Introduction to Structure Learning for Gaussian and Pair Copula Bayesian Networks.

Master thesis submitted to Delft University of Technology
in partial fulfilment of the requirements for the degree of
MASTER OF SCIENCE
in **Applied Mathematics**
Faculty of Electrical Engineering, Mathematics and Computer Science

by

Amadeo Villar Guardia
Student number: 5377447

To be defended in public on 21st December 2022

Graduation committee:

Supervisor:
Dr. Dorota Kurowicka. Faculty of EMMCS. Applied Probability Group.

Committee Members:
Dr. Gabriele Florentina Nane. Faculty of EMMCS. Applied Probability Group.
Dr. Alexis Derumigny. Faculty of EMMCS. Statistics Group.

ABSTRACT

Due to technological breakthrough in recent decades and the rapid increase in the availability of multidimensional data, data science has become one of the most important areas of research. Within this field, modeling dependence of random variables is gaining great interest. To cope with this task, the use of graphical models is often advocated. In this dissertation, we study Bayesian Networks (BNs), a particular type of graphical models. Concretely, structure learning algorithms for two types of continuous BNs: Gaussian Bayesian Networks (GBNs) and Pair Copula Bayesian Network (PCBNs) are investigated.

We present an overview of these two types of BNs, illustrating its properties and differences. An outline of the different existing structure learning algorithms is provided, showing their efficiency for the Gaussian case and limitations for the copula based. The problems of structure learning for PCBNs are then addressed. We investigate the performance of Gaussian structure learning algorithms for PCBNs. Based on a simulation study, we show that these procedures are not completely efficient, but prove beneficial. Second, a new approximation of the score based on logLikelihood of PCBNs is explored. We propose to solve the computational inefficiency of the exact logLikelihood by estimating the necessary copulas from data such that the copula terms in the PCBNs decomposition can be computed without need of integration. A simulation study suggests that this logLikelihood approximation yields better results than the approximation used by Pircalabelu et al. (2017). Finally, an algorithm to learn the structure of PCBNs is proposed, based on the 2 previous procedures.

Contents

LIST OF FIGURES

List of Tables

ABBREVIATIONS

Abbreviation	**Concept**
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
BN	Bayesian Network
CDF	Cumulative Distribution Function
CPD	Conditional Probability Distribution
DAG	Directed Acyclic Graph
GBN	Gaussian Bayesian Network
Ham	Hamming distance
HC	Hill Climbing
IC	Information Criterion
IFM	Inference Function for Margins
KS	Kolmogorov Smirnov
FN	False Negatives
FP	False Positive
logLik	logLikelihood
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimation
PCBN	Pair-Copula Bayesian Network
PC	Peter Spirtes and Clark Glymour algorithm
PCC	Pair-Copula Constructions
PDF	Probability Density Function
PIT	Probability Integral Transformation
SHD	Structural Hamming Distance
TP	True Positive

NOMENCLATURE

Mathematical Symbol	**Concept**
n	Number of variables involved
$C_{1,\dots,n}$	Copula Cumulative Distribution Function
$c_{1,\dots,n}$	Copula Probability Density Function
τ	Kendall's tau
V	Regular Vine
T_i	i-th Tree in a Vine
N_i	i-th Set of Nodes in the Vine
E_i	i-th Set of Edges in the Vine
$e(j,k)$	Edges of the Vine
D_e	Conditioning Set of the Edge $e(j,k)$
U_i	Uniform random variable
$\mathrm{CV}_{1,\dots,n}$	C-Vine
$\mathrm{DV}_{1,\dots,n}$	D-Vine
N	Sample Size
N	Gaussian Copula
G	Gaussian Copula
C	Clayton Copula
G	Gumbell Copula
F	Frank Copula
J	Joe Copula
I	Independent Copula
r_n	Number of Vines in n dimensions
$\mathcal{G}$	Graph
$\mathbb{V}$	Set of vertices/nodes
v_i	Vertex
E	Set of edges
n	Number of variables involved
$pa(v_i)$	Parents of node v_i
$ch(v_i)$	Children of node v_i
$an(v_i)$	Ancestors of node v_i
$de(v_i)$	Descendants of node v_i
non-de(v_i)	Descendants of node v_i
$\mathbb{C}$	Cycle
$<$	Order
$M[\mathcal{G}]$	Moralised graph
$\mathrm{sep}_{\mathcal{G}}(\mathbf{V}_I;\mathbf{V}_J\|\mathbf{V}_K)$	Separation in undirected graphs
$\text{d-sep}_{\mathcal{G}}(\mathbf{V}_I;\mathbf{V}_J\|\mathbf{V}_K)$	Separation in directed graphs
$\mathcal{P}$	Probabilistic structure
$\boldsymbol{X}$	Multivariate random vector
X_i	Continuous random variable
$f_{X_i}(x_i)$	Probability density function of X_i
$F_{X_i}(x_i)$	Cumulative distribution function of X_i

