomposable. A perfect numbering of the vertices is given as c,b,f,d,e,a.

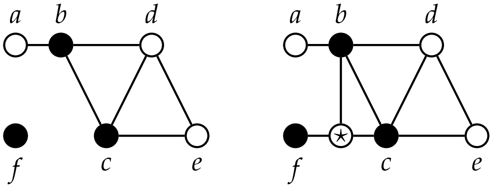


Fig. 1: A decomposable mixed graph (left) and its corresponding star graph (right), where ● represents discrete variables and ○ represents continuous variables, respectively.

4 Notation and the Likelihood Function

Let I be a $s-$dimensional discrete random vector and Y a $r-$dimensional real random vector. A realised value of the random vector $X = (I,Y)$ is denoted $x = (i,y)$, where $i = (i_d)_{d \in \Delta}$ is a tuple of discrete outcomes, also referred to as cell i, and $y = (y_\gamma)_{\gamma \in \Gamma}$ is a real-valued vector. The state space of I is denoted as $\mathcal{I} = \times_{d \in \Delta} \mathcal{I}_d$, where $\mathcal{I}_d$ is the level set of d; hence $i \in \mathcal{I}$. Marginal vectors are written $x_a = (i_{a \cap \Delta}, y_{a \cap \Gamma})$, where $i_{a \cap \Delta}$ and $y_{a \cap \Delta}$ are sub-vectors restricted to the sets $a \cap \Delta$ and $a \cap \Gamma$, respectively. We usually use the shorthand notations $i_a := i_{a \cap \Delta}$ and $y_a := y_{a \cap \Gamma}$. The level set for the $a-$marginal i_a is denoted $\mathcal{I}_a$; hence $i_a \in \mathcal{I}_a = \times_{d \in a} \mathcal{I}_d$. If the vertices are numbered, we write

$$x = (x_1, x_2, \dots, x_m) = (i_1, i_2, \dots, i_s, y_1, y_2, \dots, y_r).$$

The observed counts in cell i is denoted $n(i)$, and the probability of an observation falling in cell i is $p(i)$. The $a-$marginal table is then defined by the counts

$$n_a(i_a) = \sum_{j : j_a = i_a} n(j),$$

and similarly for the marginal probabilities p_a. For the empty set, $a = \emptyset$, we write $n(i_\emptyset) = |n|$, where $|n| = \sum_{i \in \mathcal{I}} n(i)$ is the total number of counts.

We assume that I has probability mass function p, and that Y, given $I = i$, is a multivariate Gaussian model with mean and variance depending on cell i. A distribution of this form is called an inhomogeneous conditional Gaussian (CG) distribution (Lauritzen and Wermuth, 1989). The joint density is written as

$$f(x) = f(i,y) = f(y \mid i) \times p(i). \tag{2}$$

If the variance is assumed to be independent of the discrete variables, the model is referred to as a homogeneous CG distribution. The inhomogeneous case is treated below (with some additional details in Appendix A). The homogeneous case is discussed in Section 6.2. Given a perfect numbering of the vertices, Proposition 3 allows the following factorisation of the joint density

$$f(x) = p(i) \times \prod_{j=1}^{r} f(y_j | x_{\mathrm{pa}_j}), \tag{3}$$

with $\mathrm{pa}_j{:=}\mathrm{pa}(y_j)$. The univariate conditional densities in the product (3) are still Gaussian with a conditional mean depending on the parents. Such models are called CG regressions, see e.g. Edwards (2012). One particular useful feature of decomposable models is, that the maximum likelihood estimates of the parameters in (2) can be obtained from those in (3). We now derive the maximised likelihood of (3), which is then exploited in Section 5 in order to arrive at a test statistic to be used in connection with outlier detection.

Suppose we have a sample of i.i.d. observations, $x^\ell = (i^\ell, y^\ell)$ for $\ell = 1, 2, \dots, |n|$, from a decomposable mixed graphical model, and let $x = (x^1, x^2, \dots, x^{|n|})$ be the vector of observations. The likelihood of the j'th Gaussian factor then takes the form

$$L(\theta_j; x) = \prod_{\ell=1}^{|n|} f(y_j^\ell \mid x_{\mathrm{pa}_j}^\ell) = \left(\prod_{\ell=1}^{|n|} \sigma^2(i_{\mathrm{pa}_j})^{-1/2}\right) \times \exp\left\{-\frac{1}{2}\sum_{\ell=1}^{|n|} \sigma^{-2}(i_{\mathrm{pa}_j})(y_j^\ell - \mu(x_{\mathrm{pa}_j}^\ell))^2\right\}, \tag{4}$$

where $\mu(x_{\mathrm{pa}_j})$ and $\sigma^2(i_{\mathrm{pa}_j})$ are the conditional variance and mean of Y_j given $X_{\mathrm{pa}_j} = x_{\mathrm{pa}_j}$, respectively, and θ_j is the set of parameters. Note, that the variances only depend on the cell values. From here, we simply write $\mu_j(x) := \mu(x_{\mathrm{pa}_j})$ and $\sigma_j(i) := \sigma(i_{\mathrm{pa}_j})$ to ease notation. The means are assumed to have linear parameterisations of the form

$$\mu_j(x) = \alpha_j(i) + \beta_j^T(i) y_{\mathrm{pa}_j}, \quad \text{for } i \in \mathcal{I}_{\mathrm{pa}_j},$$

where α_j is a real-valued function of the cells, and β_j is a real-valued vector function of the cells with dimension $|\mathrm{pa}_j \cap \Gamma|$. Define the subset of observations in cell i_a by $\eta(a) := \{\ell \mid i_a^\ell = i_a\}$. Then, the likelihood in (4) can be written as the product of simple Gaussian likelihoods

$$L(\theta_j;x) = \left(\prod_{i\in\mathcal{I}_{\mathrm{pa}_j}} \sigma_j^2(i)^{-n_{\mathrm{pa}_j}(i)/2}\right) \times \exp\left\{-\frac{1}{2}\sum_{i\in\mathcal{I}_{\mathrm{pa}_j}} \sigma_j^{-2}(i) \sum_{k\in\eta(\mathrm{pa}_j)} (y_j^k - \mu_j(x^k))^2\right\}$$

$$= \prod_{i\in\mathcal{I}_{\mathrm{pa}_j}} \left(\sigma_j^2(i)^{-n_{\mathrm{pa}_j}(i)/2} \times \exp\left\{-\frac{1}{2\sigma_j^2(i)} \sum_{k\in\eta(\mathrm{pa}_j)} (y_j^k - \mu_j(x^k))^2\right\}\right),$$

implying that the sum of squares depends on a particular cell as $x^k = (i^k, y^k)$. The set of parameters can be written as $\theta_j = \cup_{i\in\mathcal{I}_{\mathrm{pa}_j}} \{\alpha_j(i), \beta_j(i), \sigma_j^2(i)\}$. When $\mathrm{pa}_j \cap \Delta = \emptyset$, we define $\mathcal{I}_\emptyset$ as the empty set and $\eta(\emptyset) := \{1,2\ldots,|n|\}$. In this case, the likelihood reduces to the ordinary Gaussian likelihood, where the mean only depends on continuous variables, and the variance is homogeneous. That is,

$$L(\theta_j;x) = (\sigma_j^2)^{-|n|/2} \times \exp\left\{-\frac{1}{2\sigma_j^2} \sum_{\ell=1}^{|n|} (y_j^\ell - \mu_j(y^\ell))^2\right\},$$

where $\theta_j := \{\alpha_j, \beta_j, \sigma_j^2\}$. For the pure discrete factors, the likelihood is denoted by

$$L(p;n) = \prod_{i\in\mathcal{I}} p(i)^{n(i)}, \tag{5}$$

where n is the table of counts and $p = \{p(i)\}_{i\in\mathcal{I}}$. Hence, the complete likelihood is given by

$$L(\theta;x) = L(p;n)\prod_{j=1}^{r} L(\theta_j;x), \tag{6}$$

where $\theta = \cup_{j=1}^r \{\theta_j\} \cup \{p\}$. Finally, using standard results for linear normal models, the maximum of the likelihood takes the form

$$L(\hat{\theta};x) = \prod_{j=1}^{r}\left(\prod_{i\in\mathcal{I}_{\mathrm{pa}_j}} \hat{\sigma}_j^2(i)^{-n_{\mathrm{pa}_j}(i)/2} \times \exp\{-n_{\mathrm{pa}_j}(i)/2\}\right) \times \prod_{i\in\mathcal{I}} \hat{p}(i)^{n(i)}, \tag{7}$$

where

$$\hat{\sigma}_j^2(i) = \frac{1}{n_{\mathrm{pa}_j}(i)} \sum_{k\in\eta(\mathrm{pa}_j)} (y_j^k - \hat{\mu}_j(x^k))^2 \qquad \text{and} \qquad \hat{\mu}_j(x^k) = \hat{\alpha}_j(i^k) + \hat{\beta}_j^T(i^k) y_{\mathrm{pa}_j}^k,$$

for $i \in \mathcal{I}_{\mathrm{pa}_j}$ and the linear parameters are estimated by ordinary least squares.

Let $C_1, C_2, \ldots, C_K$ be a sequence of cliques in G_Δ satisfying the RIP ordering. It can then be shown (Lauritzen, 1996) that

$$\hat{p}(i) = \frac{1}{|n|} \frac{\prod_{k=1}^{K} n_{C_k}(i_{C_k})}{\prod_{k=2}^{K} n_{S_k}(i_{S_k})}, \quad \text{for } i \in \mathcal{I},$$

which is the maximum likelihood estimates of (5) as also exploited in the outlier detection model given in Lindskou et al. (2019).

5 The Null Hypothesis and Deviance Test Statistic

We aim to test if the observation $z = (i^0, y^0) := x^{|n|}$ is an outlier, i.e. it deviates significantly from all other observations. Suppose that i^0 is an observation sampled from a distribution, q, different from p, the distribution of $i^1, i^2, \ldots, i^{|n|-1}$ and that $Y_j^0 \mid Z_{\mathrm{pa}_j} = (i^0_{\mathrm{pa}_j}, y^0_{\mathrm{pa}_j}) \sim N(\lambda_j^0, \sigma_j^2(i^0))$. Then, the null hypothesis takes the compound form

$$H_0 : \{\lambda_j^0 = \alpha_j(i^0) + \beta_j(i^0)^T y^0_{\mathrm{pa}_j},\ j = 1, 2, \ldots, r\} \quad \wedge \quad \{q = p\}.$$

Let $E_{0,j} := \{\alpha_j(i), \beta_j(i); i \in \mathcal{I}_{\mathrm{pa}_j}\}$, and define the set of mean parameters under H_0 as $E_0 = \cup_{j=1}^{r} E_{0,j}$. The set of mean parameters under the alternative hypothesis is then given by $E = \cup_{j=1}^{r} E_j$, where $E_j = E_{0,j} \cup \{\lambda_j\}$ and λ_j is a single parameter describing the conditional mean of Y_j^0 given $Z_{\mathrm{pa}_j} = (i^0_{\mathrm{pa}_j}, y^0_{\mathrm{pa}_j})$. Hence, under the alternative hypothesis, the mean λ_j of Y_j^0 is not restricted and is free to vary, and hence $\hat{\lambda}_j = y_j^0$. Let

$$\theta_0 = E_0 \cup \{p\} \cup \{\sigma_j^2(i)\}_{j=1,2,\ldots,r,\ i \in \mathcal{I}_{\mathrm{pa}_j}} \quad \text{and} \quad \theta = E \cup \{q\} \cup \{\sigma_j^2(i)\}_{j=1,2,\ldots,r,\ i \in \mathcal{I}_{\mathrm{pa}_j}}.$$

The likelihood ratio is then given by

$$LR(z) = \frac{L(\hat{\theta}_0; x)}{L(\hat{\theta}; x)}.$$

Using (7), we obtain

$$LR(z) = \prod_{j=1}^{r} \prod_{i \in \mathcal{I}_{\mathrm{pa}_j}} \left(\frac{\hat{\sigma}_j^2(i)}{\hat{\sigma}_{j,0}^2(i)} \right)^{n_{\mathrm{pa}_j}(i)/2} \times \frac{L(\hat{p}; n)}{L(\hat{q}; n)} = \prod_{j=1}^{r} \left(\frac{\hat{\sigma}_j^2(i^0)}{\hat{\sigma}_{j,0}^2(i^0)} \right)^{n_{\mathrm{pa}_j}(i^0_{\mathrm{pa}_j})/2} \times \frac{L(\hat{p}; n)}{L(\hat{q}; n)},$$

by exploiting that the two variance estimates coincide in all cells but $i^0_{\mathrm{pa}_j}$.

Further define

$$Q_j := \hat{\sigma}_j^2(i^0)/\hat{\sigma}_{j,0}^2(i^0), \tag{8}$$

and let the degrees of freedom $df_j = n_{\mathrm{pa}_j}(i^0_{\mathrm{pa}_j}) - |\mathrm{pa}_j \cap \Gamma| - 1$. Using that $E_{0,j}$ and E_j differ by exactly one parameter, λ_j, together with Cochran's theorem (Cochran, 1934), it then follows that

$$Q_j \sim \mathrm{Beta}(df_j/2, 1/2), \quad \text{for } n_{\mathrm{pa}_j}(i^0_{\mathrm{pa}_j}) > |\mathrm{pa}_j \cap \Gamma| + 1. \tag{9}$$

The likelihood ratio for the pure discrete part, $Q_D := L(\hat{p}; n)/L(\hat{q}; n)$, was investigated by Lindskou et al. (2019): Given a RIP ordering $C_1, C_2, \dots, C_K$ of the cliques in G_Δ, it was shown that

$$-2 \log Q_D(z) = -2\left(\sum_{k=1}^{K} H(n_{C_k}(i^0_{C_k})) - \sum_{k=2}^{K} H(n_{S_k}(i^0_{S_k})) - H(|n|) \right), \tag{10}$$

where $H(x) := G(x-1) - G(x)$ and

$$G(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x \log(x) & \text{if } x > 0. \end{cases}$$

The total deviance test statistic for testing H_0 is therefore given by

$$D(z) := -2 \log LR(z) = -\sum_{j=1}^{r} n_{\mathrm{pa}_j}(i^0_{\mathrm{pa}_j}) \log Q_j - 2 \log Q_D(z). \tag{11}$$

The following result state, that the quantities in (8) can be calculated independently when i^0 is known.

Proposition 6. *The quantities $Q_1, Q_2, \dots, Q_r$ are jointly independent given i^0.*

Proof. Notice first, that the distribution of Q_j in (9) is conditional on $x_1^0, \dots, x_{j-1}^0$. But since the distribution only depends on df_j, we conclude that Q_j is independent of $Q_1, Q_2, \dots, Q_{j-1}$ given i^0. Repeating this argument for $j = r, r-1, \dots, 2$, the result follows. □

6.1 A Note on Studentized Residuals

For each j, the ratio Q_j can be used for an outlier test on data that conforms with $i^0_{\mathrm{pa}_j}$ using studentized residuals. Recall, that a studentized residual is of the form

$$r_j^0 = \frac{y_j^0 - \tilde{\mu}_j(z)}{\sqrt{\tilde{\sigma}_j^2(i^0)(1-h_j^0)}} \sim t_{df_j},$$

where h_j^0 is the so called leverage, which is the j'th diagonal element of the hat matrix, and where $\tilde{\sigma}^2$ and $\tilde{\mu}$ are the estimates under the alternative hypothesis, i.e. excluding z. Let $f = (r_j^0)^2$. Then, $f \sim F_{1,df_j}$, and since the Beta distribution is mirror-symmetric, we obtain

$$Q_j := 1 - \frac{f/df_j}{1 + f/df_j} \sim \text{Beta}(df_j/2, 1/2).$$

Hence, the contribution of the j'th ratio Q_j in (11) is large when $|r_j^0|$ is large, i.e. when y_j^0 is deviating from the expectation under H_0 in cell $i^0_{\text{pa}_j}$.

6.2 The Homogeneous Case

In the homogeneous case, the conditional variance of Y_j given $X_{\text{pa}_j} = (i_{\text{pa}_j}, y_{\text{pa}_j})$ is assumed to be independent of the discrete parents. That is, it is assumed that $\sigma_j^2(i) = \sigma_j^2$ for all $i \in \mathcal{I}_{\text{pa}_j}$. It follows that the maximised likelihood function in (7) reduce to

$$\prod_{j=1}^{r} (\hat{\sigma}_j^2)^{-|n|/2} \times \exp\{-|n|/2\} \times \prod_{i \in \mathcal{I}} \hat{p}(i)^{n(i)},$$

where

$$\hat{\sigma}_j^2 = \frac{1}{|n|} \sum_{i \in \mathcal{I}_{\text{pa}_j}} \sum_{k \in \eta(\text{pa}_j)} \left(y_j^k - \hat{\mu}_j(x^k)\right)^2 \quad \text{and} \quad \hat{\mu}_j(x^k) = \hat{\alpha}_j(i^k) + \hat{\beta}_j^T y^k_{\text{pa}_j}. \qquad (12)$$

Notice, that $\hat{\beta}_j$ does not depend on any cells, since otherwise the marginal variance would not be independent of the discrete parents. Under the null hypothesis, the likelihood ratio now takes the form

$$\prod_{j=1}^{r} \left(\frac{\hat{\sigma}_j^2}{\hat{\sigma}_{j,0}^2}\right)^{-|n|/2} \times Q_D.$$

Let $Q_j^h = \hat{\sigma}_j^2/\hat{\sigma}_{j,0}^2$, and define $df_j^h = |n| - |\mathrm{pa}_j \cap \Gamma| - |\mathcal{I}_{\mathrm{pa}_j}^+|$, where $\mathcal{I}_{\mathrm{pa}_j}^+$ is the non-zero cells in $\mathcal{I}_{\mathrm{pa}_j}$, $j = 1, \ldots, r$. Then

$$Q_j^h \sim \mathrm{Beta}(df_j^h/2, 1/2),$$

when $df_j^h > 0$. In the case where $\mathrm{pa}_j \cap \Delta = \emptyset$, the degrees of freedom coincide in the homogeneous and inhomogeneous case, such that $df_j^h = df_j = |n| - |\mathrm{pa}_j \cap \Gamma| - 1$.

6.3 Evaluating Deviances

In order to evaluate the deviance, $D(z)$, for a new observation, z, in the inhomogeneous case, one must compute the variance estimates $\hat{\sigma}_{j,0}^2(i^0)$ and $\hat{\sigma}_j^2(i^0)$ to obtain Q_j for $j = 1, 2, \ldots, r$. Similarly, in the homogeneous case, the estimates $\hat{\sigma}_j^2$ and $\hat{\sigma}_{j,0}^2$ must be computed to obtain Q_j^h for $j = 1, 2, \ldots, r$. In the following, we give efficient methods for the calculations.

Inhomogeneous Case

A natural way of estimating the variances in the inhomogeneous case is by fitting two linear regression models, one under the null hypothesis and one under the alternative hypothesis. Exploiting the connection to studentized residuals, it is only required to fit a single linear regression model under the alternative hypothesis (i.e. excluding z) and then calculate the quantities $Q_j = 1 - (f/df_j)/(1 + f/df_j)$ as explained in Section 6.1.

Homogeneous Case

In order to estimate the variances in the homogeneous case using linear regression, $(|\mathrm{pa}_j \cap \Gamma| + |\mathcal{I}_{\mathrm{pa}_j}^+|) \times (|\mathrm{pa}_j \cap \Gamma| + |\mathcal{I}_{\mathrm{pa}_j}^+|)$–dimensional matrices must be inverted. Such inversions can be expensive even when $|\mathrm{pa}_j|$ is small since $|\mathcal{I}_{\mathrm{pa}_j}^+|$ may be large if some of the discrete variables have many levels. However, it is not of interest to know the estimated mean parameters; these are only required to estimate the variances and hence calculate Q_j^h. We circumvent this problem by centring the observations. As a consequence, we only need to invert matrices of dimension $|\mathrm{pa}_j \cap \Gamma| \times |\mathrm{pa}_j \cap \Gamma|$. See Appendix A for details.

7 The Outlier Test

In this section, we summarise the results of the previous sections and suggest a novel outlier detection procedure, ODMGM, using CGR models in

Algorithm 2. We first reiterate the method given in Lindskou et al. (2019) for simulating discrete cells in Algorithm 1, which is needed in Algorithm 2. The method is based on a RIP ordering of the cliques in a pure discrete graph and, exploiting, the chain rule

$$P(I = i) = P(I_{C_1} = i_{C_1}) \prod_{k=2}^{K} P(I_{C_k} = i_{C_k} \mid I_{S_k} = i_{S_k}),$$

where the RIP ordering ensures that the cell value i_{S_k} is known, since it holds that $S_k \subset C_j$ for some $j < k$ (this is exploited in line 7 of Algorithm 1).