Mathematical Symbol	**Concept**
$\boldsymbol{\Theta}$	Set of parameters
$\mathscr{B}$	Bayesian Network
$\mathscr{I}(\mathscr{G})$	Set of conditional Independencies encoded in $\mathscr{G}$
$\mathscr{I}(\mathscr{P})$	Set of conditional independencies that hold in $\mathscr{P}$
$\mathscr{N}$	Gaussian distribution
$\boldsymbol{\mu}$	Mean vector Gaussian distribution
$\boldsymbol{\Sigma}$	Covariance matrix Gaussian distribution
$\boldsymbol{\Omega}$	Precision matrix Gaussian distribution
$\Sigma_{i,j}$	Component i-th row, j-th column of $\boldsymbol{\Sigma}$
$\boldsymbol{\Sigma}_i$	i-th row of $\boldsymbol{\Sigma}$
$\mathbb{E}$	Expectation
Var	Variance
Cov	Covariance
$<_v$	Order in the node v
$\mathscr{O}$	Set of parental orderings
$\mathscr{D}$	Data
λ	Regularisation Parameter
$\hat{\boldsymbol{\theta}}^{\boldsymbol{\lambda}}$	Matrix of L^1 regressors
$\boldsymbol{X}_{\mathscr{D}}$	Data Matrix with dimensions $N \times n$
$\boldsymbol{\mathit{\Theta}}$	Set of $n \times n$ matrices with 0 on the diagonal
H_0	Null Hypothesis
H_1	Alternative Hypothesis
α	Significance level
$T_\alpha(x_i, x_j; \boldsymbol{x}_K)$	Decision rule Hypothesis Testing
$\rho_{X_i,X_j\|\boldsymbol{X}_k}$	Conditional Correlation of X_i and X_j given $\boldsymbol{X}_K$
P	Correlation Matrix
$ad(v_i)$	Set of edges adjacent to v_i
p_n	Number of DAGs in n dimensions
l	Number of random restarts in Hill-Climbing algorithm
Φ	CDF of the Gaussian random variable
Q_i	Quantile i

ACKNOWLEDGEMENTS

Esta tesis está dedicada a mi padre, Amadeo y a mi madre, Leonor. Gracias por apoyarme en cada una de las decisiones que he tomado en la vida y gracias por haber sido y por ser un referente en esta vida, que me ha permitido llegar a ser la persona que soy hoy. Sin vosotros nada de esto sería posible.

También me gustaría dar las gracias a mi supervisora, Dorota, por todo el tiempo dedicado a guiarme a lo largo de este proceso de investigación y a mejorar la calidad del trabajo. Así como al resto de miembros del tribunal, por tomarse el tiempo de leer este documento.

Por último, mención especial para toda mi familia, Pudrio Fans a la cabeza, y también para toda la gente maravillosa que he conocido y que me han acompañado durante esta etapa en Holanda: mis flatmates, Sunday, Bouldering bible adepts, mis pongos putitos, la Chapel, la Rama-gang, compis de master...

1
Introduction

"As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality."

Albert Einstein

1.1. Motivation

Due to technological breakthrough in recent decades and the rapid increase in the availability of multidimensional data, data science has become one of the most important areas of research. Within this field, modeling dependence of random variables (Joe, 1997) is gaining great interest. One example from the area of finance where the dependence is of interest would be an investment problem, in which neglecting the dependence between stock returns might result in a poor investment strategy and, eventually, a loss of money.

The most popular approach to model dependencies between continuous random variables is to use the multivariate Gaussian distribution. Indeed, this distribution provides computational simplicity and efficient sampling algorithms (Scutari and Denis, 2021). However, it is also generally acknowledged that multivariate Gaussian distribution is quite restrictive (all marginal distributions are Gaussian, there is no tail dependence) and does not provide adequate model in many situations (e.g. in finance (McNeil et al., 2015)).

To make it possible to build distributions with various types of one dimensional marginal distributions Sklar (1959) introduced the so-called copulas. Copulas are multivariate cumulative distribution functions on the unit hypercube with uniform margins. The importance of copulas resides in Sklar's theorem, which states that every multivariate joint distribution may be expressed in terms of univariate marginal distribution functions and a copula that represents the dependency structure of the variables. In high dimension the copula is still a complicated object to model. The flexible construction of multivariate copulas is obtained by specifying the product of bivariate and conditional bivariate copulas associated with a graphical structure of a vine Joe (1997),(Bedford and Cooke, 2002). This construction is referred to, in the literature, as vine copula or pair copula construction (PCC).

Vine copula is a graphical model that is most often used to represent dependencies between random variables in a flexible way. This is in contrast to other graphical models (directed, undirected) (Lauritzen, 1996, Spirtes et al., 2000) that utilize graphs to allow representation of (conditional) independence between random variables. The nodes represent random variables and edges dependencies between these random variables. Lack of connection between nodes in a graph can be interpreted with conditional independencies present in the joint distribution corresponding to the model.

This dissertation focuses on Bayesian Networks (BNs) which are graphical models whose underlying structure is a directed acyclic graph (Pearl, 1988). BNs are composed of: a Directed Acyclic Graph[1] (DAG) and a set of Conditional Probability Distributions (CPDs). The DAG provides a compact representation of the set of conditional independence in the distribution. Moreover, BNs can be seen as a structure that allow to factorised the joint distribution as the product of CPDs specified for this graph.

The moralised graph of the BN will also be crucial during this project. This graph gives a connection between directed and undirected graphs. It is the result of placing undirected edges between nodes that are connected by directed edges, and between nodes that share common children. The importance of the moralized graph of a BN is that it is possible to read from it, part of the conditional independencies that exist in the distribution.

There are different types of BN, depending on whether the random variables involved are all discrete, all continuous, or a mixture of both types. In this dissertation only the case where all the variables involved are continuous is discussed. In that case the structure of a graph leads to specification of the joint probability density (PDF) of random variables, in the form of the product of a non-unique sequence of conditional densities of variables given their parents in the graph.

Continuous BNs are gaining popularity due to their wide range of applications, which range from medicine (Vepa et al., 2021), to biology (Sachs et al., 2005) or natural disaster risk analysis (Li et al., 2010). See Pourret et al., 2008, for a review of different applications. Despite its extensive applicability, continuous BNs have been mostly limited to the case where the joint distribution is multivariate Gaussian. We call them Gaussian Bayesian Networks (GBNs).

In Kurowicka and Cooke (2004) copula based BNs were discussed. They demonstrated that every continuous multivariate distribution associated with a DAG, can be expressed as the product of bivariate copulas corresponding to the underlying graph's edges. The ides presented in this paper were further analysed in A. M. Hanea et al. (2006) and Bauer et al. (2012), where they have been referred to as Pair Copula Bayesian Network (PCBNs). These models may account for a wide range of distributional features, such as tail dependency, non-linearity, and asymmetric dependence. Thus relaxing the Gaussian assumptions.