Algorithm 1 Simulate Cells in Pure Discrete Decomposable Graphical Models

```
 1: procedure ($G$: Pure discrete decomposable graph. $U$: Dataset of observations)
 2:     Form the contingency table $n$ of all observations in $U$
 3:     Construct a sequence of cliques, $C_1, C_2, \dots, C_K$, having RIP from $G$
 4:     Let $i := \{\}$ be an ordered list
 5:     Simulate $i_{C_1}$ using the probability table $n_{C_1}(i_{C_1})/|n|$ and append $i_{C_1}$ to $i$
 6:     for $k = 2, 3, \dots, K$ do
 7:         Simulate $i_{C_k \setminus S_k}$ using the table $n_{C_k}(i_{C_k \setminus S_k}, i_{S_k})/n_{S_k}(i_{S_k})$
 8:         Append $i_{C_k \setminus S_k}$ to $i$
 9:     end for
10: end procedure
```

Notice that, in Algorithm 2 the new observation, z, is appended to the data, U; i.e. under the null hypothesis it is assumed that z originates from the same generating process as all the observations in U. Next, due to the results in (9), it is not necessary to simulate the associated continuous part of each simulated cell in order to simulate the deviances. This implies a large reduction in the computational time needed for simulation. As a consequence of Proposition 6, the quantities Q_j, can be computed in parallel due to conditional independence.

The homogeneous version follows by replacing df_j and Q_j with their respective counterparts, df_j^h and Q_j^h, and replacing all $n_{\mathrm{pa}_j}(i_{\mathrm{pa}_j})$ in line 12 with $|n|$.

8 Real Data Example

In this section, we apply ODMGM to the cover type (CT) data from the UCI Machine Learning Repository (Dua and Graff, 2017). This dataset demands the usage of non-trivial models in order to capture the large amount of information. This has caught the attention of researchers in the machine learning

Algorithm 2 Outlier Detection in Mixed Graphical Models (ODMGM)

```
procedure ($G$: Decomposable mixed graph, $U$: Dataset of observations, $z$: New observation.)
    Append $z$ to $U$ and form the contingency table $n$ of all observations
    Find a perfect ordering of the vertices in $G$
    Construct the pure graph $G_\Delta$
    for $\ell = 1,2,\dots,N$ do
        Simulate cell $i^\ell$ by applying Algorithm 1 on $G_\Delta$
        for $j = 1,2,\dots,r$ do
            if $df_j \geq 0$ then
                simulate $Q_j$ from Beta$(df_j/2, 1/2)$
            end if
        end for
        Calculate $D(x^\ell)$ by applying (10) to cell $i^\ell$ and add it to $-\sum_{j:df_j \geq 0} n_{\mathrm{pa}_j}(i^\ell_{\mathrm{pa}_j}) \log(Q_j)$
    end for
    Let $F(x) = N^{-1} \sum_{\ell=1}^{N} \mathbb{1}[D(x^\ell) \leq D(x)]$, and calculate $D(z)$ using (11)
    if $F(z) \geq 1 - \alpha$ then
        declare $z$ as outlier in $U$ at an $\alpha-$level
    end if
end procedure
```

community in order to benchmark different classification models. Each sample in the data is taken from a $30m \times 30m$ patch of forest that is classified as one of seven CTs represented as integers: 1: Spruce/Fir (37%), 2: Lodgepole Pine (48%), 3: Ponderosa Pine (6%), 4: Cottonwood/Willow (1%), 5: Aspen (2%), 6: Douglas-fir (3%) and 7: Krummholz (4%). In addition, the CT data contains 53 explanatory variables of which 44 are discrete with two levels (i.e. binary). Of these, 40 describe the presence (or absence) of a particular soil type, and four describes the presence (or absence) of the wilderness area. The remaining variables are continuous and includes for example elevation, slope, horizontal distance to hydrology and hillshade at noon. Dua and Graff (2017) gave a thorough explanation of the entire dataset. The original dataset consists of 581,000 samples, however, we have down-sampled to 20,000 samples to keep the CPU running time down while preserving the frequency distribution of the CTs.

Recently, Kumar and Sinha (2020) applied a random forest model to obtain a classification accuracy of 95% when predicting the CT of a sample. We demonstrate, that classification should be conducted with caution since, in many situations, more than a single CT is a statistically plausible explanation of a sample. Furthermore, some authors, e.g. Zhiwei et al. (2017), assumed the explanatory variables to be independent, which we show is an invalid exorbitant assumption.

We start the analysis by fitting an interaction graph using the R package gRapHD (de Abreu et al., 2009) to investigate the complexity of the CT data. The interaction graph, G, is depicted in Figure 2 (left), where white vertices represent continuous variables, black vertices represent discrete variables, and the grey vertex is the class variable. Clearly, the explanatory variables are associated with CT either by a direct relation or implicitly through other explanatory variables. The interaction graph is rather complex and it is thus questionable to assume independence among all variables. There are four isolated variables, i.e. they are not connected to any other variable in the graph. We have removed these four variables (columns 21, 22, 50 and 51 in the UCI dataset) to reduce the complexity.

In order to benchmark ODMGM as an outlier tool, we construct seven interaction graphs, one for each class. Figure 2 shows one of the more complex interaction graphs for class 1 (G_1, middle) and the simplest one for class 7 (G_7, right). Notice, that these are quite different implying that samples from different classes are, most likely, generated by different mechanisms with intrinsic association differences among the explanatory variables.

In Section 8.2, we use these interaction graphs to calculate the proportion of samples that ODMGM is able to declare as outliers in each class. We benchmark the results to those of iForest. In the following section, we investigate if the underlying assumptions of the CGR model in class 7, which was chosen for simplicity due to its simpler interaction structure compared to the other classes, is valid.

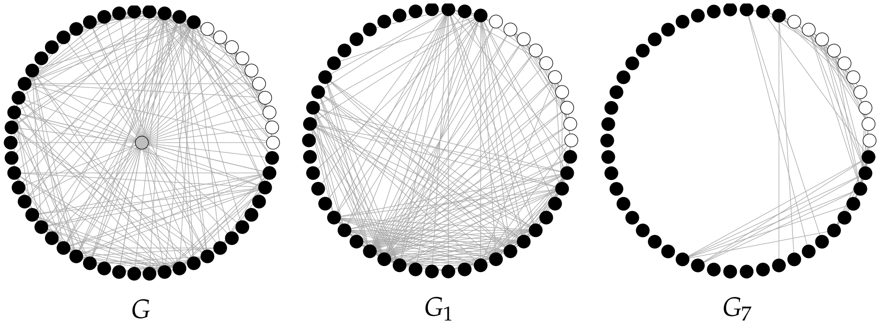


Fig. 2: Left: Interaction graph, G, for the down-sampled CT data including the class variable. Middle: Interaction graph, G_1, for class 1. Right: Interaction graph, G_7, for class 7. Discrete variables are black, continuous variables are white and the central, grey vertex in the left graph represent the class variable.

8.1 Verifying CGR Assumptions

Consider the subgraph H_7 of G_7 in Figure 3, where we have named the vertices according the column position of the corresponding variables in the cover type data.

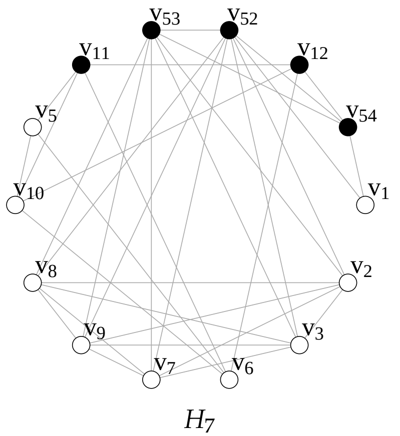


Fig. 3: The subgraph H_7 of G_7 illustrating some of the associations between the CT variables for class 7.

The subgraph H_7 in Figure 3 consists of the continuous variables and the corresponding discrete parents. Using MCS together with Proposition 3, a perfect numbering of the vertices can be computed such that $\text{pa}(v_1) = \{v_{52}, v_{54}\}$, $\text{pa}(v_5) = \{v_6, v_{10}, v_{11}\}$, $\text{pa}(v_7) = \{v_2, v_3, v_{52}, v_{53}\}$, $\text{pa}(v_8) = \{v_2, v_3, v_7, v_{52}, v_{53}\}$, $\text{pa}(v_9) = \{v_2, v_3, v_7, v_8, v_{52}, v_{53}\}$ and $\text{pa}(v_{10}) = \{v_{11}, v_{12}\}$. First, we make a graphical check for v_1 and v_{10} being approximately Gaussian given their parents, see Figure 4. In the light of a rather complex model, the density plots in Figure 4 look fairly symmetric and bell-shaped for v_1 (top row, Figure 4). For v_{10}, the densities are neither symmetric nor bell-shaped, but the deviations from the Gaussian distribution are not large (bottom row, Figure 4).

It is difficult graphically to verify, whether v_5, v_7, v_8 and v_9 are Gaussian with mean values depending on their parents, since they all have more than one continuous parent. Instead, we shall assess the adequacy of the assumptions simply by calculating the squared coefficient of determination, R^2, for each combination of the discrete parents. The results are summarised in Table 1, where the numbers represent the values of R^2 for the given configurations of v_{52} and v_{54}. It can be noticed, that no samples had the configuration $(v_{52}, v_{54}) = (1,1)$. The values of R^2 are overall satisfactory. Notice, that the values for the model of v_9 indicates a nearly perfect linear association. The model of v_5 has R^2-values of 0.011 and 0.258 for $v_{11} = 0$ and $v_{11} = 1$, respectively which is less impressive.

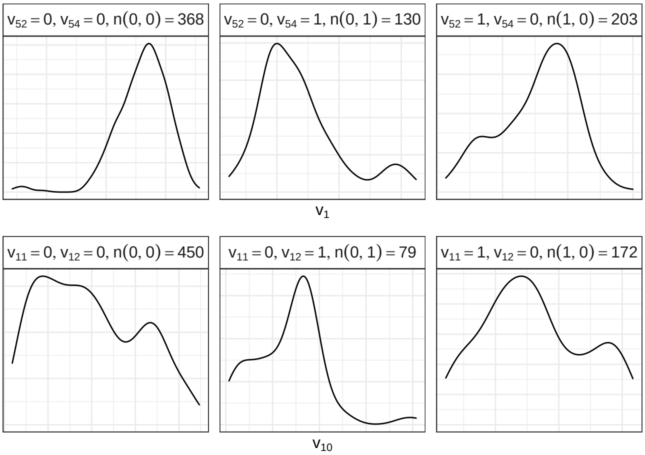


Fig. 4: Density plots of v_1 (top row) and v_{10} (bottom row) given the configurations of their discrete parents. The panel headers indicate the level of the discrete parents, e.g. $v_{52} = 0, v_{54} = 0$, and the numbers of observations, e.g. $n(0,0) = 368$.

Model	$(v_{52}, v_{54}):$	(0,0)	(1,0)	(0,1)
$v_7 \sim v_2 + v_3$		0.174	0.580	0.396
$v_8 \sim v_2 + v_3 + v_7$		0.575	0.405	0.599
$v_9 \sim v_2 + v_3 + v_7 + v_8$		0.992	0.998	0.988

Table 1: Summary of model performance using the squared coefficient of determination, R^2.

8.2 Performance

We now apply both ODMGM and iForest to the CT data and summarise the results in Figure 5. Given a specific class, one of the facets, we calculate the proportion of observations for all other classes that ODMGM and iForest, respectively, are able to declare as outliers in that specific class. Proportions for ODMGM are shown by circles whereas results for iForest are shown as filled dots. The grey bands highlight the proportion of in-class outlier detection; proportions in this band should optimally equal the significance level, here, 0.05. Notably, it is difficult to detect outliers in class 1 and 2 regardless of which method is used. In general, though, ODMGM outperforms iForest and iForest is in fact worse than random guessing in the majority of the tests (many rejection fractions less than 0.5). Specifically, for class 7 the difference in performance is heavily pronounced.

The presented methodology allows each observation to be tested as outlier in each of the classes in a dataset. Consequently, an observation can be declared an outlier in no, some, or all classes. In the cover type dataset with

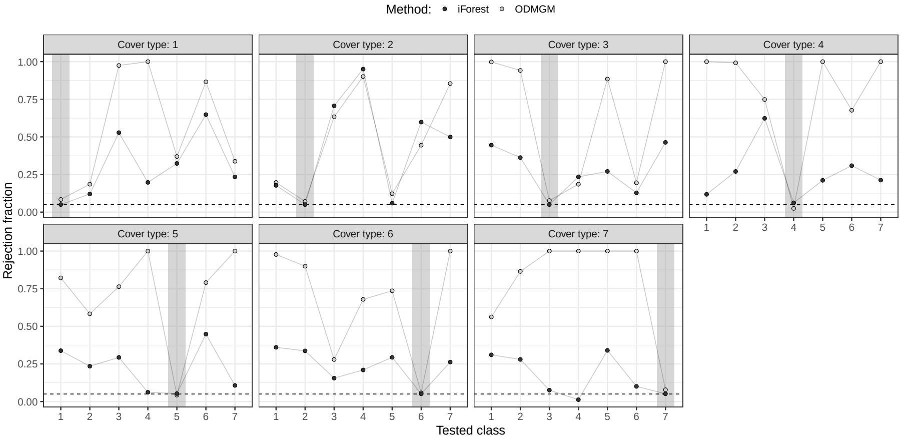


Fig. 5: Each observation in the down-sampled dataset was tested as outlier in each class using both iForest (dark points) and ODMGM (light points), respectively. The panels show how often each of the cover types is rejected in the classes. The grey bands highlights the proportion of in-class outlier detection.

seven classes, this implied that there may be up to $2^7 = 128$ different rejection/acceptance combinations. In Figure 6, we plotted the different types of combinations seen in the down-sampled dataset (created by the `UpSetR` `R`-package, Gehlenborg, 2019). There are 819 observations that were rejected in all classes and 1,784 observations accepted in a single class. Furthermore, as expected from Figure 5, many observations are simultaneously accepted in both class 1 and 2 (e.g. 5,479 in just those two, and 6,931 with an additional class (class 5: 3,639, class 7: 3,151, and class 6: 141, respectively). The aggregation in Figure 6 does not take the true class into consideration as in Figure 5. However, since the majority of the observations are of class 1 or 2 (85%), these also belong to the classes in which most observations are accepted. Furthermore 77% ($n = 15{,}365$) and 80% ($n = 16{,}078$) were accepted as being of class 1 and 2, respectively. This emphasises the risk of assigning an observation to a single class, which is the typical approach in a classification setup.

9 Conclusions and Future Work

In this paper, we present a new probabilistic method, ODMGM, for outlier detection in high-dimensional data with mixed variables. The methodology uses a theoretically sound formulation of what is meant by an outlier. We studied the performance of ODMGM on a real data set and benchmarked it with the performance of the state-of-the-art algorithm iForest. We found that

ODMGM was superior to iForest and that iForest, in general, is worse than random guessing for outlier detection for this particular dataset. This contradicts the findings of several other authors, see e.g. Eiras-Franco et al. (2019), Xu et al. (2019), Aryal et al. (2016, 2019), Domingues et al. (2018), Emmott et al. (2015). Furthermore, we saw from Figure 6 that in many cases, it is not, statistically, possible to assign a given sample to a single class. This is in contrast to classification methods where, if a sample is plausible to originate from two or more different classes, the method assign the sample to the most probable class, which may arguably be undesirable in healthcare and forensics e.g. where the cost of a false positive may be fatal. Thus, we suggest an outlier detection method like ODMGM, and leave the further investigation to a specialist if it is plausible for a sample to belong to several classes or to be excluded from all classes. The latter case may reveal a new interesting finding.

Furthermore, we provide software for use in the R language (Lindskou, 2020) together with all code snippets used to generate all the results in this paper.

Another approach in the homogeneous case, which we hope to investigate in future research is to assume a different parameterisations of the conditional mean value in (12). Consider a generic continuous variable, y, and suppose the discrete parents consist of $i = (i_1, \dots, i_k)$. The conditional mean of y could

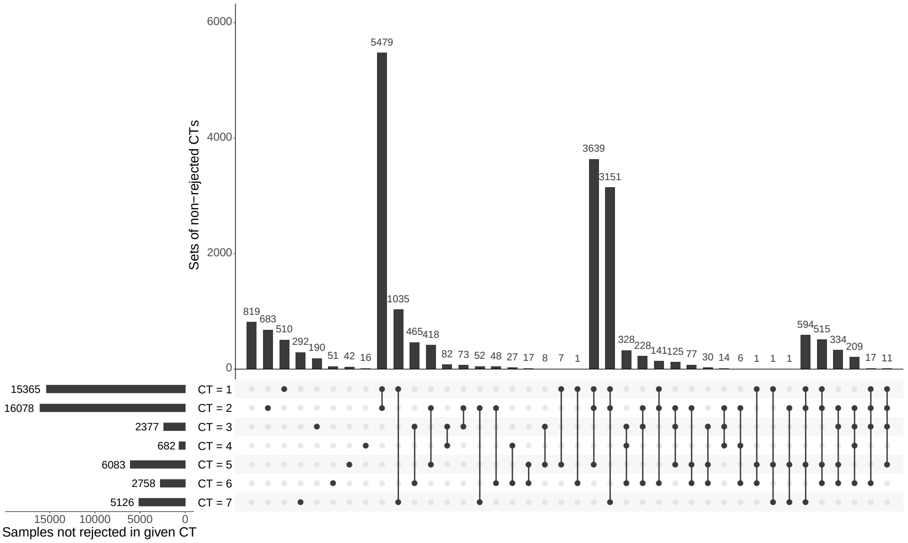


Fig. 6: Diagram showing the distribution of the various combinations of accepted classes for the down-sampled CT dataset. Dots (and lines) in the lower part correspond to the accepted CT classes (created by the `UpSetR` R-package, Gehlenborg, 2019).

then, for example, only include main effects, i.e.

$$\alpha_1(i_1) + \alpha_2(i_2) + \cdots + \alpha_k(i_k) + \beta^T y_{\text{pa}(y)}.$$

In this setup, we only require $|n| \geq \sum_j^k |\mathcal{I}_j| - k + |\text{pa}(y) \cap \Gamma| + 1$. All though much simplified, the estimates would be more robust and the model, if appropriate, will have more power.

Learning a graphical model from high dimensional data is a notoriously hard task, not least because of the many possible structures and the vast amount of data. However, it is more tangible if the graph is assumed to be decomposable since the computational advantages of such an assumption are tremendous. One of the most promising approaches was suggested by Deshpande et al. (2001) which offered a detailed algorithm, named ESS, for efficient stepwise model selection in mixed graphical models (including the pure case) is given. Altmueller and Haralick (2004) discovered a flaw in ESS and gave a proof for the correction. For the pure discrete case the ESS algorithm is implemented in the `R` software package `ess`, originally a part of the `molic` package (Lindskou, 2019). The ESS algorithm is not yet implemented to handle the mixed case in any known software to our knowledge. The `R` package `gRapHD` (Edwards et al., 2010) was designed for model selection in high-dimensional mixed graphical models. Unfortunately, the package is no longer maintained. To our knowledge, the only maintained `R` package for model selection in the mixed case is the `mgm` package. However, one must specify the highest order of interaction in advance and even for small orders the procedure is much too slow for model selection in high-dimensional data. In connection to outlier detection where the procedure may need to run several times, it is crucial that fitting the interaction graph can be done reasonably fast. We plan to implement the ESS procedure for mixed graphs in the future.

It is well-known, that linear regression models are not robust when data is contaminated with outliers (Yu and Yao, 2017). The robustness of the parameter estimators in linear regression is often characterised by the breakdown point which indicates the proportion of outliers that the estimators can resist. It can be shown, that the breakdown point of OLS estimates is $1/|n|$ which tends to zero when the sample size $|n|$ increases. Since the estimates in ODMGM are calculated within sub-tables with $n_{\text{pa}_j}(i^0_{\text{pa}_j})$ observations, the effective sample size is markedly decreased. Hence, in a way, ODMGM is more robust against outliers compared to a global outlier test that uses all $|n|$ observations for parameter estimation. Practical computations of robust estimates are challenging and therefore increases the computational time. We hope to explore the issue of robustness in connection to ODMGM in more detail in the future to make the method more robust.