[1]All the edges have a single direction and no loops are allowed

In application BNs are often used by specifying the structure which is assessed by experts A. Hanea et al., 2015. In this thesis we will consider learning the structure of BN model with data-driven methods.

1.2. Research Problem

This dissertation addresses the problem of Structure Learning of GBNs and PCBNs using data driven methods. Three different approaches are studied: the Graphical Lasso (Meinshausen and Bühlmann, 2006), Constraint-based and Score-based algorithms (Koller and Friedman, 2009).

The Graphical Lasso results in an undirected graph corresponding to the moralised graph of the BN. This method progresses by fitting a regression for each of the variables, using the rest as predictors and imposing a penalty of L^1 (Tibshirani, 1996) on regression coefficients. The method was developed under Gaussian assumptions, so it may lead to misspecification of the structure when applied to PCBN data.

Constraint-based algorithms derive the DAG through a sequence of conditional independence tests. For the Gaussian case 0-conditional (partial) correlation implies conditional independence between random variables. Therefore, a simple and efficient test can be applied to check the presence of conditional independence between random variables, just by looking at the correlation matrix. In case the data is not Gaussian another type of test is needed. An example could be a test of the independence of conditional copula between random variables (Bauer and Czado, 2016). However, these tests are very expensive and cannot be efficiently used to learn structures of BNs.

Lastly, score-based algorithms approach the problem of structure learning as an optimization problem. They assign a score to each network based on how well it fits the data. Then they select the graph that optimizes that score. Often used scores are log-Likelihood based. For the Gaussian case score-based algorithms are fast and efficient. The non-Gaussian case is very different. Then the LogLikelihood based scores involve the computation of non-analytic integrals. Expensive Monte Carlo methods (Hammersley, 1964) must then be used for their evaluation. These calculations make algorithms based on these scores computationally very expensive.

To ease the computational issues of score based algorithms for PCBNs Elidan (2010) proposed a score based on fitting a multivariate copula for each term in the DAG density factorization. However, this method usually requires high-dimensional copulas that lack flexibility. Pircalabelu et al. (2017) proposed a score based on quotient of Vines for each term in the DAG density factorization. This approximation is more flexible, but it does not generally produce any consistent joint distribution, causing structure misspecification.

1.3. Knowledge Gap

In this thesis we start our investigations with the application of methods designed for GBNs to PCBNs. We apply Gaussian structure learning algorithms for the non-Gaussian case. The main goal is to examine if these procedures are still sufficiently efficient for PCBN data and will allow us to find the structure of the graph.

Then we compare the results of GBN algorithms with ones designed for PCBN data. Our contribution is to explore a new approximation of the score based on logLikelihood of PCBNs. We propose to solve the computational inefficiency of the exact logLikelihood by estimating the necessary copulas from data such that the copula terms in the PCBNs decomposition can be computed without need of integration. The main goal is to see if this approximation yield better results than those proposed by Pircalabelu et al. (2017). Simulation studies are carried out to compare different methods.

1.4. OBJECTIVES

The objectives of this dissertation are:

1. To present an overview of two types of continuous BNs:
   - Gaussian Bayesian Networks. Showing their good computational properties, but also their lack of flexibility.
   - Pair Copula Bayesian Network. Highlighting their greater flexibility, but also their computational inefficiency.
2. To provide an outline of the different existing structure learning algorithms:
   - Showing how efficient they are for GBNs.
   - Illustrating all the computational problems that arise when dealing with PCBNs.
3. To assess how Gaussian structure learning algorithms work for GBNs vs PCBNs.
4. To study how good the new PCBNs logLikelihood approximation is, and compare it with the score proposed by Pircalabelu et al. (2017).

1.5. OUTLINE

The outline of the dissertation is as follows:

- Chapter 2 provides an introduction to copula and Vine copula models.
- Chapter 3 introduces Bayesian Networks
- Chapter 4 presents an overview of two types of continuous BN: GBNs and PCBNs, addressing objective 1.
- Chapter 5 includes an outline of the different existing structure learning algorithm, achieving objective 2.