A Variance Estimation for Inhomogeneous Models

First, we need a little more notation, and to ease this, we define $a := \mathrm{pa}_j$. Denote by $y_j^k(i_a)$ the k'th observation of y_j in cell i_a, i.e. the observation of y_j corresponding to the k'th index in $\eta(a)$. Similarly, denote by $y_a^k(i_a)$ the k'th observation of y_a in cell i_a. The centred observations in cell i_a is then given as

$$\gamma_j^k(i_a) := y_j^k(i_a) - \bar{y}_j(i_a), \quad \text{where} \quad \bar{y}_j(i_a) = \sum_{k \in \eta(a)} y_j^k(i_a),$$

and

$$\gamma_a^k(i_a) := y_a^k(i_a) - \bar{y}_a(i_a), \quad \text{where} \quad \bar{y}_a(i_a) = \sum_{k \in \eta(a)} y_a^k(i_a).$$

Our goal is to minimise the sums of squared errors

$$SSE_j(\alpha, \beta) = \sum_{i_a \in \mathcal{I}_a} \sum_{k \in \eta(a)} \left(y_j^k(i_a) - \alpha(i_a) - \beta^T y_a^k(i_a)\right)^2.$$

Using the centred observations, it can be seen that

$$SSE_j(\alpha, \beta) = \sum_{i_a \in \mathcal{I}_a} \sum_{k \in \eta(a)} \left(\gamma_j^k(i_a) - \beta^T \gamma_a^k(i_a)\right)^2 + \sum_{i_a \in \mathcal{I}_a} \sum_{k \in \eta(a)} \left(\bar{y}_j(i_a) - \alpha(i_a) - \beta^T \bar{y}_a(i_a)\right)^2.$$

Thus, $\hat{\alpha}(i_a) = \bar{y}_j(i_a) - \beta^T \bar{y}_a(i_a)$ and $\hat{\beta} = S_{a,a}^{-1} S_{a,j}$ where

$$S_{u,v} = \sum_{i_a \in \mathcal{I}_a} \sum_{k \in \eta(a)} \gamma_u^k(i_a) \{\gamma_v^k(i_a)\}^T.$$

Finally, it follows that

$$SSE_j(\hat{\alpha}, \hat{\beta}) = S_{j,j} - S_{j,a} S_{a,a}^{-1} S_{a,j} \quad \text{and} \quad \hat{\sigma}_j = |n|^{-1} SSE_j(\hat{\alpha}, \hat{\beta}). \tag{13}$$

The problem is now reduced to inverting a $(|\mathrm{pa}_j \cap \Gamma|) \times (|\mathrm{pa}_j \cap \Gamma|)$ matrix. The quantities Q_j^h can thus be computed using (13) twice; one with the new observation included and one without. Notice that the minimised sums of squared errors reduces as follows in the special cases:

$$SSE_j = \begin{cases} S_{j,j} & \text{when} \quad \mathrm{pa}_j \subset \Delta \\ \tilde{S}_{j,j} - \tilde{S}_{j,a} \tilde{S}_{a,a}^{-1} \tilde{S}_{a,j} & \text{when} \quad \mathrm{pa}_j \subset \Gamma \\ \tilde{S}_{j,j} & \text{when} \quad \mathrm{pa}_j = \emptyset, \end{cases} \tag{14}$$

where $\tilde{S}_{u,v} = \sum_{\ell=1}^{|n|} \gamma_u^\ell (\gamma_v^\ell)^T$ and $\gamma_u^\ell = y_u^\ell - \bar{y}_u$.

References

Altmueller, S. M. and Haralick, R. M. (2004). Practical aspects of efficient forward selection in decomposable graphical models. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 710–715. IEEE.

Aryal, S., Baniya, A. A., and Santosh, K. (2019). Improved histogram-based anomaly detector with the extended principal component features. *arXiv preprint arXiv:1909.12702*.

Aryal, S., Ting, K. M., and Haffari, G. (2016). Revisiting attribute independence assumption in probabilistic unsupervised anomaly detection. In *Pacific-Asia Workshop on Intelligence and Security Informatics*, pages 73–86. Springer.

Cochran, W. G. (1934). The distribution of quadratic forms in a normal system, with applications to the analysis of covariance. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 30, pages 178–191. Cambridge University Press.

de Abreu, G. C., Labouriau, R., and Edwards, D. (2009). High-dimensional graphical model search with graphd r package. *arXiv preprint arXiv:0909.1234*.

Deshpande, A., Garofalakis, M., and Jordan, M. I. (2001). Efficient stepwise selection in decomposable models. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 128–135. Morgan Kaufmann Publishers Inc.

Domingues, R., Filippone, M., Michiardi, P., and Zouaoui, J. (2018). A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74:406–421.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Edwards, D. (2012). *Introduction to graphical modelling*. Springer Science & Business Media.

Edwards, D., de Abreu, G. C., and Labouriau, R. (2010). Selecting high-dimensional mixed graphical models using minimal aic or bic forests. *BMC Bioinformatics*, 11(1):18.

Eiras-Franco, C., Martinez-Rego, D., Guijarro-Berdinas, B., Alonso-Betanzos, A., and Bahamonde, A. (2019). Large scale anomaly detection in mixed numerical and categorical input spaces. *Information Sciences*, 487:115–127.

Emmott, A., Das, S., Dietterich, T., Fern, A., and Wong, W.-K. (2015). A meta-analysis of the anomaly detection problem. *arXiv preprint arXiv:1503.01158*.

Garchery, M. and Granitzer, M. (2018). On the influence of categorical features in ranking anomalies using mixed data. *Procedia Computer Science*, 126:77–86.

Gehlenborg, N. (2019). *UpSetR: A More Scalable Alternative to Venn and Euler Diagrams for Visualizing Intersecting Sets*. R package version 1.4.0.

"Hawkins, D. M. ("1980"). *"Identification of outliers"*, volume "11". "Springer".

Kumar, A. and Sinha, N. (2020). Classification of forest cover type using random forests algorithm. In *Advances in Data and Information Sciences*, pages 395–402. Springer.

Lauritzen, S. L. (1996). *Graphical models*, volume 17 of *Oxford Statistical Science Series*. Clarendon Press.

Lauritzen, S. L. and Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The annals of Statistics*, pages 31–57.

Leimer, H.-G. (1988). Triangulated graphs with marked vertices. In *Annals of Discrete Mathematics*, volume 41, pages 311–324. Elsevier.

Lindskou, M. (2019). molic: An R package for multivariate outlier detection in contingency tables.

Lindskou, M. (2020). mlindsk/odmgm v1.1. `https://doi.org/10.5281/zenodo.3999522` and `https://github.com/mlindsk/odmgm`.

Lindskou, M., Eriksen, P. S., and Tvedebrink, T. (2019). Outlier detection in contingency tables using decomposable graphical models. *Scandinavian Journal of Statistics*.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE.

Xu, H., Wang, Y., Wang, Y., and Wu, Z. (2019). Mix: A joint learning framework for detecting both clustered and scattered outliers in mixed-type data. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1408–1413. IEEE.

Yannakakis, M. (1981). Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79.

Yu, C. and Yao, W. (2017). Robust linear regression: A review and comparison. *Communications in Statistics-Simulation and Computation*, 46(8):6261–6282.

Zhiwei, Z., Xiaomin, G., Lin, W., Mengmeng, Z., Zhikang, L., Yamin, Z., and Jianhua, Z. (2017). Research on search engine of knowledge adaptation system based on large scale data set. *Procedia engineering*, 174:308–316.

Paper C

The **jti** and **sparta** Packages: Junction Tree Inference using Sparse Tables with a View Towards High Dimensional Graphical Models

Mads Lindskou, Torben Tvedebrink, Poul Svante Eriksen, Søren Højsgaard and Niels Morling

The paper has been submitted to
Journal of Statistical Software

The layout has been revised.

Abstract

*A graphical model is a multivariate (potentially very high dimensional) probabilistic model, formed by combining lower-dimensional components. Inference (computation of conditional probabilities) is based on message passing algorithms that utilize conditional independence structures. In graphical models for discrete variables with finite state spaces, there is a fundamental problem in high dimensions: A discrete distribution is represented by a table of values, and in high dimensions, such tables can become prohibitively large. In inference, such tables must be multiplied which can lead to even larger tables. The **jti** package meets this challenge by using the package **sparta** by implementing methods that efficiently handle multiplication and marginalization of sparse tables. The two packages are written in the* R *programming language and are freely available from the Comprehensive* R *Archive Network. We show that **jti** is able to handle highly complex graphical models, which are otherwise infeasible due to lack of computer memory, using **sparta** as a back-end for table operations.*

A probability mass function can be represented by a multi-dimensional array. However, for high-dimensional distributions where each variable may have a large state space, lack of computer memory can become a problem. For example, an 80-dimensional random vector in which each variable has 10 levels will lead to a state space with 10^{80} cells. Such a distribution can not be stored in a computer; in fact, 10^{80} is one of the estimates of the number of atoms in the universe. However, if the array consists of only a few non-zero values, we need only store these values along with information about their location. That is, a sparse representation of a table. We describe the R (R Core Team, 2020) package **sparta** (Lindskou, 2021c) for efficient multiplication and marginalization of sparse tables.

The family of graphical models (Pearl, 2014, Lauritzen, 1996, Højsgaard et al., 2012, Maathuis et al., 2018) is vast and includes many different models. In this paper, we consider Bayesian networks and decomposable undirected graphical models. Undirected graphical models are also known as Markov random fields (MRFs). Decomposability is a property ensuring a closed form of the maximum likelihood parameters. Graphical models enjoy the property that conditional independencies can be read off from a graph consisting of nodes, representing the random variables. In Bayesian networks, edges are directed from a node to another and represent directed connections. In a MRF the edges are undirected and these should be regarded as associations between pairs of nodes in a broad sense. A Bayesian network is specified through a collection of conditional probability tables (CPTs) as we outline in Section 1 and a MRF is specified through tables described by the cliques of the graph. Graphical models are used in a diverse range of applications, e.g., forensic identification problems, traffic monitoring, automated general medical diagnosis and risk analysis Andrade and Ferreira (2009), Kumar et al. (2003), Zagorecki et al. (2013), Weber et al. (2012). Finally, we mention that the word 'inference' within the realm of graphical models is synonymous with estimating conditional probabilities.

The R packages **slam** (Hornik et al., 2020) and **tensorr** (Zamora, 2019) do also implement sparse tables (arrays). In fact, **sparta**, **slam**, and **tensorr** have almost the exact same coordinate list representation of sparse tables where: 1) the index vectors of the non-zero elements are stored in an integer matrix, 2) the values in a corresponding vector of doubles and 3) the dimension names is an attribute. In connection with **jti**, it is crucial to be able to convert high-dimensional sparse conditional probability tables represented as R arrays into an equivalent sparse representation. An important difference between **sparta** and **slam**/**tensorr** is that **sparta** allows fast coercion between dense tables (arrays in R) and sparse tables. Neither **tensorr** nor **slam** provides a method for converting a data frame of character or factor variables into a sparse table which is crucial for a data-driven approach using the junction tree algorithm (JTA). Finally, neither **slam** nor **tensorr** implements the algebras on sparse tables as introduced in Section 4, and implemented in **sparta**, facilitating multiplication and marginalizaion of sparse probability mass functions etc.

There are several interesting suggestions in the litterature for exploiting sparsity in tables. Some interesting approaches includes probability trees (PTs) (Boutilier et al., 2013) and value-based potentials (VBPs) (Gómez-Olmedo et al., 2021), which we discuss further in Section 4.3.

We illustrate **sparta** using the **jti** package (Lindskou, 2021b), which implements JTA for discrete variables using the Lauritzen-Spiegelhalter updating scheme (Lauritzen and Spiegelhalter, 1988) with table operations relying on **sparta**. JTA is used for inference in Bayesian networks (BNs). We also show, that multiplication and marginalization of sparse tables are fast compared with standard built-in R functions and is comparable to **gRbase** when the tables are sparse. On the other hand, for high-dimensional tables it is possible that the dense version can not be kept in memory and the sparse version comes to the rescue.

In addition to **jti**, there are to our knowledge three other packages for belief propagation in R; **gRain** (Højsgaard, 2012), **BayesNetBP** (Yu et al., 2020) and **RHugin** (Konis, 2014), where the latter is not on the Comprehensive R Archive Network (CRAN). The only R package on CRAN that has an API for (dense) table operations is **gRbase**, on which **gRain** depends. Some R packages that rely on **gRain** and **gRbase** are **geneNetBP** (Moharil, 2016) and **bnspatial** (Masante, 2020). The **bnclassify** package (Mihaljeviȅ et al., 2018) has an internal lower C++ class implementation of dense conditional probability tables.

Although the paper is not concerned with learning the graph structure, we mention the packages **bnlearn** (**?**), **gRain**, **sparsebn** (Aragam et al., 2019), **deal** (Boettcher and Dethlefsen, 2003) and **bnstruct** (**?**) which can all be used to learn the DAG of a Bayesian network. In addition, **gRain** and **ess** (Lindskou, 2021a) can be used to estimate the interaction graph of a MRF.

The goal of this paper is twofold: Firstly, to provide an efficient back-end for sparse table operations for other R packages dealing with discrete graphical models. Secondly, to provide a new implementation of JTA, namely **jti**, using **sparta** as a back-end. In Section 1, we introduce basic notation and terminology and in Section 2 we motivate the use of sparse tables through JTA. Section 3 serves as a primer to our novel representation of tables and their algebra given in Section 4. In Section 4, we also demonstrate the use of **sparta**. Section 5 outlines how to use **jti** and gives specific examples using two BNs, which are well known from the literature. Moreover, we demonstrate the strength of **sparta** using the **ess** (Lindskou, 2021a) package to fit a decomposable MRF on real data. In Section 6, we show that the trade-off between execution time and memory allocation using **sparta** is acceptable for small and medium-sized tables and comparable to **gRbase** in high dimensional sparse tables. Finally, we mention that **sparta** leverages from the **RcppArmadillo** package (Eddelbuettel and Sanderson, 2014) by implementing compute-intensive procedures in C++ for better run-time performance.

1 Notation and Terminology

Let p be a discrete probability mass function of a random vector $X = (X_v \mid v \in V)$ where V is a set of labels. The state space of X_v is denoted I_v and the state space of X is then given by $I = \times_{v \in V} I_v$. A realized value $x = (x_v)_{v \in V}$ is called a cell. Given a subset A of V, the A-marginal cell of x is the vector, $x_A = (x_v)_{v \in A}$, with state space $I_A = \times_{v \in A} I_v$. A Bayesian Network can be defined as a directed acyclic graph (DAG), see Figure 1, for which each node represents a random variable together with a joint probability of the form

$$p(x) = \prod_{v \in V} p(x_v \mid x_{pa(v)}), \tag{1}$$

where $x_{pa(v)}$ denotes the parents of x_v; i.e. the set of nodes with an arrow pointing towards x_v in the DAG. Also, x_v is a child of the variables $x_{pa(v)}$. Notice, that $p(x_v \mid x_{pa(v)})$ has domain $I_v \times I_{pa(v)}$. Hence, we can encode the conditional probabilities in a table, say $\phi(x_v, x_{pa(v)})$, of dimension $|I_v| \cdot |I_{pa(v)}|$. It is also common in the literature to refer to these tables as potentials and we shall use these terms interchangeably. In general, a potential does not have an interpretation. Sometimes, we also use subscript notation to explicitly show the set of variables on which a potential depends. That is, ϕ_A is a potential defined over the variables X_A. The product $\phi_A \otimes \phi_B$ of two generic tables over A and B is defined cell-wise as

$$(\phi_A \otimes \phi_B)(x_{A \cup B}) := \phi_A(x_A)\phi_B(x_B).$$

In other words, the product is defined over the union of the variables of each of the two potentials. Division of two tables, $\phi_A \oslash \phi_B$, is defined analogously. The marginal table, $\phi_A^{\downarrow B}$, over the variables $B \subseteq A$ is defined cell-wise as

$$\phi_A^{\downarrow B}(x_B) := \sum_{x_{A \setminus B} \in I_{A \setminus B}} \phi_A(x_B, x_{A \setminus B}).$$

Finally, for some $B \subseteq A$, fix x_B^*. The x_B^* slice of $\phi_A(x)$ is then given by

$$\phi_A^{x_B^*}(x_{A \setminus B}) = \phi_A(x_{A \setminus B}, x_B^*).$$

2 Motivation Through Message Passing in Bayesian Networks

Consider the simple DAG given in Figure 1(a), from which the joint density can be read off:

$$p(x_a, x_b, x_c, x_d, x_e) = p(x_c)p(x_a \mid x_c)p(x_b \mid x_a, x_d)p(x_d \mid x_c)p(x_e \mid x_c, x_d) \qquad (2)$$

If, for example, interest is in the joint distribution of (x_a, x_d) we have to sum over x_b, x_c and x_e and exploiting the factorization we could calculate this as

$$p(x_a, x_d) = \sum_{x_c} p(x_c)p(x_a \mid x_c)p(x_d \mid x_c) \sum_{x_e} p(x_e \mid x_c, x_d) \sum_{x_b} p(x_b \mid x_a, x_d).$$

The junction tree algorithm can be seen as an algorithm for automatically factorizing to circumvent the direct summation as described in what follows using a minimal example (a more general and technical exposition of the algorithm can be found in e.g., Højsgaard et al., 2012): First,

- moralize the DAG; i.e. connect nodes that share a common child node,
- remove directions in the DAG to obtain an undirected graph, and
- triangulate the resulting graph.

Moralization ensures that the corresponding parent and child nodes are put in the same maximal clique. A clique is a subset of the nodes for which the

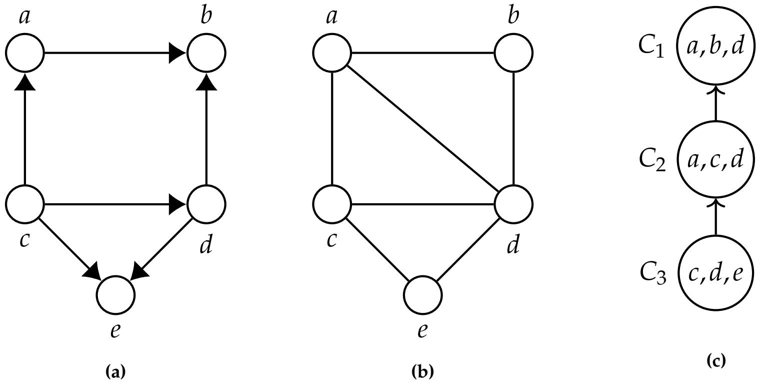


Fig. 1: (a) A DAG. (b) A moralized and triangulated version of (a). (c) A rooted junction tree representation of (b) with root C_1.

induced subgraph is complete, and it is maximal if it is not contained in any other clique. From here, by clique, we always mean a maximal clique, and we refer to these as the cliques of the graph.

A graph is triangulated (or chordal) if it has no cycles of length greater than 3. If such cycles are present, the fill edges must be added to produce a triangulated graph. A triangulated graph is also called decomposable, and hence the connection to decomposable MRFs shows here. Finding an optimal triangulation (in terms of minimizing the number of fill edges) is a NP hard problem, but good heuristic methods exists, see Flores and Gámez (2007). Finding a good triangulation can have a huge impact on the performance of JTA, which we detail in Section 5.2. A moralized and triangulated version of the graph in Figure 1(a) is shown Figure 1(b), where no fill edge was necessary to make the graph triangulated.

A triangulated graph can always be represented as a junction tree. A junction tree is a tree where the nodes are given by the cliques of the triangulated graph with the property that for two cliques, C and C', the intersection $C \cap C'$ is contained in all clique nodes on the unique path between C and C'. This is a consequence of the running intersection property.

The cliques of the graph in Figure 1(b) are given as $C_3 = \{c,d,e\}, C_2 = \{a,c,d\}$ and $C_1 = \{a,b,d\}$ where we arbitrarily designate C_1 as the root to obtain the rooted junction tree in Figure 1(c). Now, assign each potential in (2) to a clique potential for which the variables conform, e.g.,

$$\begin{aligned}\phi_{C_1}(x_a,x_b,x_d) &\leftarrow p(x_a \mid x_b,x_d),\\ \phi_{C_2}(x_a,x_c,x_d) &\leftarrow p(x_c)p(x_a \mid x_c),\\ \phi_{C_3}(x_c,x_d,x_e) &\leftarrow p(x_d \mid x_c)p(x_e \mid x_c,x_d).\end{aligned}$$

The clique potentials are now initialized (the network is also said to be initialized) and note that the clique potentials in general do not have any interpretation at this stage. We have obtained the clique potential representation

$$p(x_a,x_b,x_c,x_d,x_e) = \phi_{C_1}(x_a,x_b,x_d)\phi_{C_2}(x_a,x_c,x_d)\phi_{C_3}(x_a,x_c,x_d). \tag{3}$$

The network is said to be compiled at this stage, i.e. when moralization and triangulization have been performed, and the clique potential representation has been obtained. In complex networks with large clique potentials, it might not be feasible to even initialize the clique potentials due to lack of memory. We give an example of overcoming this in Section 5.2 by entering evidence into the model as described in Section 2.1.