$\Rightarrow$ **Literature Review**

- Chapter 6 assess how Gaussian structure learning algorithm work for GBNs and PCBNs, tackling objective 3.
- Chapter 7 studies different scores for Score-based algorithms for PCBNs, attaining objective 4.

$\Rightarrow$ **Methodology**

- Chapter 8 discusses the findings, concludes and propose different lines of future work.

2
Copula and Vine Copula Models

This chapter provides an overview of Copula and Vine Copula frameworks to model dependence of random variables. Simulations and examples of these models are included, illustrating the great flexibility they offer.

2.1. Copulas

Gaussian distribution is one of the most popular approaches for modeling dependence. However, most of the data we encounter in practice possess properties such as skewness, tail dependence or heavy tails. In such cases the Gaussianity assumption is violated and there is a need for different techniques.

To be able to model distributions with various types of one dimensional margins copulas have been introduced. A copula is a multivariate cumulative distribution function on the unit hypercube $[0,1]^n$, which has uniform margins. The importance of these copulas resides in Sklar's Theorem (Sklar, 1959).

Theorem 2.1 *Let $\boldsymbol{X} = (X_1, \dots, X_n)$ be random vector with joint Cumulative Distribution Function (CDF) $F_{X_1,X_2,\dots,X_n}$ and with margins $F_{X_1}, \dots, F_{X_n}$. Then there exists a copula $C_{1,2,\dots,n} : [0,1]^n \to [0,1]$ such that, $\forall\ x_1, \dots, x_n \in \mathbb{R}^n$,*

$$F_{X_1,X_2,\dots,X_n}(x_1, \dots, x_n) = C_{1,2,\dots,n}(F_{X_1}(x_1), \dots, F_{X_n}(x_n)). \tag{2.1}$$

If $X_1, \dots, X_n$ are continuous then $C_{1,2,\dots,n}$ is unique.

Copulas allow us to describe the dependence between the different variables. The above theorem can be rewritten for densities by differentiating the previous expression:

$$f_{X_1,X_2,\dots,X_n}(x_1, \dots, x_n) = c_{1,2,\dots,n}(F_{X_1}((x_1), \dots, F_{X_n}(x_n)) f_{X_1}(x_1) f_{X_2}(x_2) \cdots f_{X_n}(x_n), \tag{2.2}$$

where $c_{1,2,\dots,n}$ is the copula density function and f_{X_i} are the marginal densities.

There are many types of copulas that are able to represent tail dependencies, asymmetries etc. In high dimensions, however, the most flexible model results from decomposing the multivariate copula as the product of conditional bivariate copulas, assigned to a graphical structure called vine.

This section then focuses on two-dimensional copulas. In particular, we deal only with parametric copula families. In this dissertation we parameterize them by means of the dependence measure called Kendall's τ. This choice facilitates the comparison between different copula families. The relationship between a copula and Kendall's τ is:

$$\tau = 4 \int\int_{[0,1]^2} C_{1,2}(u_1, u_2) dC_{1,2}(u_1, u_2) - 1. \tag{2.3}$$

In the literature, numerous copulas with different properties are found (Joe, 2014). In this dissertation the following parametric families are considered: Gaussian, t, Clayton, Gumbell, Frank and Joe [1]. In Figure 2.1 examples of scatter plots of these copula families are shown. The overall dependence of all these bivariate copulas is the same, $\tau = 0.75$.

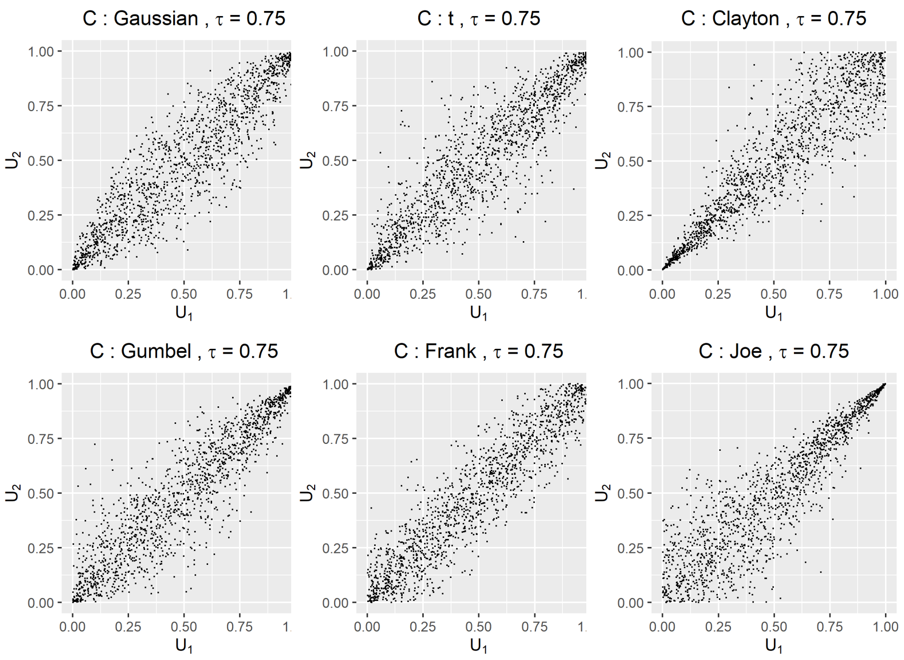


Figure 2.1: Scatter plots of bivariate copulas: Gaussian, t, Clayton, Gumbell, Frank and Joe.

Despite having the same overall dependence, their properties are very different. Indeed, Clayton, Gumbel and Joe copulas are not symmetric. Clayton copula has lower tail dependence and upper tail independence, in contrast, Gumbel copula exhibits upper tail

[1] Their rotated versions will also be used.

dependence and lower tail independence. Frank and Gaussian copula have both upper and lower tail independence. On the other hand, t copula symmetric and has both lower and upper tail dependence.

VISUAL ANALYSIS AND GOODNESS OF FIT TEST

To analyse, assess and compare different copulas presented in this dissertation, the following methodology will be used:

1. Qualitative methods. They are based on visual analysis of data by means of scatter plots using the `ggplot2` package [2]. They have two main purpose:

   (a) To check whether data is sampled correctly (see for instance Figure 4.7).

   (b) To compare data generated from a copula with data used to fit that copula (see for instance Figure 7.4).

2. Quantitative methods.

   - Comparison of different models based on quantities of interest: values of copulas Kendall's τ specified in (2.3), as well as values of the logLikelihoods and penalized logLikelihoods (Information Criteria).
   - Hypothesis testing, used as goodness of fit test. To check whether we accept or reject the null hypothesis: H_0 : data comes from the fitted copula. For this purpose, we use the tests implemented in the `VineCopula` package (Nagler, 2021) using the command `BiCopGofTest`. These tests are based on the so-called Kendall process proposed by Wang and Wells (2000), where the Cramer-von Mises and Kolmogorov Smirnov statistics are included.

As we previously introduced, more flexible models can be built using the product of bivariate copulas. This approach is called the Pair Copula Construction (PCC), initially introduced by Joe (1997). The most widely researched models arising from PCCs are the so-called Vine Copulas introduced by Bedford and Cooke (2002).

2.2. VINE COPULA

In this section an overview of Vine Copula models is presented. First, Construction and Sampling are shown. Estimation and Structure Learning of vines are then discussed.

2.2.1. VINE DECOMPOSITION

Any copula density $c_{1,2,\dots,n}$ can be decomposed into a product of $n \times (n-1)/2$ bivariate copula densities (Bedford and Cooke, 2002). Below, we follow the procedures outlined in Aas et al. (2009) to show the ideas behind the PCC.