Next, the message passing scheme can now be applied to the junction tree. We describe here, the Lauritzen-Spiegelhalter (LS) scheme which works as follows. Locate a leaf node, here we choose C_3, and find the intersection, $S_{32} = C_3 \cap C_2 = \{c,d\}$, with its parent clique C_2. Then calculate the marginal potential $\phi_{S_{32}}(x_c,x_d) = \sum_{x_e} \phi_{C_3}(x_c,x_d,x_e)$ and perform an inward message by setting

$$\phi_{C_2}(x_a,x_c,x_d) \leftarrow \phi_{C_2}(x_a,x_c,x_d)\phi_{S_{32}}(x_c,x_d)$$

and update the leaf node as

$$\phi_{C_3}(x_c,x_d,x_e) \leftarrow \phi_{C_3}(x_c,x_d,x_e)/\phi_{S_{32}}(x_c,x_d), \tag{4}$$

where $0/0 := 0$. We say that C_2 has collected its messages from all of its children. This procedure must be repeated until the root, C_1, has collected all its messages. Hence, we perform another inwards message by setting $\phi_{S_{21}}(x_a,x_d) = \sum_{x_c} \phi_{C_2}(x_a,x_c,x_d)$ and update:

$$\begin{aligned}\phi_{C_1}(x_a,x_b,x_d) &\leftarrow \phi_{C_2}(x_a,x_c,x_d)\phi_{S_{21}}(x_a,x_d),\\ \phi_{C_2}(x_a,x_c,x_d) &\leftarrow \phi_{C_2}(x_a,x_c,x_d)/\phi_{S_{21}}(x_a,x_d).\end{aligned}$$

The inward phase terminates when the root clique potential has been normalized:

$$\phi_{C_1}(x_a,x_b,x_d) \leftarrow \phi_{C_1}(x_a,x_b,x_d)/\sum_{x_a,x_b,x_d} \phi_{C_1}(x_a,x_b,x_d).$$

To summarize, we have now obtained the set chain representation

$$\begin{aligned} p(x_a, x_b, x_c, x_d, x_e) &= \phi_{C_1}(x_a, x_b, x_d)\phi_{C_2}(x_a, x_c, x_d)\phi_{C_3}(x_c, x_d, x_e) \\ &= p(x_a, x_b, x_d)p(x_a \mid x_c, x_d)p(x_c, x_d \mid x_e), \end{aligned}$$

where the clique potentials are now conditional probability tables. Notice especially that $\phi_{C_1}(x_a, x_b, x_d) = p(x_a, x_b, x_d)$. In the outward phase we start by sending messages from the root by performing an outward message by letting $\phi_{S_{12}}(x_a, x_d) = \sum_{x_b} \phi_{C_1}(x_a, x_b, x_d)$ and update:

$$\phi_{C_2}(x_a, x_c, x_d) \leftarrow \phi_{C_2}(x_a, x_c, x_d)\phi_{S_{12}}(x_a, x_d). \tag{5}$$

We say that C_1 has distributed evidence to C_2. Notice, that ϕ_{C_2} is now identical to the probability distribution defined over the variables $x_a, x_c,$ and x_d. Finally, let $\phi_{S_{23}}(x_c, x_d) = \sum_{x_a} \phi_{C_2}(x_a, x_c, x_d)$ and update ϕ_{C_3}:

$$\phi_{C_3}(x_c, x_d, x_e) \leftarrow \phi_{C_3}(x_c, x_d, x_e)\phi_{S_{23}}(x_c, x_d).$$

As a consequence we finally obtain the clique marginal representation

$$\begin{aligned} p(x_a, x_b, x_c, x_d, x_e) &= \frac{\phi_{C_1}(x_a, x_b, x_d)\phi_{C_2}(x_a, x_c, x_d)\phi_{C_3}(x_c, x_d, x_e)}{\phi_{S_{12}}(x_a, x_d)\phi_{S_{23}}(x_c, x_d)} \\ &= \frac{p(x_a, x_b, x_d)p(x_a, x_c, x_d)p(x_c, x_d, x_e)}{p(x_a, x_d)p(x_c, x_d)}, \end{aligned}$$

where all clique and separator potentials are identical to the marginal probability distribution over the variables involved. Hence, we can now find $p(x_a, x_d)$ by locating a clique containing x_a and x_d and sum out all other variables. If we choose C_2, we get

$$p(x_a, x_d) = \sum_{x_c} \phi_{C_2}(x_a, x_c, x_d).$$

Each time we multiply, divide or marginalize potentials, a number of binary operations (addition, multiplication and division) are conducted under the machinery. For a network with 41 variables and a maximum size of the state space for each variable being 3, Lepar and Shenoy (1998) recorded a total number of $2,371,178$ binary operations. We do not intend to follow the same analysis here. For sparse tables, however, the number of necessary binary operations is potentially much smaller.

2.1 Evidence and Slicing

Suppose it is known, before message passing, that $X_E = x_E^*$ for some labels $E \subset V$. We refer to x_E^* as evidence. Evidence can be entered into the clique potential representation (3) as follows. For each $v \in E$, choose an arbitrary clique, C, where $v \in C$, and set entries in ϕ_C that are inconsistent with x_v^* equal to zero. The resulting clique potential is then said to be sliced. After message passing, all queries are then conditional on $X_E = x_E^*$. Thus, entering evidence leads to more zero-cells, and in a sparse setup, the resulting clique potentials will be even more sparse. After message passing, the clique potential $\phi_C(x_C)$ is now equal to the conditional probability $p(x_{C \setminus E} \mid x_E^*)$.

It suffices to modify a single clique potential such that it is inconsistent with $v \in E$, for all v as described above. However, for sparse tables, it is advantageous to enter evidence in all clique potentials containing v since this leads to a higher degree of sparsity.

This is how evidence is handled in **jti**. In fact, this is one of the reasons why **jti** is able to handle very complex networks by exploiting the evidence using sparse tables from **sparta**. Whenever a clique, say C, receives evidence on some variables, the size of the sparse potential corresponding to C will be reduced. This will be illustrated in Section 4.1.

It is also possible to enter evidence into the factorization (2). This is the key to handle complex networks that are otherwise infeasible due to lack of memory as we show in Section 5.2. This is related to cutset conditioning, and if one can enter evidence into variables that breaks cycles, the effect can be huge (Pearl, 2013). One can exploit the law of total probability to sum out the variables conditioned on (since we obtain the probability of the evidence during propagation) and recover whatever probability is of interest.

3 An Intuitive way of Representing Sparse Tables

Before describing our method for multiplication and marginalization of sparse tables, it is illuminating to describe sparse tables in a standard R language setup. Consider two arrays `f` and `g`:

```
R> dn <- function(x) setNames(lapply(x, paste0, 1:2), toupper(x))
R> d  <- c(2, 2, 2)
R> f  <- array(c(5, 4, 0, 7, 0, 9, 0, 0), d, dn(c("x", "y", "z")))
R> g  <- array(c(7, 6, 0, 6, 0, 0, 9, 0), d, dn(c("y", "z", "w")))
```

with flat layouts

```
R> ftable(f, row.vars = "X")

   Y y1    y2
   Z z1 z2 z1 z2
X
x1    5  0  0  0
x2    4  9  7  0
```

```
R> ftable(g, row.vars = "W")

   Y y1    y2
   Z z1 z2 z1 z2
W
w1    7  0  6  6
w2    0  9  0  0
```

Converting f and g to data.frame objects and exclude the cases with a value of zero:

```
R> df <- as.data.frame.table(f, stringsAsFactors=FALSE)
R> df <- df[df$Freq != 0,]
R> dg <- as.data.frame.table(g, stringsAsFactors=FALSE)
R> dg <- dg[dg$Freq != 0,]
```

```
R> print(df, row.names = FALSE)

  X  Y  Z Freq
 x1 y1 z1    5
 x2 y1 z1    4
 x2 y2 z1    7
 x2 y1 z2    9
```

```
R> print(dg, row.names = FALSE)

  Y  Z  W Freq
 y1 z1 w1    7
 y2 z1 w1    6
 y2 z2 w1    6
 y1 z2 w2    9
```

This leaves us with two sparse tables, df and dg, respectively. To multiply df by dg, we must, by definition, determine the cases that match on the variables Y and Z that they have in common. For example, row 4 in df must be multiplied with row 4 in dg such that (y1, z2, x2, w2) is an element in the product with the value 81. And since the tables are sparse, no multiplication by zero will be performed. The multiplication can be performed with the following small piece of R code (which will be used in Section 6 in connection with the bench-marking):

```
R> sparse_prod <- function(df, dg){
+    S   <- setdiff(intersect(names(df), names(dg)), "Freq")
+    mrg <- merge(df, dg, by = S, suffixes = c("_df", "_dg"))
+    mrg <- within(mrg, val <- Freq_df * Freq_dg)
+    mrg[, setdiff(names(mrg), c("Freq_df", "Freq_dg"))]
+  }
```

The merge function performs, by default, what is also called an inner join or natural join in SQL terminology, which is exactly how we defined table multiplication in Section 1. Multiplying df and dg yields

```
R> sparse_prod(df, dg)
```

```
   Y  Z  X  W val
1 y1 z1 x1 w1  35
2 y1 z1 x2 w1  28
3 y1 z2 x2 w2  81
4 y2 z1 x2 w1  42
```

Marginalization is even more straightforward. Marginalizing out X from `df` can for example be done using the built-in R function `aggregate` (which is also used in Section 6 for benchmarking):

```
R> aggregate(Freq ~ Y + Z, data = df, FUN = sum)
```

```
   Y  Z Freq
1 y1 z1    9
2 y2 z1    7
3 y1 z2    9
```

Thus, we have the necessary tools to implement JTA using sparse tables. So why should we bother redefining sparse tables and algebras on these; because of execution time and memory storage. In Section 6, we show the effect of the effort of going beyond the `merge` and `aggregate` functions.

4 Sparse Tables

Let T be a dense table with domain $I = \times_{v \in V} I_v$. Define the level set $\mathcal{L} := \times_{v \in V} \mathcal{L}_v$ where $\mathcal{L}_v = \{1, 2, \dots, |I_v|\}$ and let $\# : I \to \mathcal{L}$ be a bijection. We define the sparse table $\tau = (\Phi, \phi)$, of T as the pair where Φ is a matrix with columns given by the set of vectors in the sparse domain $\mathcal{I} := \{\#(x) \mid T(x) \neq 0, x \in I\}$, consisting of non-zero cells and where ϕ is the corresponding vector of values. Thus, a column in Φ represents a cell in $\mathcal{I}$ and is written a tuple $i = (i_1, i_2, \dots, i_{|V|}; i_v \in \mathcal{L}_v)$ which explicitly determines the ordering of the labels and hence the order of the rows in Φ. The order of the columns in Φ is not important as long as it agrees with ϕ. We denote by $\Phi[j]$ the j'th column of Φ and by ϕ_j the corresponding j'th value in ϕ. The sub-matrix Φ_S defined over the set of labels, $S \subseteq V$, is the resulting matrix when rows corresponding to labels in $V \setminus S$ have been removed. Let T be the table `f` from Section 3:

```
   Y y1    y2
   Z z1 z2 z1 z2
X
x1    5  0  0  0
x2    4  9  7  0
```

The domain is given by

$$I = \{x_1, x_2\} \times \{y_1, y_2\} \times \{z_1, z_2\}$$

and we can choose # as the map $(x_{\ell_1}, y_{\ell_2}, z_{\ell_3}) \mapsto (\ell_1, \ell_2, \ell_3)$ for $\ell_1, \ell_2, \ell_3 \in \{1,2\}$. The non-zero cells can then be identified as $\mathcal{I} = \{(1,1,1),(2,1,1),(2,2,1),(2,1,2)\}$. Hence

$$\Phi = \begin{bmatrix} 1 & 2 & 2 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix},$$

with values $\phi = (5,4,7,9)$, corresponding to `df` in Section 3. Let G be another dense table with domain $J = \times_{u \in U} J_u$ and sparse representation $\gamma = (\Psi, \psi)$ with sparse domain $\mathcal{J}$. We then aim at defining the sparse multiplication $\tau \otimes \gamma$ of $T \otimes G$. Let $S = V \cap U$ be the separator labels shared between the two sparse tables τ and γ. Next, define the map $M_S(\Phi)$, which transform Φ_S into a look-up table[1] as follows: the keys are the unique columns of Φ_S and the value of $M_S(\Phi)$ at key k is the set of column indices where column k can be found in Φ_S and hence also in Φ and is given by

$$M_S(\Phi)[k] = \{j \in \{1,2,\ldots,|\mathcal{I}|\} : \Phi[j] = k\}.$$

Let $\mathcal{K}$ denote the mutual keys of $M_S(\Phi)$ and $M_S(\Psi)$. The number of columns in the matrix of the resulting product $\tau \otimes \gamma$ is then given as

$$N := \sum_{k \in \mathcal{K}} |M_S(\Phi)[k]| \cdot |M_S(\Psi)[k]|.$$

This observation is crucial, since the memory storage of the sparse product can then be computed in advance. If (Π, π) is the sparse product of τ and γ, we can therefore initialize Π as a matrix with $|V| + |U \setminus V|$ rows and N columns and π as an $N-$dimensional vector. Finally, π is given by the values $\phi_j \cdot \psi_{j'}$ for $j \in M_S(\Phi)[k]$ and $j' \in M_S(\Psi)[k]$ for all $k \in \mathcal{K}$. The procedure is formalized in Algorithm 3.

The number of binary operations is smaller than the equivalent dense table multiplication since every multiplication with zero is avoided. Moreover, since we only loop over the mutual keys, $\mathcal{K}$, the execution time will depend on the table having the least unique keys over the separator labels. Trivially, division of two sparse tables can be obtained by substituting line 11 of Algorithm 3 with $\pi_l = \phi_j / \phi'_j$.

[1] A lookup table is a list arranged as key-value pairs. In R one can think of a look-up table as a named list where the names are the keys and the values are the elements of the list.

Algorithm 3 Multiplication of Sparse Tables

```
procedure ($\tau = (\Phi, \phi)$: sparse table, $\gamma = (\Psi, \psi)$: sparse table)
    $S := V \cap U$
    $\mathcal{K}$: Mutual keys of $M_S(\Phi)$ and $M_S(\Psi)$
    $N := \sum_{k \in \mathcal{K}} |M_S(\Phi)[k]| \cdot |M_S(\Psi)[k]|$
    Initialize the matrix $\Pi$ with $|V| + |U \setminus V|$ rows and $N$ columns
    Initialize the vector $\pi$ of dimension $N$
    $l := 1$
    for $k \in \mathcal{K}$ do
        for $j \in M_S(\Phi)[k]$ and $j' \in M_S(\Psi)[k]$ do
            $\Pi[l] := (\Phi[j], \Psi_{U \setminus V}[j'])$
            $\pi_l := \phi_j \cdot \psi_{j'}$
            $l = l + 1$
        end for
    end for
    return $(\Pi, \pi)$
end procedure
```

Now, let G be the table $\mathfrak{g}$ from Section 3 where the domain is given by

$$J = \{w_1, w_2\} \times \{y_1, y_2\} \times \{z_1, z_2\}.$$

Choose the map $(w_{\ell_1}, y_{\ell_2}, z_{\ell_3}) \mapsto (\ell_1, \ell_2, \ell_3)$ for $\ell_1, \ell_2, \ell_3 \in \{1,2\}$. In summary, we have the tables

$$\Phi = \begin{bmatrix} 1 & 2 & 2 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}, \quad \Psi = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 1 & 2 \end{bmatrix},$$

along with $\phi = (5,4,7,9)$ and $\psi = (7,6,6,9)$. The separator labels are given by $S = \{y, z\}$ and the lookup tables of Φ and Ψ are then given by

$$\begin{aligned} M_S(\Phi) = {} & \{(1,1) := \{1,2\}, \quad (2,1) := \{3\}, \quad (1,2) := \{4\}\} \\ M_S(\Psi) = {} & \{(1,1) := \{1\}, \quad (2,1) := \{2\}, \quad (1,2) := \{4\}, \quad (2,2) := \{3\}\}. \end{aligned}$$

Above, y corresponds to row two, and z corresponds to row three in Φ. So, for example, $M_S(\Phi)[(1,1)] = \{1,2\}$ means that the key $(1,1)$ has the value $\{1,2\}$, which in turn means that $(1,1)$ is found in columns 1 and 2 in Φ. Therefore, all values ϕ_j for $j \in M_S(\Phi)[(1,1)]$ must be multiplied with all values ψ_j for

$j \in M_S(\Psi)[(1,1)]$, etc. Hence,

$$\Pi = \begin{bmatrix} 1 & 2 & 2 & 2 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \end{bmatrix},$$

and

$$\pi = (\phi_1 \cdot \psi_1, \phi_2 \cdot \psi_1, \phi_3 \cdot \psi_2, \phi_4 \cdot \psi_4) = (35, 28, 42, 81),$$

as expected from the result in Section 3 using `sparse_prod`. Notice, that we save any computation with ψ_3 since $(2,2)$ is not a key in $M_S(\Phi)$.

We mention that, addition and subtraction of sparse tables are more demanding since we have to reconstruct zero-cells if one of the tables has a non-zero cell-value while the other table has a zero-cell in the corresponding separator cell. Fortunately, these operations are not needed in JTA.

The marginal sparse table $\tau^{\downarrow A} = (\Phi^{\downarrow A}, \phi^{\downarrow A})$ of τ, corresponding to $T^{\downarrow A}$, can be calculated using the map $M_A(\Phi)$ and, for each key $k \in M_A(\Phi)$, sum the corresponding values in ϕ. However, for massive tables, the memory footprint of $M_A(\Phi)$ is unnecessarily large. Instead, we construct the lookup-table $H_A(\Phi)$ where the keys are the unique columns of Φ_A, as was the case in $M_A(\Phi)$. However, the values are themselves pairs where the first element is an index to any of the column indices where the corresponding key can be found in Φ_A. The second element is the final cell value in the marginalized table corresponding to the key. The pair corresponding to the key k is therefore on the form

$$H_A(\Phi)[k] = (j, v), \quad v = \sum_{\ell : \Phi_A[\ell] = k} \phi_\ell \text{ and } \Phi_A[j] = k.$$

The value v can easily be computed iteratively. The point here is, that we never have to store Φ_A since we can deduce all information from Φ on the fly given the row indices corresponding to A in Φ. The number of columns in the final matrix $\Phi^{\downarrow A}$, and hence the number of elements in $\phi^{\downarrow A}$, is given by $|H_A(\Phi)|$. The procedure is formalized in Algorithm 4. Consider again the sparse table τ of T and let $A = \{y, z\}$. Then, the resulting sparse marginal table have two rows corresponding to y and z. The construction of $H_A(\Phi)$ is as follows. The first column in Φ is extracted, and the entry corresponding to x is deleted. Call the resulting vector (key) k_1. Now, set $H_A(\Phi)[k_1] = (j = 1, v = 5)$ since $\phi_1 = 5$. Extract now, the second column of Φ and let k_2 be the resulting key when removing the entry corresponding to x. Since $k_1 = k_2$ and

Algorithm 4 Marginalization of Sparse Tables

```
procedure ($\tau = (\Phi, \phi)$: sparse table, $A$: Set of labels)
    Construct $H_A(\Phi)$
    $N = |H_A(\Phi)|$
    Initialize the matrix $\Phi^{\downarrow A}$ with $|A|$ rows and $N$ columns
    Initialize the vector $\phi^{\downarrow A}$ of dimension $N$
    Let $\mathcal{K}$ be the keys of $H_A(\Phi)$
    $l := 1$
    for $k \in \mathcal{K}$ do
        $(j, v) := H_A(\Phi)[k]$
        $\Phi^{\downarrow A}[l] := \Phi_A[j]$          ▷ deduced by picking elements from $\Phi[j]$
        $\phi_l^{\downarrow A} := v$
        $l = l + 1$
    end for
    return $(\Phi^{\downarrow A}, \phi^{\downarrow A})$
end procedure
```

$\phi_2 = 4$ we update $H_A(\Phi)[k_1] = (j = 2, v = 9)$. Proceeding this way gives

$$H_A(\Phi) = \{(1,1) := (j=2, v=9), (2,1) := (j=3, v=7), (1,2) := (j=4, v=9)\}\}.$$

Thus

$$\Phi^{\downarrow A} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix},$$

and $\psi^{\downarrow A} = (9, 7, 9)$. For $B \subset V$, the i_B^*- slice of a sparse table $\tau = (\Phi, \phi)$ is obtained by removing columns, k, in Φ for which k does not agree with i_B^*. We leave out the formal procedure for slicing.

Finally, we mention that Algorithm 3 is generic and applies in every situation. However, if the domain of one of the tables is a subset of the domain in the other table, multiplication can be performed much faster since we do not have to create new cells. This is always the case in JTA since the domain of the message is a subset of the domain in the potential receiving this message. We leave out the formal algorithm and just mention, that this algorithm also exploits the lookup table M whereafter it locates the cells to keep in the larger table without constructing new cells.

4.1 How to use sparta

To demonstrate the use of **sparta**, we revisit the example from Section 3 of the two (dense) tables f and g with mutual variables, Y and Z

```
R> ftable(f, row.vars = "X")      R> ftable(g, row.vars = "W")

   Y y1    y2                        Y y1    y2
   Z z1 z2 z1 z2                     Z z1 z2 z1 z2
X                                 W
x1    5  0  0  0                  w1    7  0  6  6
x2    4  9  7  0                  w2    0  9  0  0
```

We can convert these to their equivalent sparta versions as

```
R> library("sparta")
R> sf <- as_sparta(f); sg <- as_sparta(g)
```

Printing the object by the default printing method yields

```
R> print.default(sf)

  [,1] [,2] [,3] [,4]
X    1    2    2    2
Y    1    1    2    1
Z    1    1    1    2
attr(,"vals")
[1] 5 4 7 9
attr(,"dim_names")
attr(,"dim_names")$X
[1] "x1" "x2"

attr(,"dim_names")$Y
[1] "y1" "y2"

attr(,"dim_names")$Z
[1] "z1" "z2"

attr(,"class")
[1] "sparta" "matrix"
```

The columns are the cells in the sparse matrix and the vals attribute are the corresponding values which can be extracted with the vals function. Furthermore, the domain resides in the dim_names attribute, which can also be extracted using the dim_names function. From the output, we see that (x2, y2, z1) has a value of 2. Using the **sparta** print method prettifies things:

```
R> print(sf)

  X Y Z val
1 1 1 1   5
2 2 1 1   4
3 2 2 1   7
4 2 1 2   9
```

where row *i* corresponds to column *i* in the sparse matrix. We settled for this print method because printing column wise leads to unwanted formatting when the values are decimal numbers. Consider the cell (2,1,1). The corresponding named cell is then

```
R> get_cell_name(sf, sf[, 2L])

   X    Y    Z
"x2" "y1" "z1"
```

where `sf[, 2L]` is the second column (row in the output) of `sf`, which is (2,1,1). The product of `sf` and `sg` is

```
R> mfg <- mult(sf, sg); mfg

  X Y Z W val
1 2 1 2 2  81
2 2 2 1 1  42
3 1 1 1 1  35
4 2 1 1 1  28
```

The equivalent dense table has $2^4 = 16$ entries. However, `mfg` stores 20 values after all, 16 of which are information about the cells. That is, there is some overhead storing the information about the cells, see Section 4.2. Converting `sf` into a conditional probability table (CPT) with conditioning variable `Z`:

```
R> sf_cpt <- as_cpt(sf, y = "Z"); sf_cpt

  X Y Z   val
1 1 1 1 0.312
2 2 1 1 0.250
3 2 2 1 0.438
4 2 1 2 1.000
```

Slicing `sf` on `X = x1`

```
R> slice(sf, s = c(X = "x1"), drop = TRUE)

  Y Z val
1 1 1   5
```

reduces `sf` to a single non-zero element, whereas the equivalent dense case would result in a `(Y,Z)` table with one non-zero element and three zero-elements. This `slice` function is used in **jti** when the evidence `X = x1` is entered into the clique potential corresponding to `sf`. Marginalizing out `Y` in `sg` yields

```
R> marg(sg, y = c("Y"))
```

```
  Z W val
1 2 2   9
2 2 1   6
3 1 1  13
```

This is in correspondence with the example in Section 4. Finally, we mention that a sparse table can be created using the constructor `sparta_struct`, which can be necessary to use if the corresponding dense table is too large to have in memory.

4.2 When to use sparta

As shown in Section 4.1, there is an overhead of storing the information in a **sparta** object. A dense array with x elements takes up $8x$ bytes plus some negligible memory of storing the variable names etc. On the contrary, a **sparta** object with $y < x$ elements takes up $y(4k + 8)$ bytes, where k is the number of variables (these can be stored as integers and hence only requires 4 bytes each). In Figure 2, we have plotted this relation for $k = 4$, 6 and 8, and different levels of sparsity. That is, a sparsity of $1/2$ implies that $y = x/2$. The black identity line indicates the number of gigabytes needed to store the dense table with x elements. The size of the state spaces of the variables are implicitly reflected by the memory needed to store the dense table. The more memory needed, the larger state space of the variables. However, more variables and a larger state space of the variables will intuitively result in a more sparse table, making **sparta** efficient even for several variables.

The take away message from Figure 2 is that when the state space of the variables and the sparsity increases the benefit of storing the tables using **sparta** will outweigh the overhead of storing the additional information.

In connection to JTA, **sparta** is favorable when cliques with many variables imply a high degree of sparsity. In particular, this is often the case for tables in a Bayesian network representing a genetic pedigree. In this case, cliques tend to be small, but the state space of the variables can be arbitrarily large due to the large amount of DNA information for each member of the pedigree. In Section 5.1, we fit a decomposable MRF to real data and show that the sparsity of the clique potentials is much favorable towards **sparta**.

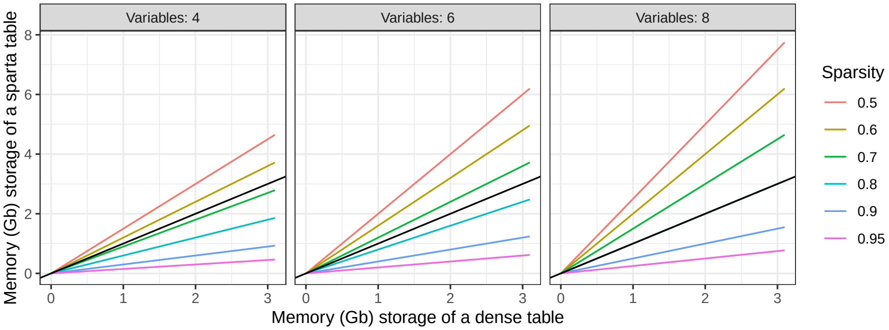


Fig. 2: The black identity line indicates the number of gigabytes needed to store the dense table with x elements. The colored lines indicate the number of gigabytes required to store the equivalent **sparta** object with the respective number of variables and sparsity.

4.3 Probability Trees and Value Based Potentials

This section is devoted to discussing differences between **sparta** and probability trees (PTs) (Boutilier et al., 2013) and value based potentials (VBPs) (Gómez-Olmedo et al., 2021). Firstly, PTs are potentials that are represented as trees where context specific information (CSI) (conditional independencies that only hold in specific contexts) can be exploited by collapsing nodes in the tree. This, however calls for methods to learn these CSIs, which can be computationally demanding. Moreover, PTs can be pruned, further resulting in an approximation of the potential by collapsing leaf nodes (in the same branch though) for which all values are close to the mean value of that branch. Finally, PTs can not disregard zero-cells in any way, which, by construction, is the power of **sparta**. We are not aware of any open source code that implements PTs.

Very recently, and almost parallel to this paper, VBPs were introduced with the goal of overcoming the limitations of PTs (Gómez-Olmedo et al., 2021). VBPs is an acronym for four new potentials that can be either value driven or index driven. We shall only focus on the new potentials called index driven with map (IDM) since these have the most resemblance with **sparta**, and also because they seem to perform best overall according to the benchmarks provided in Gómez-Olmedo et al. (2021).

To introduce IDMs, we first consider a dense table, `dt`, over the variables Z, Y, and X where all cells are represented by a vector of integers and assign to each cell a unique index:

```
R> dt <- cbind(
+    expand.grid(Z = 1:2, Y = 1:2, X = 1:2),
+    Freq = c(.4, .1, .7, .1, .6, 0, 0, .9),
+    idx = 1:2^3
+  )
R> print(dt, row.names = FALSE)
```

```
 Z Y X Freq idx
 1 1 1  0.4   1
 2 1 1  0.1   2
 1 2 1  0.7   3
 2 2 1  0.1   4
 1 1 2  0.6   5
 2 1 2  0.0   6
 1 2 2  0.0   7
 2 2 2  0.9   8
```

Since indices are unique, the table can be represented by the vector

```
R> structure(dt$Freq, names = dt$idx)
```

```
  1   2   3   4   5   6   7   8 
0.4 0.1 0.7 0.1 0.6 0.0 0.0 0.9 
```

where the names are the indices. The correspondence between indices and cells is as follows (Gómez-Olmedo et al., 2021): First assign to variable ℓ the weight $w_\ell = w_{\ell+1} \cdot |I_{\ell+1}| \cdot$ for $\ell < |V|$ and $w_{|V|} = 1$ where $|V|$ is the number of variables. The ordering is given from first to last, i.e. Z is the first, Y is the second, and X is the third variable in the case of `dt`.

The index of a given cell, i, is then given by

$$\text{idx}(i) = \sum_{\ell=1}^{|V|} (i_\ell - 1) \cdot w_\ell. \tag{6}$$

Thus, the index of cell (z_1, y_2, x_2) is $(1-1) \cdot 4 + (2-1) \cdot 2 + (2-1) \cdot 1 = 4$ as expected. Given an index, ℓ, the k'th component of the corresponding cell is given by

$$\lfloor k/w_l \rfloor \text{ modulo } |I_\ell|, \tag{7}$$

where I_ℓ is the statespace of the ℓ'th variable. Thus, it is a fairly cheap operation to convert between the index and the cell, which is the backbone when manipulating IDMs. The IDM, IDM$_\phi$, of the potential ϕ is a representation of ϕ consisting of a lookup table D and an array A. A holds all unique values of ϕ, excluding zero. The keys in D are the indices in ϕ excluding indices corresponding to zero-cells and the values are indices from A corresponding to the cell value. We can now form the IDM of dt:

```
R> dt_no_zeroes <- subset(dt, Freq != 0)
R> unique_values <- unique(dt_no_zeroes$Freq)
R> A <- structure(unique_values, names = seq_along(unique_values))
R> D <- structure(sapply(dt_no_zeroes$Freq, function(k) {
+    match(k, A)
+  }), names = dt_no_zeroes$idx)
R> (IDM <- list(A = A, D = D))
```

```
$A
  1   2   3   4   5 
0.4 0.1 0.7 0.6 0.9 

$D
1 2 3 4 5 8 
1 2 3 2 4 5 
```

First, notice that A and D are not true lookup tables with constant lookup, but ordinary R vectors. At first, it may seem redundant to form the A instead of just forming the structure

```
R> (B <- structure(dt_no_zeroes$Freq, names = dt_no_zeroes$idx))
```

```
  1   2   3   4   5   8 
0.4 0.1 0.7 0.1 0.6 0.9 
```

This is because that for IDMs, repeated values, like 0.1, take up a single float in the memory as opposed to B that must store a float for each repetition of the same value. For tables with many repeated values, this can potentially save a lot of memory. The question is now whether or not IDMs can be multiplied and marginalized within reasonable time. There are no benchmarks of multiplication and marginalization in Gómez-Olmedo et al. (2021). Still, we can with reasonable confidence state that multiplication of IDMs becomes increasingly slower than **sparta** as the state space of the product increases. We state this for two reasons. For one, we did actually consider a variant of IDMs for which multiplication turned out to be extremely slow for large tables. Second, when IDMs are multiplied one must loop over the entire dense state space of the resulting table. In more detail, suppose we want to find the product $\mathrm{IDM}_{\phi_Z} = \mathrm{IDM}_{\phi_X} \otimes \mathrm{IDM}_{\phi_Y}$. Let z_l be the ℓ'th cell in IDM_{ϕ_Z}. Then for all $\ell = 1,2,\ldots,|I_Z|$, one must use (7) to construct the cell z_l, project this cell onto A, use (6) to convert to and index in IDM_{ϕ_A} and lookup the value and test if this is zero. If not, also project z_ℓ onto B, multiply the values and append the scalar product and ℓ to IDM_{ϕ_C}. Imagine $|I_C|$ being huge, then multiplication of IDMs is cumbersome.

Based on this, we do not intend to benchmark IDMs against **sparta**, but we summarize the discussion by saying that neither IDMs nor **sparta** is, generally, better than the other. It depends on the problem at hand. IDMs are most often more memory efficient than **sparta** but for larges tables with high degrees of sparsity, **sparta** is faster. For very large networks, it may even make sense to encode a Bayesian network to hold both IDMs and **sparta**s such that the IDM potentials ensure that it is even possible to compile the network. This calls for methods to combine IDMs and **sparta** tables which we leave for future research.

5 Usecases of jti and sparta

In **jti**, there are two ways of specifying a Bayesian network. Either by a list of CPTs or a dataset together with a DAG. In the latter case, the CPTs are found using maximum likelihood estimates. Here, we describe how to use **jti** with the classic Bayesian network `asia` (Lauritzen and Spiegelhalter, 1988) where the corresponding CPTs is part of **jti**. The network represents a simplified model to help diagnose patients arriving at a respiratory clinic. A history of smoking has a direct influence on both whether or not a patient has bronchitis and whether or not a patient has lung cancer. Both lung cancer and bronchitis can result in dyspnoea. An x-ray result depends on the presence of either tuberculosis or lung cancer. Finally, a visit to Asia influences the probability of having tuberculosis. The DAG is depicted in Figure 3.

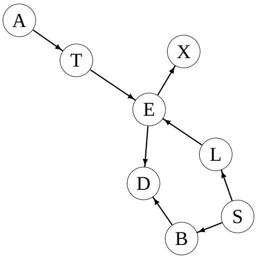


Fig. 3: The DAG for the `asia` network with variables `asia` (A), `tuberculosis` (T), `either` (E), `x-ray` (X), `lung` (L), `dysp` (D), `smoke` (S) and `bronc` (B).

We use the version of `asia` called `asia2`, both shipped with **jti**, which is a list of CPTs and which is shipped with **jti**. The first step is to call `cpt_list` for some initial checks and conversion to **sparta** tables:

```
R> cl <- cpt_list(asia2)
R> cl
```

```
 List of CPTs
 -------------------------
  P( asia )
  P( tub | asia )
  P( smoke )
  P( lung | smoke )
  P( bronc | smoke )
  P( either | lung, tub )
  P( xray | either )
  P( dysp | bronc, either )

  <cpt_list, list>
 -------------------------
```

From the output, we see the inferred CPTs corresponds to Figure 3, giving rise to a factorization in the same way as in (2). The network is now ready for compilation which involves moralization and triangulation. In **jti** there are four different choices for triangulation which are all based on the elimination game algorithm, see Flores and Gámez (2007). One of the most well-known heuristics is `min_fill` which tries to minimize the number of fill edges. Evidence can be entered either at compile stage or just before message passing begins. It is always advisable to enter evidence at compile stage since we know from Section 2.1 that this reduces the number of non-zero elements in the CPTs and hence the memory footprint and execution time. A good strategy might be to locate one or more of the largest cliques and enter evidence on the nodes contained in these. We can investigate the cliques and their state spaces prior to compilation by triangulating the graph as follows:

```
R> tri <- triangulate(cl, tri = "min_fill")
```

The `tri` object is a list containing the triangulated graph, `new_graph` as a matrix, a list of `fill_edges`, the `cliques`, and the size of the dense `statespace` of each clique. In Section 5.2, we show how to exploit information about the cliques with largest state space in order to compile and propagate in a network that is otherwise infeasible on a 'standard' laptop. Now, let for example `tub = yes` be the evidence indicating that a given person has tuberculosis. The compiled network is then constructed as

```
R> cp <- compile(cl, evidence = c(tub = "yes"), tri = "min_fill")
R> cp
```

```
Compiled network
-------------------------
 Nodes: 8
 Cliques: 6
  - max: 3
  - min: 2
  - avg: 2.67
 Evidence:
  - tub: yes
 <charge, list>
-------------------------
```

The cliques can be extracted from the compiled object with `get_cliques(cp)`. The compiled object can now be entered into the message passing procedure as:

```
R> j <- jt(cp)
```

The junction tree can be visualized by plotting the object as `plot(j)`, see Figure 4. Finally, we can calculate the probability of a given person having a positive x-ray result, `xray = yes`, given that the person has tuberculosis as

```
R> query_belief(j, nodes = "xray")
```

```
$xray
xray
 yes   no
0.98 0.02
```

Thus, given that a person has tuberculosis, the probability of observing a positive x-ray result is 0.98. The probability of observing positive x-ray result given that `tub = "no"` can be calculated accordingly and equals 0.1012. Joint queries can be calculated by specifying `type = "joint"` in `query_belief`.

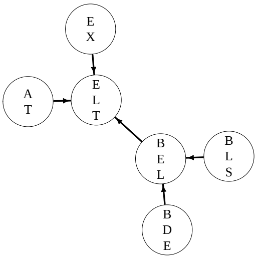


Fig. 4: A junction tree for the asia network.

5.1 Inference in Decomposable Markov Random Fields

In this section, we illustrate the gain of using **sparta** for the analysis of the public-domain `derma` data set (part of the **ess** package) originally obtained from the UCI Machine Learning Repository (Dua and Graff, 2017). The data set consists of 358 observations, 35 variables of which there is one class variable called `ES`. The class variable has six states, seboreic dermatitis, psoriasis, lichen planus, chronic dermatitis, pityriasis rosea and pityriasis rubra pilaris, each representing a skin disease. The remaining 34 clinical variables (all with 4 states except `age`, which has been discretized into six bins) are used to predict the type of skin disease. We first fit a decomposable MRF

```
R> library("ess")
R> g <- fit_graph(derma, q = 1.5, sparse_qic = TRUE)
```

The fitted graph is plotted in Figure 5, where we just notice, that it is a rather complex graph with large cliques. Hence, for such a small data set and a graph with big cliques, the chance of sparse clique potentials is significant. Extracting the sparse tables clique potentials using `pot_list` and compile:

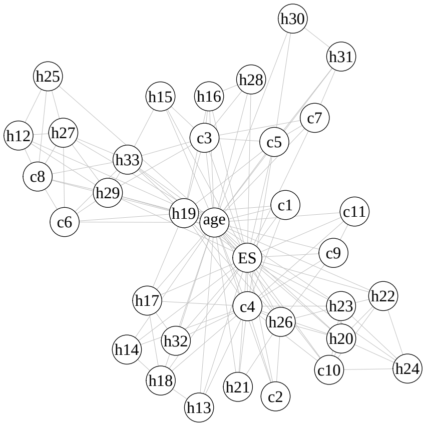


Fig. 5: The fitted decomposable MRF for the derma data set.

```
R> cl <- pot_list(derma,  as_igraph(g))
R> j  <- jt(compile(cl, root_node = "ES"), propagate = "no")
```

The argument `root_node = "ES"` forces `ES` into the root clique. Hence, if we are interested in queries about `ES`, we only need to conduct the inwards message passing, collect, since the root clique potential is identical to the probability distribution over the variables in the root clique. The argument `propagate = "no"` means that message passing should be postponed. By doing so, the initialization is only performed once, and the junction tree can be exploited as many times as needed with different evidence.

The largest clique of `j` has 6 variables, and the mean sparsity of the clique potentials is given by

```
R> mean(sapply(j$charge$C, sparta::sparsity))
```

```
[1] 0.9233838
```

showing that the tables are extremely sparse. The junction tree can now be used for e.g., classification. Consider the first observation in `derma`

```
R> z <- unlist(derma[1, -ncol(derma)])
```

and update the junction tree with evidence corresponding to `z` and perform inwards message passing

```
R> jz <- propagate(set_evidence(j, z), prop = "collect")
```

The probability distribution of the class variable given the evidence `z` is then

```
R> q <- query_belief(jz, nodes = "ES", type = "marginal")[[1]]
R> which.max(q)
```

```
seboreic dermatitis
                  1
```

Hence, we classify `z` as an observation from class seboreic dermatitis.

5.2 The Impact of Evidence

The Bayesian Network `Link` (Jensen and Kong, 1999) is a large Bayesian network with 724 nodes and 1,125 arcs. The network has been used for linkage analysis where, previously, only approximate methods have been applied to make inference in the network. We have verified that **sparta** generates a large amount of overhead in the CPTs for the `Link` network. In fact, **jti** uses approximately 16 times more memory to store the clique potentials than the dense version. However, in the following, we show how **sparta** leverages from entering evidence and hence reduces the CPTs.

We tested three of the most comprehensive software packages, across different programming languages for belief propagation and tested if they were able to handle `Link` on a "standard" laptop machine. We used the R package **gRain**, the R package **BayesNetBP**, the Python package **pyAgrum** (Gonzales et al., 2017), the Julia package **BayesNet** (Stanford, 2020). Interestingly, all of these failed due to lack of computer memory on the first author's laptop machine[2] while **jti** succeeded. The **pyAgrum** package is a high-level inter-

[2] See Section 7 for details about this machine.

face to the extremely efficient C++ library **aGrum** (Gonzales et al., 2017). In Gonzales et al. (2017), the authors of **pyAgrum** themselves failed to propagate in Link on a computer with 32Gb memory. An investigation revealed that exploiting the triangulation found by **jti**, **gRain** is now also able to make inference in Link. We have not tried this with the other packages mentioned but conjecture that the other packages would also be able to handle the Link network. Interestingly, this triangulation was the result of one of the most well-known heuristics, namely min_fill. The reason that triangulation has such a big impact is, that it determines the resulting cliques and hence the size of the clique potentials. If the cliques are too large, and hence the number of elements in the state space is large, it may not be possible to initialize the clique potentials due to lack of memory.