- $\boldsymbol{n = 2}$

[2] https://ggplot2.tidyverse.org/

Applying (2.2) to the two-dimensional case with the variables X_1 and X_2, we obtain:

$$f_{X_1,X_2}(x_1,x_2) = c_{1,2}\left(F_{X_1}(x_1), F_{X_2}(x_2)\right) \cdot f_{X_1}(x_1) \cdot f_{X_2}(x_2). \tag{2.4}$$

Dividing the left-hand side by the term $f_{X_2}(x_2)$ and using the definition of conditional probability:

$$\boxed{f_{X_1|X_2}(x_1|x_2) = c_{1,2}\left(F_{X_1}(x_1), F_{X_2}(x_2)\right) \cdot f_{X_1}(x_1).} \tag{2.5}$$

- $\boldsymbol{n = 3}$

The joint probability distribution of X_1, X_2 and X_3 can be expressed as:

$$f_{X_1,X_2,X_3}(x_1,x_2,x_3) = f_{X_1|X_2,X_3}(x_1|x_2,x_3) \cdot f_{X_2|X_3}(x_2|x_3) \cdot f_{X_3}(x_3). \tag{2.6}$$

Let's see how the above equation can be formulated in terms of bivariate copulas. We first start focusing on $f_{X_1|X_2,X_3}$:

$$f_{X_1,X_3|X_2}(x_1,x_3|x_2) = c_{1,3|2}\left(F_{X_1|X_2}(x_1|x_2), F_{X_3|X_2}(x_3|x_2); x_2\right) \cdot f_{X_1|X_2}(x_1|x_2) \cdot f_{X_3|X_2}(x_3|x_2).$$

Dividing the left-hand side by the term $f_{X_3|X_2}(x_3|x_2)$, using the definition of conditional probability and using (2.5) to replace the value of $f_{X_1|X_2}(x_1|x_2)$, we get:

$$\begin{aligned} f_{X_1|X_2,X_3}(x_1|x_2,x_3) = & c_{1,3|2}\left((F_{X_1|X_2}(x_1|x_2), F_{X_3|X_2}(x_3|x_2); x_2\right) \\ & \cdot c_{1,2}\left(F_{X_1}(x_1), F_{X_2}(x_2)\right) \cdot f_{X_1}(x_1). \end{aligned} \tag{2.7}$$

Replacing (2.7) and using (2.5), we get:

$$\boxed{\begin{aligned} f_{X_1,X_2,X_3}(x_1,x_2,x_3) = & c_{1,3|2}\left(F_{X_1|X_2}(x_1|x_2), F_{X_3|X_2}(x_3|x_2); x_2\right) \cdot c_{1,2}\left(F_{X_1}(x_1), F_{X_2}(x_2)\right) \\ & \cdot c_{2,3}(x_2,x_3) \cdot f_{X_1}(x_1) \cdot f_{X_2}(x_2) \cdot f_{X_3}(x_3). \end{aligned}} \tag{2.8}$$

However, applying the same procedure for the conditional function $f_{X_1,X_2|X_3}$ instead of for $f_{X_1,X_3|X_2}$, it follows that:

$$\boxed{\begin{aligned} f_{X_1,X_2,X_3}(x_1,x_2,x_3) = & c_{1,2|3}\left(F_{X_1|X_3}(x_1|x_3), F_{X_2|X_3}(x_2|x_3); x_3\right) \cdot c_{1,3}\left(F_{X_1}(x_1), F_{X_3}(x_3)\right) \\ & \cdot c_{2,3}(x_2,x_3) \cdot f_{X_1}(x_1) \cdot f_{X_2}(x_2) \cdot f_{X_3}(x_3). \end{aligned}} \tag{2.9}$$

This yields two equivalent decompositions: (2.8) and (2.9), each with distinct pair-copula terms. Actually, by choosing other orders of the variables that we condition in (2.6), we would obtain more equivalent structures.

These decompositions can be represented using graphical models that are called regular vines, namely a sequence of trees $T_i = (N_i, E_i)$.