We now compare the cliques with large state spaces resulting from the two heuristics min_fill and min_nei (a variant of min_fill which also tries to minimize the number of fill-in edges). We have extracted the Link network from https://www.bnlearn.com/bnrepository/ as a list of CPTs with information about child and parent-node relations. In **jti**, we just need a list of CPTs, which we extract as

```
R> cpts <- bnfit_to_cpts(Link)
```

Next, we convert these CPTs to their **sparta** equivalent using cpt_list while deducing the underlying network, conducting some sanity checks and triangulate with the two heuristics

```
R> cl <- cpt_list(cpts)
R> tri_min_fill <- triangulate(cl, tri = "min_fill")
R> tri_min_nei  <- triangulate(cl, tri = "min_nei")
```

The size of the five largest clique state spaces can then be extracted as

```
R> sp_min_fill <- sort(tri_min_fill$statespace, decreasing = TRUE)[1:5]
R> sp_min_nei  <- sort(tri_min_nei$statespace, decreasing = TRUE)[1:5]
```

and the memory usage, for dense tables, in gigabytes of allocating memory for each of these are

```
R> round(sum(sp_min_fill) / 1e9 * 8, 2)
```

```
[1] 0.2
```

```
R> round(sum(sp_min_nei) / 1e9 * 8, 2)
```

```
[1] 26.98
```

We imagine that there is no triangulation of Link such that any software packages, including **jti**, are able to compile and propagate in Link on a standard laptop. In what follows we show that we can overcome the initialization step with **jti**, by exploiting evidence and sparsity. We use the `min_nei` heuristic since this was the poorest method, locate one of the cliques with largest state space, and insert evidence on 10 variables.

```
R> max_idx  <- which.max(tri_min_nei$statespace)
R> max_vars <- tri_min_nei$cliques[[max_idx]]
R> max_dim  <- dim_names(cl)[max_vars[1:10]]
R> e        <- sapply(max_dim, '[[', 1L)
R> cp       <- compile(cl, e, tri = "min_nei")
R> j        <- jt(cp)
```

The memory size (Gb), in the dense sense, in the resulting five largest clique potentials is

```
R> round(sort(sum(sapply(j$charge$C, sparta::table_size) / 1e9 * 8),
+  TRUE), 2)
```

```
[1] 0.06
```

which is negligible compared to 26.98 Gbs. The sparse tables take up 0.02 Gbs.

6 Time and Memory Trade off in Sparta

We investigate the trade off between memory allocation and execution time for multiplication and marginalization on **sparta** objects. If **sparta** objects do not perform reasonably well for small and medium sized tables, their practical usage is limited in real usecases.

Thus, we compare three functions for multiplication and three functions for marginalization: 1) The `mult` and `marg` functions from **sparta**, 2) the `tabMult` and `tabMarg` functions from **gRbase** and 3) the `sparse_prod` and `aggregate` functions from Section 3. In the latter case, we just refer to R as the package.

We randomly generate pairs of tables such that the number of cells in the product of the two tables does not exceed 10^6. We varied the sparsity of the product of the tables; 0% sparsity (dense tables), $(1-75]$% sparsity, and (75,99]% sparsity. For each pair of tables, we multiply them together and record the memory usage (in megabytes) of the product and the execution time (in seconds). As **gRbase** is the standard and most mature package for graphical models in R, the performance comparisons are relative to that of **gRbase**. Hence, in Figure 6, **gRbase** performs better for tables with the relative scores above one (indicated by horizontal dashed lines), whereas values below one show cases where the alternative approaches are better. The comparisons are plotted for different ranges of table sizes (panels) and sparsity of the resulting table (first axis).

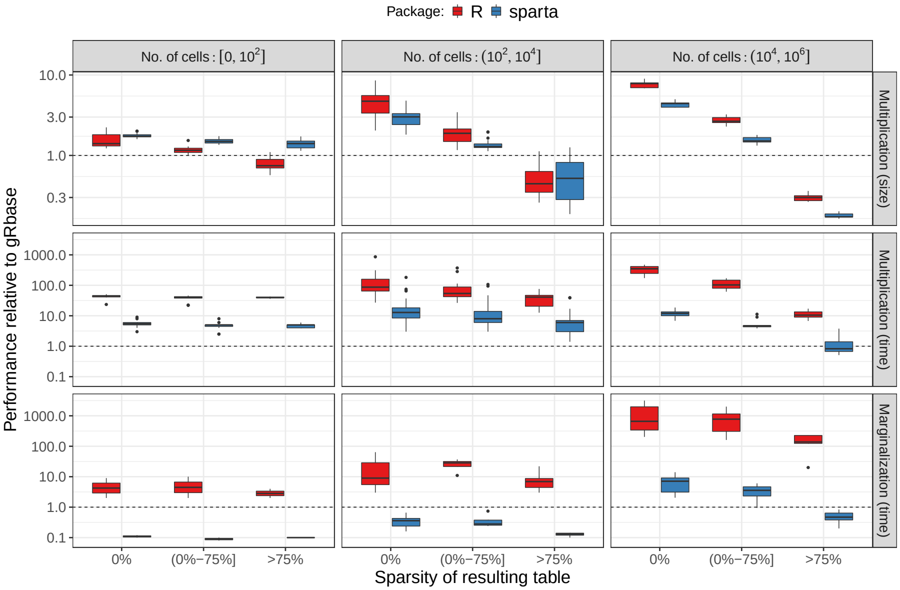


Fig. 6: Comparison of R (`sparse_prod` and `aggregate`) and **sparta** (`mult` and `marg`) to **gRbase** (`tabMult` and `tabMarg`) in terms of memory usage and timing. The top row shows the relative memory usage for multiplication, whereas the two lower rows show the relative timing for multiplication and marginalizaion, respectively.

Multiplication (size): The first row of Figure 6 describes the size of the table resulting from multiplying two tables. It can be seen that R consistently produces tables of smaller sizes than **sparta** for very small tables with 100 (10^2) cells. For tables with more than 100 cells, **sparta** consistently produces smaller tables except for a single case. Increasing the degree of sparsity leads to reduced object sizes for both R and **sparta**, and for tables with more than 75% sparsity, **sparta** outperforms both R and **gRbase** except in the first two panels with small tables.

Multiplication (time): The second row of Figure 6 describes the computing time for multiplying two tables. Clearly, **sparta** outperforms R by orders of magnitude. For larger tables (the two rightmost columns), there is also a clear effect of the degree of sparsity on the computing time.

Marginalization (time): The third row of Figure 6 describes the computing time for marginalizing out all variables in a table. When the degree of sparsity increases, the computing time decreases. In the comparison between **gRbase** and **sparta**, we see that the marginalization implementation of **sparta** is competitive to that of **gRbase**. For tables with 10,000 (10^4) or fewer cells, it is faster irrespective of the sparsity except in a single case. For 75% sparsity, **sparta**'s marginalization is consistently faster. Efficient marginalization is not only relevant for propagation, but also for querying propababilites in a junction tree that has been fully propagated.

7 Summary

We have presented a novel method for the multiplication and marginalization of sparse tables. The method is implemented in the R package **sparta**. However, the method is generic, and we have provided detailed pseudo algorithms facilitating the extension to other languages. In addition, we have presented the companion package **jti** to illustrate some of the advantages of **sparta** in connection with the junction tree algorithm. We hope to explore the benefit of the C API for working with external pointers to reduce the memory usage for **sparta** objects in the future. We also described IDM potentials which are very memory efficient but less time efficient. We intend to explore how IDMs and **sparta** can be combined in the future.

The memory footprint of the clique potentials can become prohibitively large when the sizes of the cliques are large This may not be true in general for sparse tables. As a matter of fact, it may be optimal to have large cliques if they are very sparse and/or if it is common to observe evidence variables in such cliques.

The benchmark study indicates that our proposed method for table multiplication and marginalization performs well for small, medium, and large tables. However, our real interest is in the performance on massive tables, which is impossible to benchmark in this paper due to the increased running time and memory usage of **gRbase** and R. For pedigree networks e.g., the CPTs can be huge. Thus, it is possible to construct the sparse tables without representing the dense arrays. Finally, we mention that the benchmark did not consider tables where the domain of one of the tables is contained in the domain in the second table. As discussed in Section 4, these are the typical cases in JTA and the performance of `mult`, in these cases, is considerably faster.

Although JTA, and hence **jti**, can not handle non-decomposable MRFs, it would be possible to utilize **sparta** in connection to loopy belief propagation for this task.

Acknowledgment

We are thankful for the constructive comments from the anonymous reviewers. The paper has been greatly improved by these.

Computational Details

We used versions 0.8.3 of **jti** and version 0.8.3 of **sparta** in this paper. Detailed examples, source code, and information about installation can be found at

`https://github.com/mlindsk/jti`

and

`https://github.com/mlindsk/sparta`

jti have a GNU general public license whereas **sparta** has a MTI license.

In addition to the packages already mentioned, the following R packages were used for the benchmark results:

- `dplyr` version 1.0.7.
- `glue` version 1.4.2.
- `tictoc` version 1.0.1.
- `ggplot2` version 3.3.5.

We used R version 4.1.0. All computations were carried out on a 64-bit Linux computer with Ubuntu 20.04.2 and Intel(R) Core(TM) i7-6600U CPU 2.60GHz LTS. The machine has approximately 6Gb of free memory for use in calculations.

References

Andrade, M. and Ferreira, M. A. M. (2009). Bayesian networks in forensic identification problems.

Aragam, B., Gu, J., and Zhou, Q. (2019). Learning large-scale bayesian networks with the sparsebn package. *Journal of Statistical Software*, 91(11):1–38.

Boettcher, S. G. and Dethlefsen, C. (2003). **deal**: A package for learning bayesian networks. *Journal of Statistical Software*, 8(20):1–40.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. (2013). Context-specific independence in bayesian networks. *arXiv preprint arXiv:1302.3562*.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Eddelbuettel, D. and Sanderson, C. (2014). **RcppArmadillo**: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063.

Flores, M. J. and Gámez, J. A. (2007). A review on distinct methods and approaches to perform triangulation for bayesian networks. In *Advances in Probabilistic Graphical Models*, pages 127–152. Springer-Verlag.

Gómez-Olmedo, M., Cabañas, R., Cano, A., Moral, S., and Retamero, O. P. (2021). Value-based potentials: Exploiting quantitative information regularity patterns in probabilistic graphical models. *International Journal of Intelligent Systems*.

Gonzales, C., Torti, L., and Wuillemin, P.-H. (2017). **aGrUM**: a graphical universal model framework. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 171–177. Springer-Verlag.

Højsgaard, S., Edwards, D., and Lauritzen, S. (2012). *Graphical models with R*. Springer Science & Business Media.

Hornik, K., Meyer, D., and Buchta, C. (2020). ***slam**: Sparse Lightweight Arrays and Matrices*. R package version 0.1-48.

Højsgaard, S. (2012). Graphical independence networks with the **gRain** package for R. *Journal of Statistical Software, Articles*, 46(10):1–26.

Jensen, C. S. and Kong, A. (1999). Blocking gibbs sampling for linkage analysis in large pedigrees with many loops. *The American Journal of Human Genetics*, 65(3):885 – 901.

Konis, K. (2014). ***RHugin**: R Package Version 7.8.*

Kumar, P., Ranganath, S., and Weimin, H. (2003). Bayesian network based computer vision algorithm for traffic monitoring using video. In *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, volume 1, pages 897–902. IEEE.

Lauritzen, S. L. (1996). *Graphical models*, volume 17 of *Oxford Statistical Science Series*. Clarendon Press.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194.

Lepar, V. and Shenoy, P. P. (1998). A comparison of lauritzen-spiegelhalter, hugin, and shenoy-shafer architectures for computing marginals of probability distributions. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 328–337. Morgan Kaufmann Publishers Inc.

Lindskou, M. (2021a). ***ess**: Efficient Stewise Selection in Decomposable Markov Random Fields*. R package version 1.1.2.

Lindskou, M. (2021b). ***jti**: Junction Tree Inference*. R package version 0.8.0.

Lindskou, M. (2021c). ***sparta**: Tables*. R package version 0.8.1.

Maathuis, M., Drton, M., Lauritzen, S., and Wainwright, M. (2018). *Handbook of graphical models*. CRC Press.

Masante, D. (2020). ***bnspatial**: Spatial Implementation of Bayesian Networks and Mapping*. R package version 1.1.1.

Mihaljeviċ et al. (2018). ***bnclassify**: Learning Bayesian Network Classifiers*.

Moharil, J. (2016). *Belief Propagation in Genotype-Phenotype Networks*. R package version 2.0.1.

Pearl, J. (2013). A constraint propagation approach to probabilistic reasoning. *arXiv preprint arXiv:1304.3422*.

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Stanford, I. S. L. (2020). *A Julia package for Bayesian Networks*. Julia package version v.3.3.2.

Weber, P., Medina-Oliva, G., Simon, C., and Iung, B. (2012). Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*, 25(4):671–682.

Yu, H., Moharil, J., and Blair, R. (2020). **BayesNetBP**: An R package for probabilistic reasoning in bayesian networks. *Journal of Statistical Software, Articles*, 94(3):1–31.

Zagorecki, A., Orzechowski, P., and Hołownia, K. (2013). A system for automated general medical diagnosis using bayesian networks. In *MEDINFO 2013*, pages 461–465. IOS Press.

Zamora, R. (2019). ***tensorr**: Sparse Tensors in R*. R package version 0.1.1.

Paper D

Unity Smoothing for Handling Inconsistent Evidence in Bayesian Networks and Unity Propagation for Faster Inference

Mads Lindskou, Torben Tvedebrink, Poul Svante Eriksen, Søren Højsgaard and Niels Morling

The paper has been submitted to
International Journal of Approximate Reasoning

The layout has been revised.

Abstract

We propose Unity Smoothing (US) for handling inconsistencies between a Bayesian network model and new unseen observations. We show that prediction accuracy, using the junction tree algorithm with US is comparable to that of Laplace smoothing. Moreover, in applications were sparsity of the data structures is utilized, US outperforms Laplace smoothing in terms of memory usage. Furthermore, we detail how to avoid redundant calculations that must otherwise be performed during the message passing scheme in the junction tree algorithm which we refer to as Unity Propagation (UP). Experimental results shows that it is always faster to exploit UP on top of the Lauritzen-Spigelhalter message passing scheme for the junction tree algorithm.

1 Introduction

Bayesian networks (BNs) (Pearl, 2014, Cowell et al., 2007) are statistical models that encode complex joint probability distributions using *directed acyclic graphs* (DAGs), from which conditional independencies among the variables can be inferred. Generally, the variables can be of any type, but we assume that the variables are discrete. In some applications, it is intractable to calculate probabilities from the joint distribution. BNs alleviate this by factorizing the joint distribution into lower-dimensional *conditional probability tables* (CPTs), which can be identified from the DAG. BNs can be fully data-driven, entirely expert specified, or a combination of these, which is a very elegant feature not shared by many statistical methods. By knowing the DAG, one can utilize both exact and approximate methods to answer inference questions, i.e., calculate posterior probabilities. In more general terms, BNs are used to support decision-making under uncertainty.

As for most other statistical methods, BNs can be used to make predictions for one or more of the variables given evidence on a subset of the other variables. In cases where the evidence specifies a previously unobserved configuration of the variables, the BN would, in many cases, lead to a degenerate model and assignment of zero probability to this evidence configuration. To circumvent this, Laplace smoothing may be applied to ensure non-zero probabilities for all possible configurations by adding a pseudo-count $\alpha > 0$ to each cell of the contingency table. However, in effect this also discards all *structural zeroes*, which are domain-induced constraints representing impossible outcomes. Structural zeroes are frequent in expert-specified BNs. However, they may also occur in data-driven BNs. We propose *Unity Smoothing* (US) as an alternative to Laplace smoothing, which has several attractive properties: (1) if the evidence does not violate the structural zeroes, these are preserved, (2) prediction accuracy in BNs using US is comparable to that of Laplace smoothing (see Section 6), (3) sparse CPTs (Lindskou et al., 2021) are equally sparse after smoothing, and (4) the smoothing takes place only in case of inconsistent evidence and only for the zero-probability CPTs involved (just-in-time smoothing).

For BNs to succeed in real applications, it is crucial that run-time performance is at an acceptable level. Especially for online problems, where posteriors must be computed continuously, even small improvements matter. Memory may be the biggest concern in other situations, where the multiplication of CPTs may become intractable.

Exact methods for inference in BNs include variable elimination (VE) (**?**) and the junction tree algorithm (JTA). In VE, one specifies a *query* (evaluation of a posterior density) in advance and sum out the relevant variables of the factorization to reach the query. JTA is much more involved and consists of many sub-routines. Here, a second structure called the *junction tree* is constructed, in which messages are passed between nodes. After a full round of message passing, one can query posterior probabilities on all variables given the evidence, that refers to a set of variables that are instantiated to have a specific value. There exists different architectures for sending messages where the most well-known ones are Lauritzen-Spigelhalter (LS) (Lauritzen and Spiegelhalter, 1988), HUGIN (Jensen et al., 1990), Shafer-Shenoy (SS) (Shafer and Shenoy, 1990), and Lazy Propagation (LP) (Madsen and Jensen, 1999). The LS and HUGIN architectures are very similar, though with small differences that have their own advantages. SS differs substantially from LS and HUGIN in that it keeps a factorization of the CPTs, whereas the former two does not. While LS, HUGIN, and SS do not use the independency-information from the DAG, LP exploits the factorization of SS and the DAG to further reduce the complexity of the message passing by removing irrelevant CPTs. This introduces some overhead and (Butz et al., 2018) introduced *Simple Propagation* (SP) as a lighter and faster version of LP.

In the need for speed and quest of reducing the required memory during JTA, we propose a new architecture, which we call *Unity Propagation* (UP).

This architecture can be combined with any of the aforementioned schemes. In this paper, we use LS as back-end to showcase UP. In essence, UP avoids multiplication with the trivial unity tables.

The paper is organized as follows: Section 2 reviews BNs and introduces necessary notation to introduce UP. The novel just-in-time method US, for inconsistent evidence is introduced in Section 2.3.

Sections 4 and 5 details how UP modifies the typical steps of the Junction Tree Algorithm and how some specific computations can be avoided to speed up the propagation. The effect of Laplace Smoothing versus US is compared in context of classification prediction accuracy for a number of standard machine learning datasets. The numerical experiments in Section 6 also include an investigation of the computational time-gain from using UP for a larger number of BNs encompassing both data-driven and expert-specified structures.

2 Preliminaries

2.1 Bayesian Networks

Let V be a set of discrete random variables with finite statespaces and let G be a directed acyclic graph (DAG), where the set of nodes is given by V. The joint probability mass function (pmf) over V is then given by

$$p(V) = \prod_{X \in V} p(X \mid \pi_X), \tag{1}$$

where $p(X \mid \pi_X)$ is the probability of X given π_X, and π_X is the set of parent nodes of X, i.e., the nodes for which a directed edge points towards X. Usually, the inference task of interest is to calculate posterior marginals of p. For prediction and classification problems, interest is most often to compute posteriors of the form $p(X \mid U)$, where U is the evidence. By evidence, we mean variables that have been instantiated to take on a specific value. Let $E \subset V$ be a set of *evidence variables*, the evidence is then on the form $U = \{X = x \mid X \in E\}$, where x is an instantiation of the random variable X.

Because of the discrete nature of the conditionals in (1), $p(X \mid \pi_X)$ can be represented as a CPT, i.e., a table where each element is a conditional probability. A BN consists of CPTs together with a DAG.

2.2 Potentials

Sometimes, we use the shorter notation, $p_{X|\pi_X}$, for the corresponding CPT of $p(X \mid \pi_X)$. However, we also denote it as $\phi_A = p_{X|\pi_X}$, $A = \{X, \pi_X\}$, when the specific relation between X and π_X does not matter. We say that ϕ_A is a *potential*, and the subscript notation explicitly denotes that ϕ_A is a potential defined over the variables in the *domain* $A \subseteq V$. In general, a potential is a real-valued and non-negative function. A CPT is always a potential, whereas a potential is not necessarily a CPT.

Let $a \in A$ be a variable with k possible outcomes, then $I_a = \{a_1, a_2, \ldots, a_k\}$ denotes the *levelset* of variable a. The levelset of a set, A, is defined as the product $I_A = \times_{a \in A} I_a$ with $|I_A| = \prod_{a \in A} |I_a|$ elements. The elements in I_A are called the *cells* of the potential ϕ_A, and the value of ϕ_A at cell $i_A \in I_A$ is denoted $\phi_A(i_A)$. The sum of all cell values of ϕ_A is denoted as $|\phi_A|$. The product $\phi_A \otimes \phi_B$ of two potentials with domain A and B is defined cell-wise as

$$(\phi_A \otimes \phi_B)(i_{A\cup B}) := \phi_A(i_A)\phi_B(i_B) \tag{2}$$

for $i_A \in I_A, i_B \in I_B$ and $i_{A\cup B} \in I_{A\cup B}$. Division is defined similarly, where $0/0 := 0$. We also use the notation

$$\phi_A^{\downarrow B}(i_B) = \sum_{I_{A\setminus B}} \phi_A(i_B, i_{A\setminus B}), \quad B \subseteq A \tag{3}$$

to denote the i_B-th cell-value of the *projection* of ϕ_A onto to the set B.

Two potentials are of special interest in the following: A *null potential*, $\mathbf{0}_g A$, is a potential in which all cell values are zero. Such cells are also termed *zero-cells*. A *unity potential*, $\mathbf{1}_A$, is a potential in which all cell values are one. The product of a null potential with any potential is a null potential but possible with a larger domain, that is $\phi_A \otimes \mathbf{0}_B = \mathbf{0}_{A\cup B}$. Likewise, the unity potential $\mathbf{1}_B$, has the property that $\phi_A \otimes \mathbf{1}_B = \phi_A$ for any potential ϕ_A with domain A if $B \subseteq A$ and $\phi_A \otimes \mathbf{1}_B = \phi_A \otimes \mathbf{1}_{B\setminus A}$ if $B \not\subseteq A$. Conceptually, we can think of this operation as creating $|B \setminus A|$ copies of ϕ_A. We use the convention that $\phi_\emptyset \equiv 1$ and define $\gamma \otimes \phi_A = \gamma\phi_A$ for all $\gamma \in \mathbb{R}$ which amounts to multiplying all cells in ϕ_A by γ.

Suppose we are given evidence on the variables $E \subseteq A$. Entering evidence into a potential, ϕ_A, is done in two steps: First, all cell values that do not agree with the evidence are set to zero. Second, we realize that the modified potential effectively has domain $A \setminus E$ and remove the dimensions of the ϕ_A corresponding to E. We refer to this as *evidence-reduction* and write $\partial_E \phi_A$ to denote the resulting potential with domain $A \setminus E$.

Inconsistent evidence is evidence, which happens with probability zero. Inconsistent evidence on a set of variables $E \subseteq A$ is equivalent to $\partial_E \phi_A = \mathbf{0}_{A\setminus E}$, and in this case the Bayesian network becomes degenerate. Evidence reduction is central for reducing the complexity, both in the memory storage and the time it takes to multiply the potentials.

For disjoint sets A and B, write (ϕ_A, B, γ) for the *full potential* $\phi_A \otimes \gamma\mathbf{1}_B$, with domain $A \cup B$. We say that ϕ_A is the *partial potential*, B is the set of *unity variables* and $\gamma \in \mathbf{R}$ is the *weight*. This triple object induces a more compact representation of the full potential since the unit potential, $\mathbf{1}_B$, does not have to be stored in memory. We just store the levelset of B. That is, the multiplication $\phi_A \otimes \mathbf{1}_B$ is never carried out. It follows from (2) that

$$(\phi_{A_1}, B_1, \gamma_1) \otimes (\phi_{A_2}, B_2, \gamma_2) = \big(\phi_{A_1} \otimes \phi_{A_2}, (B_1 \cup B_2) \setminus (A_1 \cup A_2), \gamma_1\gamma_2\big) \tag{4}$$

	b	
a	b^+	b^-
a^+	5	7
a^-	6	0

(a)

	c	
b	c^+	c^-
b^+	3	0
b^-	8	4

(b)

	c^+		c^-	
a	b^+	b^-	b^+	b^-
a^+	15	56	0	28
a^-	18	0	0	0

(c)

Table 1: Partial potentials: (a) $\phi_{\{a,b\}}$ (b) $\phi_{\{b,c\}}$ (c) $\phi_{\{a,b,c\}}$.

and from (3) that

$$(\phi_A, B, \gamma)^{\downarrow C} = \begin{cases} (\phi_A^{\downarrow A \cap C}, B \cap C, \gamma |I_{B \setminus C}|), & \text{for } A \cap C \neq \emptyset \\ (1, C, \gamma |\phi_A| |I_{B \setminus C}|), & \text{for } A \cap C = \emptyset, \end{cases} \tag{5}$$

for $C \subseteq A \cup B$.

Example

Consider the two potentials $\psi_{\{a,b,c\}} = (\phi_{\{a,b\}}, \{c\}, 2)$ and $\psi_{\{b,c,e\}} = (\phi_{\{b,c\}}, \{e\}, 3)$, where $\phi_{\{a,b\}}$ and $\phi_{\{b,c\}}$ are given in Tables 1a and 1b. The partial potential $\phi_{\{a,b,c\}} = \phi_{\{a,b\}} \otimes \phi_{\{b,c\}}$ of the product

$$\psi_{\{a,b,c\}} \otimes \psi_{\{b,c,e\}} = (\phi_{\{a,b,c\}}, \{e\}, 6),$$

is given in Table 1c. In comparison, Table 2 shows the full potential $\phi_{\{a,b,c\}} \otimes \mathbf{1}_{\{e\}} \cdot 6$, which obviously does not contain any additional information compared to $(\phi_{\{a,b,c\}}, \{e\}, 6)$. In fact, it is nothing but $\phi_{\{a,b,c\}}$ copied $|I_e|$ times and finally multiplied by 6.

		e^+		e^-	
a	b	c^+	c^-	c^+	c^-
a^+	b^+	90	0	90	0
	b^-	336	168	336	168
a^-	b^+	108	0	108	0
	b^-	0	0	0	0

Table 2: The full potential $(\phi_{\{a,b,c\}}, \{e\}, 6)$.

Recently, Lindskou et al. (2021) introduced a representation of *sparse potentials*, called *sparta*, and defined multiplication, division, and projection on these together with open source software (Lindskou, 2021c) in the **R** programming language R Core Team (2020). Let ϕ_A be a potential with levelset I_A, then the corresponding sparse potential has the *sparse levelset*

$$\mathcal{I}_A = \{i_A \in I_A \mid \phi_A(i_A) \neq 0\}$$

consisting of *non-zero cells*. Thus, the sparse version of $\phi_{\{a,b,c\}}$ has the sparse levelset

$$\{(a^+, b^+, c^+), (a^+, b^-, c^+), (a^+, b^-, c^-), (a^-, b^+, c^+)\}$$

consisting of four cells. Another interesting representation of sparse tables called value-based potentials (VBPs) was given in Gómez-Olmedo et al. (2021). VBPs can compress data more than sparta potentials, but it takes longer time to multiply and project for large potentials.

The method presented in this paper can be implemented for both *ordinary potentials* and sparse potentials. In the rest of the paper, if nothing else is stated, a potential can be of either type. However, the methods we introduce are aimed at enhancing belief propagation using sparse tables since multiplication with a unity potential inflates sparse tables unnecessarily and ruins the table's sparsity. We use the same notation for operators on both ordinary and sparse potentials.

2.3 Smoothing

Given data, the parameters of a CPT are typically estimated using maximum-likelihood estimation. Given a BN with $|V|$ nodes, $X_1, X_2, \dots, X_{|V|}$, define

$$\theta_k(i,j) = p(X_k = i \mid \pi_{X_k} = j), \quad i \in I_{X_k}, j \in I_{X_{\pi_k}}, k = 1, 2, \dots, |V|.$$

Denote by $n_k(i,j)$ the number of observations where $X_k = i$ and $\pi_{X_k} = j$. The maximum likelihood estimates (MLEs) are then given by

$$\hat{\theta}_k^{MLE}(i,j) = \frac{n_k(i,j)}{\sum_i n_k(i,j)},$$

where the denominator is the number of observations where the parents have cell j. We take $\hat{\theta}_k^{MLE}(i,j) \equiv 0$ if $\sum_i n_k(i,j) = 0$ i.e., if the parent cell, j, has not been seen. Hence, the MLE is zero for all zero-cells. In this paper, we are in particular concerned with observed zero-cells, i.e., cells for which $\hat{\theta}_k^{MLE}(i,j) = 0$, that arise from inconsistent evidence.

Assume all parameters are estimated using maximum likelihood estimation. We can then treat new observations as evidence, enter it into the model, and query for different posterior probabilities given the evidence. This is also known as the *all-marginal problem*, which is usually solved using the junction tree algorithm (see Section 4). However, for inconsistent evidence, the model becomes degenerate at zero since the concerned CPTs collapses to the null potential. A typical remedy of inconsistent evidence is to apply Laplace smoothing to all cells in every CPT by adding a pseudo-count α to each cell. The estimated parameters then take the form

$$\hat{\theta}_k^{LP}(i,j) = \frac{n_k(i,j) + \alpha}{\sum_i n_k(i,j) + \alpha |I_{X_k}|}. \tag{6}$$

These estimates equal the expected value of the posterior distribution of $\theta_k(i,j)$, using a symmetric Dirichlet distribution with α as prior. In practice, α should be chosen carefully as showed in Steck (2008). Nonetheless, it is standard to choose $\alpha = 1$ (**?**). *Structural zeroes* are zero-cells that will remain zero-cells regardless of the amount of data. That is, structural zeroes correspond to events that happen with probability zero. Laplace smoothing will inevitably turn structural zeroes into non-zero cells. However, applying Laplace smoothing in sparse CPTs will repeal the sparsity. In the following section, we provide a new method for smoothing that is essential for sparse tables, and optional for ordinary tables.

3 Unity Smoothing

Consider the sparse CPT

$$p(X = i \mid \pi_X = j) = \frac{p(X = i, \pi_X = j)}{p(\pi_X = j)}.$$

Let $j^* = (j_E^*, j_R)$ where $j_E^* \in I_E$ is observed as evidence, and $j_R \in I_R$ is not. If $p(\pi_X = j^*) = 0$, the CPT is not defined, and it seems natural to set

$$p(X = i \mid \pi_X = j^*) = 1/|I_X|, \tag{7}$$

for all $i \in I_X$ since there is no prior knowledge for the case of $\pi_X = j^*$ when $n(j^*) = 0$. Notice, that this corresponds to Laplace smoothing, see (6).

We shall in the following make a simple assumption which is most easily explained by an example. If X has state space $I_X = \{x^+, x^-\}$ in the model, we do not allow to insert the evidence $X = x^*$ for $x^* \notin I_X$. Consequently, inconsistent evidence cannot occur unless there is evidence on two or more variables. Assume now that $p(\pi_X = j^*) > 0$ and define

$$A_0(j^*) = \{i \in I_X \mid p(X = i \mid \pi_X = j^*) = 0\}, \quad A_+(j^*) = \{i \in I_X \mid p(X = i \mid \pi_X = j^*) > 0\}.$$

Hence, the set $A_0(j^*)$ consists of child indices $i \in I_X$ for which $p(X = i \mid \pi_X =$

$j^*) = 0$, corresponding to zero-cells. When $A_0(j^*) = \emptyset$, there is no need for smoothing. Therefore, assume that $A_0(j^*) \neq \emptyset$. For a small positive number ϵ, the smoothed probabilities are then set to

$$p_\epsilon(X = i \mid \pi_X = j^*) = \begin{cases} \epsilon, & i \in A_0(j^*), \\ p(X = i \mid \pi_X = j^*)\big(1 - \epsilon|A_0(j^*)|\big), & i \in A_+(j^*), \end{cases}$$

where it should be noticed that $\sum_{i \in I_X} p_\epsilon(X = i \mid \pi_X = j^*) = 1$, whenever j_R is also observed. Hence, it follows that probabilities for which $i \in A_+(j)$ are scaled according to the number of zero-cells. Suppose now that $i = i^* \in I_X$ is observed. If $i^* \in A_0(j^*)$, we smooth and set

$$p_\epsilon\big(X = i^* \mid \pi_X = j^*\big) = \epsilon,$$

for all $j_R \in I_R$. Thus, after smoothing, the evidence-reduced CPT is represented as $(1, R, \epsilon)$. This approach is different from adding the pseudo-count, α, to all cells as in Laplace smoothing, which affects all CPTs. Here, we only change those CPTs with inconsistent evidence and leave all other CPTs intact. By doing so, sparsity is not repealed for sparse CPTs, and in fact, the sparsity is increased since the unity potential $\mathbf{1}_R$ does not have to be stored.

After message passing, the probability of the entered evidence can be extracted without further computations. If this probability is of no interest, ϵ can be disregarded and we can therefore set $\epsilon = 1$. Otherwise, if the evidence is inconsistent and the probability of evidence is needed, one must choose a value of ϵ. See Section 4 and the example in Section 5.1 for details about the probability of evidence. As such, inference is independent of ϵ, whereas Laplace smoothing depends on the smoothing parameter α.

3.1 Example

Consider again Table 1c and assume that the table is used to construct the CPT $p(a \mid b, c)$. First, consider the case with inconsistent evidence on the parents, e.g., $\{b = b^+, c = c^-\}$. In this case, the resulting table after evidence reduction is identical for both Laplace smoothing and unity smoothing, and the result is $(a^+ = 1/2, a^- = 1/2)$, regardless of the choice of α. Next, consider the case where there is inconsistent evidence on the child, a, and the parent b, e.g. $\{b = b^-, a = a^-\}$. The resulting evidence-reduced potential after Laplace smoothing is $\phi_L = (c^+ = \alpha/(56 + 2\alpha), c^- = \alpha/(28 + 2\alpha))$, whereas for unity smoothing $\phi_U = (c^+ = \epsilon, c^- = \epsilon)$. If $\alpha = 1$, it follows that $\phi_L = (c^+ = 1/58, c^- = 1/30) \approx \frac{1}{58}(c^+ = 1, c^- = 2)$. This illustrates that the more uniform the cell counts are on the remaining parent variables, the more similar are Laplace and unity smoothing.

4 The Junction Tree Algorithm with the LS Scheme

Consider a DAG, G, with the set of nodes given by V. The junction tree algorithm (Lauritzen and Spiegelhalter, 1988, Højsgaard et al., 2012) consists of several steps. First, the DAG is *moralized*, meaning that all pairs of nodes that share a common child are connected by an edge, and all directions are dropped resulting in an undirected graph G^M. If the moralized graph is not *triangulated*, *fill-in edges* are added to the moralized graph until all cycles of length ≥ 4 have a *chord*, i.e., an edge connecting two non-neighbors in the cycle. The triangulated graph is denoted as G^T. A *junction tree* is a tree whose set of nodes is the (maximal) cliques, $\mathcal{C}$, of G^T, and where each pair of neighboring clique nodes, C_1 and C_2, share a *separator* $S = C_1 \cap C_2$. A junction tree satisfies that the separator between any two cliques is contained in all cliques on the unique path between these two cliques. This property is also known as the *running intersection property*.

Once a junction tree is constructed, each CPT $p(X \mid \pi_X)$ is associated with a clique, C, such that $\{X, \pi_X\} \subseteq C$. Denote by Φ_C the set of CPTs that are associated with clique C. If we observe evidence on the variables E, all CPTs in Φ_C are evidence-reduced, and the resulting tables are multiplied together to *initialize* the clique potential, ϕ_C. When all CPTs that contain E have been evidence-reduced, and all clique potentials have been initialized, we say that the model has been initialized. If $\Phi_C = \emptyset$, we set $\phi_C = \mathbf{1}_C$. Furthermore, to each separator, S, we associate the unity potential $\mathbf{1}_S$. Before the message passing can begin, a root node is chosen in order to specify the direction of the messages.

The LS message passing scheme is performed in two passes, *collect* and *distribute*. When collecting, the root node starts by collecting messages from all of its neighbors. However, a node is only allowed to distribute a message if it has collected a message from all of its *outward neighbors*, i.e., the neighbors that have already themselves collected messages. Thus, if a node has no outward neighbor, it begins sending messages. Clique C_2 collects a message from C_1 by: computing the message $\phi_{C_1}^{\downarrow S}$, set $\phi_{C_2} \leftarrow \phi_{C_2} \otimes \phi_{C_1}^{\downarrow S}$, and update ϕ_{C_1} as $\phi_{C_1} \leftarrow \phi_{C_1} / \phi_{C_1}^{\downarrow S}$. When the root node, say C_0, has collected all its messages, the root potential is normalized, i.e.,

$$\phi_{i_{C_0}} \leftarrow \phi_{C_0} / |\phi_{C_0}|,$$

and the collecting phase has ended. At this stage, it can be shown that the root potential is the joint distribution of the variables in the root clique. Furthermore, if evidence was entered into any of the CPTs or the clique potentials before the collecting phase, it holds that before the normalization, the normalizing constant $|\phi_{C_0}|$ equals the probability of observing the evidence.

In the distributing phase, each node distributes a message to its outwards neighbors, when it has collected a message from all of its *inwards neighbors*, i.e., the neighbors that have already distributed messages. The root node is the only node that can start by distributing a message. Clique C_2 distributes a message to C_1 by setting $\phi_{C_1} \leftarrow \phi_{C_1} \otimes \phi_{C_2}^{\downarrow S}$ and then update the separator potential $\phi_S \leftarrow \phi_{C_2}^{\downarrow S}$.

When both the collecting and distributing phases have ended, all clique and separator potentials are identical to the conditional distribution defined over the variables involved given the evidence.

5 Unity Propagation

Assume that the collecting phase has begun, and we are ready to send a message from clique C_1 to clique C_2. We denote by $C_1^* = C_1 \setminus E$ and $C_2^* = C_2 \setminus E$ the corresponding evidence-reduced cliques, for some evidence variables E. That is, $C_1^* \subseteq C_1$ and $C_2^* \subseteq C_2$. Let $\psi_{C_1^*} = (\phi_{A_1}, B_1, \gamma_1)$ and $\psi_{C_2^*} = (\phi_{A_2}, B_2, \gamma_2)$ be the clique potentials where $A_j \cup B_j = C_j^*$ and $A_j \cap B_j = \emptyset$ for $j = 1,2$. Notice that if $A_j \cup B_j \neq C_j$, the variables $C_j \setminus (A_j \cup B_j)$ are evidence variables. Denote by $S = C_1^* \cap C_2^*$ the evidence-reduced separator. Unity propagation arises in the following four scenarios:

i no partial potential needs to be multiplied when sending a message,

ii no partial potential needs to be divided when updating a node,

iii partial potentials must be multiplied, and $(B_1 \cup B_2) \setminus (A_1 \cup A_2)$ is non-empty

iv multiplication with an inconsistent CPT is avoided due to unity smoothing.

Scenario (i) happens if and only if $A_1 \cap S = \emptyset$ or $A_2 = \emptyset$ (or both), which follows directly from (4) and (5). For $A_1 = \emptyset$, we can even avoid marginalization as the message equals $|I_{B_1 \setminus S}|\mathbf{1}_S$, and we only need to pass on the constant $|I_{B_1 \setminus S}|$. Thus, scenario (i) is given by

$$(\phi_{A_1}, B_1, \gamma_1)^{\downarrow S} \otimes (\phi_{A_2}, B_2, \gamma_2) =$$
$$\begin{cases} (\phi_{A_1}^{\downarrow A_1 \cap S}, B_2 \setminus (A_1 \cap S), |I_{B_1 \setminus S}|\gamma_1\gamma_2), & \text{when } A_1 \cap S \neq \emptyset \text{ and } A_2 = \emptyset \\ (\phi_{A_2}, B_2, |\phi_{A_1}||I_{B_1 \setminus S}|\gamma_1\gamma_2), & \text{when } A_1 \cap S = \emptyset \text{ and } A_2 \neq \emptyset. \end{cases}$$

When $A_1 \cap S \neq \emptyset$ and $A_2 = \emptyset$, we simply replace the (empty) partial potential, ϕ_{A_2}, with the partial potential of the message and change the unity potential and weight appropriately. For $A_1 \cap S = \emptyset$ and $A_2 \neq \emptyset$, it is enough to update the weight. The case of $A_1 \cap S = \emptyset$ and $A_2 = \emptyset$ follows trivially from the above. Scenario (ii) happens when $A_1 \subseteq S$, where

$$\phi_{A_1} / \phi_{A_1}^{\downarrow S} = (1, A_1, \gamma_1),$$

and no division is needed. Thus, in the collecting phase, some clique potentials may turn into unities, and scenario (i) can be exploited again during the distributing phase. The computational savings in scenario (iii) are illustrated in Section 2.2 by the example where

$$(\phi_{\{a,b\}}, \{c\}, 2) \otimes (\phi_{\{b,c\}}, \{e\}, 3) = (\phi_{\{a,b,c\}}, \{e\}, 6).$$

Here, $(B_1 \cup B_2) \setminus (A_1 \cup A_2) = \{e\}$ and the full potential amounts to creating an extra copy of $\phi_{\{a,b,c\}}$ which is shown in Table 2. The savings in Scenario (iv) are immediate.

As discussed in Section 3, all weights can be neglected, i.e., set to one if the probability of evidence is of no interest.

5.1 Example

Consider the DAG, G, with nodes $V = \{a, b, c, d, e\}$ in Figure 1(a) with a triangulated graph, G^T, shown in Figure 1(b), and a junction tree with root C_1 in Figure 1(c). The joint pmf factorizes as

$$p_V = p_{a|b} p_b p_{c|b,e} p_{d|a,b} p_e p_{f|d,e},$$

where we have omitted $\otimes-$products for readability. Now, let

$$\Phi_{C_1} = \{p_e, p_{f|d,e}\}, \quad \Phi_{C_2} = \{\mathbf{1}_{C_2}\} \quad \Phi_{C_3} = \{p_{c|b,e}\}, \quad \Phi_{C_4} = \{p_b, p_{a|b}, p_{d|a,b}\}.$$

We say that C_2 is a *unity clique* since no CPT was associated with this clique. If, instead of associating p_b with C_4, it was associated with C_2, no unity cliques would have been created. Assume now that we have observed evidence on the variables $E = \{b, c\}$ and that this induces inconsistent evidence in $p_{c|b,e}$, i.e. $\partial_E p_{c|b,e} = \mathbf{0}_e$. Then we apply unity smoothing and initialize

$$\psi_{C_3^*} = (1, \{e\}, \epsilon),$$

where the asterisk symbolizes that the domain of the clique has been reduced by the evidence. Moreover, the remaining clique potentials are also evidence-reduced (if needed), and we initialize the remaining clique potentials as follows:

$$\psi_{C_4^*} = (\phi_{C_4^*}, \emptyset, 1) \quad \text{and} \quad \psi_{C_2^*} = (1, \{d, e\}, 1) \quad \text{and} \quad \psi_{C_1^*} = (\phi_{C_1^*}, \emptyset, 1),$$

where

$$\phi_{C_4^*} = (\partial_{\{b\}} p_b)(\partial_{\{b\}} p_{a|b})(\partial_{\{b\}} p_{d|a,b}) \quad \text{and} \quad \phi_{C_1^*} = p_e p_{f|d,e}.$$

Let S_{ij} be the (evidence-reduced) separator between clique C_i and C_j. The collecting phase can begin, and the messages

$$\psi_{C_3^*}^{\downarrow S_{23}} = \psi_{C_3^*} \quad \text{and} \quad \psi_{C_4^*}^{\downarrow S_{24}} = (\phi_{C_4^*}^{\downarrow \{d\}}, \emptyset, 1)$$

are sent to C_2^*. In other words, update C_2^* as

$$\psi_{C_2^*} \leftarrow \psi_{C_2^*} \otimes \psi_{C_3^*}^{\downarrow S_{23}} \otimes \psi_{C_4^*}^{\downarrow S_{24}} = (\phi_{C_4^*}^{\downarrow \{d\}}, \{e\}, \epsilon).$$

Moreover, update C_3^* and C_4^* as

$$\psi_{C_3^*} \leftarrow \psi_{C_3^*} / \psi_{C_3^*}^{\downarrow S_{23}} = (1, \{e\}, 1) \quad \text{and} \quad \psi_{C_4^*} \leftarrow \psi_{C_4^*} / \psi_{C_4^*}^{\downarrow S_{24}} = (\phi_{C_4^*} / \phi_{C_4^*}^{\downarrow \{d\}}, \emptyset, 1).$$

The root clique, C_1^*, is now ready to collect its message

$$\psi_{C_2^*}^{\downarrow S_{12}} = \psi_{C_2^*},$$

and C_1^* is updated as

$$\psi_{C_1^*} \leftarrow \psi_{C_1^*} \otimes \psi_{C_2^*}^{\downarrow S_{12}} = \big(\phi_{C_1^*} \otimes \phi_{C_4^*}^{\downarrow \{d\}}, \emptyset, \{e\}, \epsilon\big).$$

Finally, define $\eta = \epsilon |\phi_{C_1^*} \otimes \phi_{C_4^*}^{\downarrow \{d\}}|$ and normalize C_1^* as $\psi_{C_1^*} \leftarrow \psi_{C_1} / \eta$. Then, we have obtained

$$\phi_{C_1^*} \equiv p_{d,e,f},$$

and the probability of the evidence equals η. In summary, only one multiplication and one division with partial potentials were carried out. Without unity propagation, four multiplications and four divisions were needed.

6 Experiments

We conducted several experiments to investigate the effect of unity propagation both in terms of prediction accuracy and the execution time of the junction tree algorithm. All experiments can be reproduced via the research

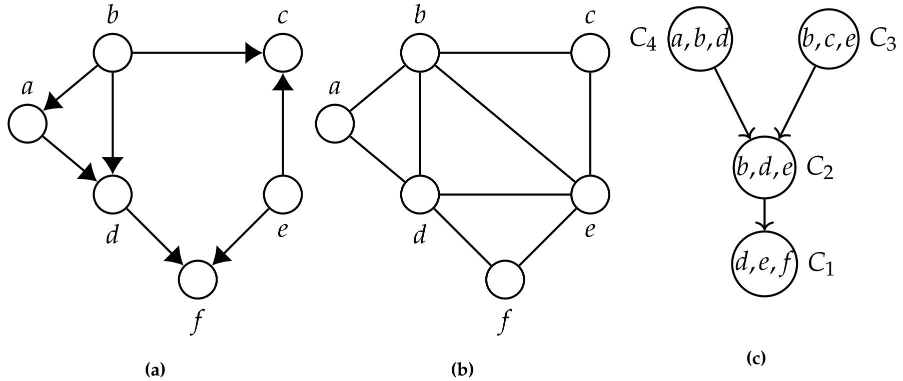


Fig. 1: (a) A DAG G. (b) The triangulated graph, G^T, of G, which equals G^M since no fill-ins are needed. (c) A rooted junction tree representation of (b) with root C_1.

compendium found at `https://github.com/mlindsk/unity_propagation_research_compendium`. We use the **R** package **jti** (Lindskou, 2021b) that uses sparse potentials from the **sparta** package as back-end for table operations. Our hypothesis is that unity propagation and unity smoothing, in general, enhances the inference time at very little expense of prediction accuracy.

6.1 Prediction Error and Inference Time with Inconsistent Evidence

We use the public available data sets `adult`, `chess`, `mushroom`, `derma`, `credit`, and `parkinson` from the UCI machine learning repository (Dua and Graff, 2017). The two latter datasets contains numerical variables, which we have discretized. For all datasets, incomplete observations were removed. The resulting number of observations and variables for each data set are summarized in Table 3. For each data set, we used the **R** package **ess** (Lindskou, 2021a) to fit a Bayesian network.

The number of observations, the number of variables, the number of cliques, the size of the largest clique, $|C_{max}|$, and the distribution of sparsity of the CPTs are reported in Table 3. The distribution of the CPT sparsity is defined on the unit interval, [0,1], where 0 means no sparsity, i.e., no zero-cells, and 1 means that all cells are zero-cells. The distributions reveals that most CPTs in `derma`, `mushroom` and `parkinson` are more than 0.75 sparse. On the other hand, most CPTs in `credit` have sparsity less than 0.25, whereas the CPTs in `adult` and `chess` are approximately 0.50 sparse on average.

Each data set contains a class variable, which we use to benchmark the prediction error of the junction tree algorithm using unity propagation.

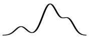
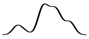
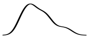
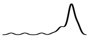
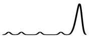
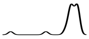
Dataset	#Obs	#Vars	#Cliques	$\lvert C_{max}\rvert$	CPT Sparsity
`adult`	30,162	15	11	5	
`chess`	3,196	37	30	7	
`credit`	653	16	12	5	
`derma`	385	35	29	5	
`mushroom`	5,644	23	16	6	
`parkinson`	195	23	18	5	

Table 3: Meta information of the UCI data sets used in the benchmarking. $\lvert C_{max}\rvert$ is the number of variables in the largest clique. The distributions of CPT sparsity is defined on the unit interval, $[0,1]$, where 0 means no sparsity, i.e., no zero-cells, and where 1 means only zero-cells.

Prediction Error

For each network, we calculated a $10-$fold cross validation score, which equals the prediction error. For a given fold, denote by D_{train} the training data and by D_{test} the test data. All parameters in the model were estimated using D_{train}, and we predicted the class of the observations in D_{test} as follows. Let z be the current observation in D_{test}, where we can think of z as a cell with $|V|$ entries, where $|V|$ is the number of variables in the data set, and the $|V|'$th entry corresponds to the class variable. Then, we successively chose $q = 2,3,\ldots,|V|-1$ entries from z at random and entered this information into the model as evidence. The collecting phase was then conducted to the root clique (containing the class variable by design) and we calculated the posterior distribution of the class variable given the evidence using a Bayes classifier. That is, the class label with the highest probability was used as the prediction. Thus, for each data set, we calculated $|V|-4$ cross-validation scores, where the first determines the prediction error we make when the model contains evidence on two variables, the second determines the prediction error we make when the model contains evidence on three variables, and so forth. As the number of evidence variables increases, so does the probability of inconsistent evidence since there are more observations in D_{test} that were never seen in D_{train}

The cross validation score was recorded where we used sparse tables with unity smoothing (dashed curves) and dense tables with Laplace smoothing with $\alpha = 1$ as the smoothing parameter (solid curves), see Figure 2.

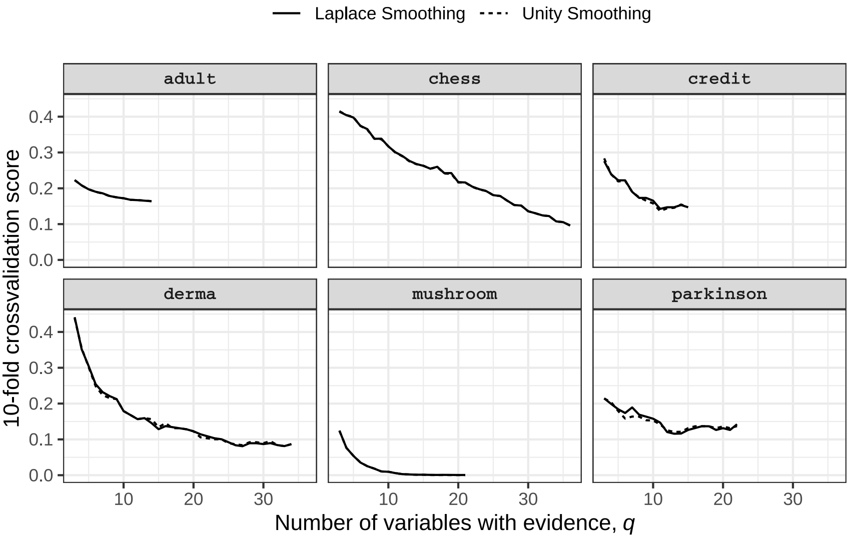


Fig. 2: Trajectories of 10−fold crossvalidation scores of six datasets based on the number of evidence variables, q.

The prediction accuracy differ slightly in `credit`, `derma`, and `parkinson`. For `credit` and `derma`, the differences are negligible, whereas for `parkinson`, unity smoothing seem to perform slightly better when $5 < q < 10$. The data `parkinson` only has 195 observations, but the CPTs are rather sparse, indicating that too many zero-cells may have been Laplace smoothed. For `adult` and `chess`, which both have larger numbers of observations and less sparse CPTs, there is no difference in prediction accuracy. Surprisingly, there is no difference between the two methods in `mushroom` even though the CPTs are very sparse.

Computation Time

As the number of inconsistent CPTs increases, the size of the clique potentials reduces when using unity smoothing leading to faster computation time. To benchmark run time performance, we ran the junction tree algorithm with and without unity propagation. Here, without unity propagation means that when a message has to be sent to a unity clique, we actually populate the unity clique with 1*s* and calculate the (unnecessary) product. We use a similar setup as in Butz et al. (2018). For each $q \in \{2, 4, \dots, |V|-1\}$, we randomly generated 200 sets of evidence, and for each q, measure the computation time for the junction tree algorithm to propagate the 200 sets of evidence. We report the ratio of computation time between unity and non-unity propagation for each q, see Figure 3. The ratio of computation time is given as

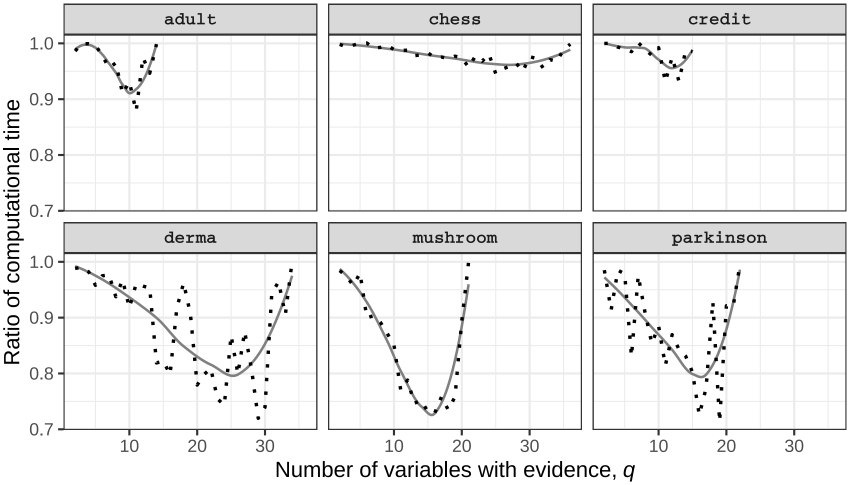


Fig. 3: Trajectories of the ratio of computational time during the junction tree algorithm between unity propagation and non-unity propagation. The solid lines are smoothed curves based on the actual measurements represented with dashed lines.

the computation time for unity propagation relative to computation time for non-unity. Hence, a value below one favors unity propagation. We first notice that unity propagation is consequently faster. In the worst case, the two methods are identical. The fluctuations of the actual measurements are pronounced in `derma` and `parkinson`, i.e., the datasets with fewest observations. The savings in computational time seem to be proportional to the sparsity of the CPTs, as one would expect. The largest saving in computational time occurs in `mushroom`, which also has the most sparse CPTs. On the other hand, `credit` has the most dense CPTs, and the savings in computational time is less than for all other datasets. For all datasets the trend is that when q is very small or very large, the savings are close to none. When q is small, almost no CPTs have inconsistencies, and when q is large, most clique potentials are reduced to scalars for both unity and non-unity propagation. However, in all cases, there is a large number of q's for which the savings in computational time is significant.

Interestingly, it seems that the largest savings, measured as the value of the critical point of the solid curve, occurs when there is evidence on approximately 70% of the variables: `adult` ($10/15 \approx 0.67$), `chess` ($26/37 \approx 0.70$), `credit` ($12/16 = 0.75$), `derma` ($25/35 \approx 0.71$), `mushroom` ($16/23 \approx 0.70$) and `parkinson` ($17/23 \approx 0.74$). The savings decrease on both sides of the critical point.

6.2 Inference Time for Unity Cliques Emerging from Triangulation and Initialization

In this section, we benchmark the gain of unity propagation on the classic expert Bayesian networks listed in Table 4, where we do not insert any evidence. Thus, the gain of unity propagation in this benchmark is solely from unity cliques emerging from triangulation and initialization. Table 4 lists the number of unity cliques and the size of the largest unity clique, $|U_{max}|$. Interestingly, for all networks the largest clique is a unity clique except for munin. Also, the more complex the network is, measured in number of variables and number of cliques, the more unity cliques there is. Again, define the ratio of computation time as the computation time for unity propagation relative to computation time for non-unity. Figure 4 shows the ratio of computational time against the ratio of unity cliques to non-unity cliques. There is a clear trend indicating that the higher the ratio of unity cliques, the larger the computational savings is which is to be expected. Two networks stand out; link and mildew. For link, the ratio of unity cliques to non-unity cliques is the smallest of all networks although the computational savings is the second best. This is due to the very large unity cliques, where the largest unity clique alone contains $16{,}777{,}216$ cells. In mildew, the CPTs are very sparse, and hence, the gain of unity propagation is pronounced.

BN	#Vars	#Cliques	#Unity Cliques	$\lvert C_{max}\rvert$	$\lvert U_{max}\rvert$
andes	223	178	49	17	17
asia	8	6	1	3	3
barley	48	36	7	8	8
diabetes	413	337	96	5	5
hailfinder	56	43	6	5	5
insurance	27	19	3	7	7
link	724	591	77	16	16
mildew	35	29	7	5	5
munin	1,041	872	114	9	8
pigs	441	368	71	11	11
win95pts	76	50	9	9	9

Table 4: Meta information of expert networks used in the benchmark.

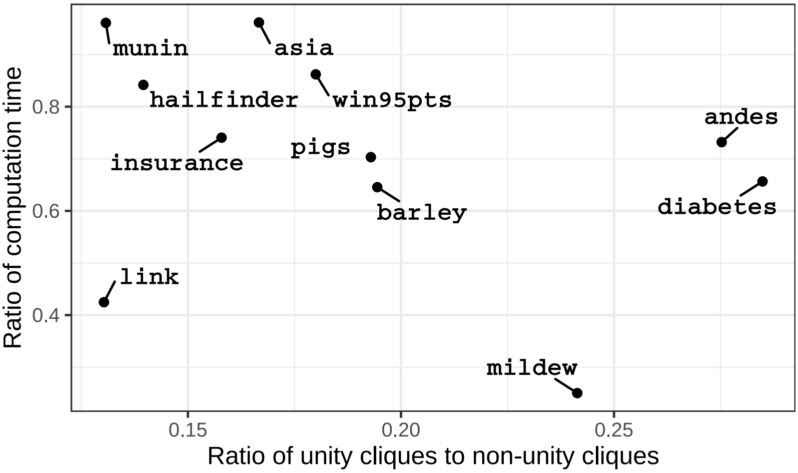


Fig. 4: Ratio of computational time of the junction tree algorithm between unity propagation and non-unity propagation against the ratio of unity to non-unity cliques. See Table 4 for meta information of the networks.

7 Conclusion

We proposed a new smoothing technique called *unity smoothing* that overcome the problem of inconsistent evidence for Bayesian networks with sparse tables. Unity smoothing also works for ordinary tables. Moreover, we introduced a set of rules called *unity propagation* that, by adhering to these, ensure fewer calculations during message passing in the junction tree algorithm. Through experiments we have shown the usefulness of both unity smoothing and unity propagation in terms of prediction accuracy and faster inference time.

References

Butz, C. J., Oliveira, J. S., dos Santos, A. E., and Madsen, A. L. (2018). An empirical study of bayesian network inference with simple propagation. *International Journal of Approximate Reasoning*, 92:198–211.

Cowell, R. G., Dawid, P., Lauritzen, S. L., and Spiegelhalter, D. J. (2007). *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Gómez-Olmedo, M., Cabañas, R., Cano, A., Moral, S., and Retamero, O. P.

(2021). Value-based potentials: Exploiting quantitative information regularity patterns in probabilistic graphical models. *International Journal of Intelligent Systems*.

Højsgaard, S., Edwards, D., and Lauritzen, S. (2012). *Graphical models with R*. Springer Science & Business Media.

Jensen, F., Lauritzen, S., and Olesen, K. (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194.

Lindskou, M. (2021a). ***ess**: Efficient Stewise Selection in Decomposable Markov Random Fields*. R package version 1.1.2.

Lindskou, M. (2021b). ***jti**: Junction Tree Inference*. R package version 0.8.0.

Lindskou, M. (2021c). ***sparta**: Tables*. R package version 0.8.1.

Lindskou, M., Højsgaard, S., Eriksen, P. S., and Tvedebrink, T. (2021). sparta: Sparse tables and their algebra with a view towards high dimensional graphical models. *arXiv preprint arXiv:2103.03647*.

Madsen, A. L. and Jensen, F. V. (1999). Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1-2):203–245.

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Shafer, G. R. and Shenoy, P. P. (1990). Probability propagation. *Annals of mathematics and Artificial Intelligence*, 2(1):327–351.

Steck, H. (2008). Learning the bayesian network structure: Dirichlet prior versus data. *Appears in Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI2008).* . arXiv preprint arXiv:1206.3287.

ISSN (online): 2446-1636
ISBN (online): 978-87-7573-937-0

AALBORG UNIVERSITY PRESS