Definition 2.1 (Vine Copula) *V is a regular vine on n elements if:*

1. $V = \{T_1, ..., T_{n-1}\}$
2. *T_1 is a tree with nodes $N_1 = \{1, \dots, n\}$, and edges E_1;*

   for $i = 2, \dots, n-1$, T_i is a tree with nodes $N_i = E_{i-1}$.
3. *(regularity) for $i = 2, \dots, n-1$, $\{a, b\} \in E_i$ and $a = \{a_1, a_2\}$, $b = \{b_1, b_2\}$ then exactly one of the a_i equals one of the b_i.*

Vine Copulas allow the decomposition of the joint density as the product of algebraically independent bivariate conditional copulas [3]. This factorization is achieved by associating a conditional bivariate copula to the edges of each tree. The copulas are assigned such that the conditioning variables correspond to the conditioning set and the conditioned variables to the conditioned set of each edge. These sets are defined below.

Definition 2.2 (Constraint, Conditioning and Conditioned set) *The constraint set, the conditioning set and the conditioned set are defined as:*

1. *The constraint set associated with $e(j,k) \in E_i$ is a subset of $N_1 = \{1, 2, \dots, n\}$ reachable from e by inclusion (membership) relationship denoted by U_e^*.*
2. *For $i = 1, \dots, n-1, e(j,k) \in E_i$, the conditioning set associated with e is:*

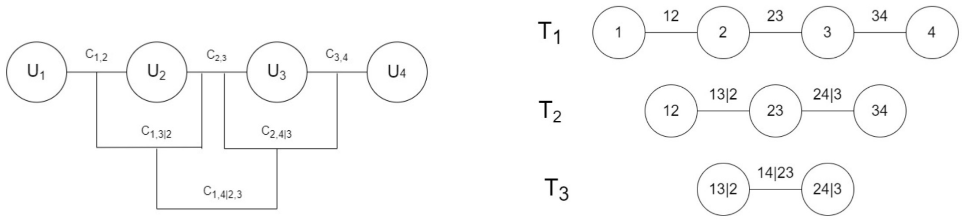
$$D_e = U_j^* \cap U_k^*,$$

   and the conditioned set associated with e is:

$$\{C_{e,j}, C_{e,k}\} = U_j^* \triangle U_k^* = \{U_j^* \setminus D_e, U_k^* \setminus D_e\}.$$

   The order of node e is $|D_e|$.

Example 2.1 (Vine Copula.) *To illustrate these concepts the 4-dimensional Vine Copula in Figure 2.2 is considered:*



(a) Graphical Representation of a Regular Vine.

(b) Trees of a Regular Vine.

Figure 2.2: Example of a 4-dimensional Regular Vine.

[3]There is no algebraic restriction on type of copula families and parameters that can be used for the construction.

We can observe three different trees: T_1, T_2 and T_3. T_1 has nodes $N_1 = \{1,2,3,4\}$ and edges $E_1 = \{(1,2),(2,3),(3,4)\}$. In T_2, we can see that $N_2 = E_1$ as defined above. Moreover, the constraint sets of the three edges of E_2 are $\{1,2,3\},\{2,3,4\}$. The conditioning sets are $\{2\},\{3\}$ and the conditioned sets are $\{1,3\},\{2,4\}$ respectively. Lastly, there are no constraint on the choice of families and parameters of: $C_{1,2}$, $C_{2,3}$, $C_{3,4}$, $C_{1,3|2}$, $C_{2,4|3}$ and $C_{1,4|2,3}$.

Combining Sklar's theorem and the decomposition showed above, Bedford and Cooke (2002) proved the following representation theorem of the joint PDF factorised using Vine Copula models:

Theorem 2.2 *Let $F_{X_1,X_2,\dots,X_n}$ be a CDF with margins F_{X_i} and density margins f_{X_i}, $i = 1,\dots,n$. Let $V = (T_1,\dots,T_{n-1})$ be a regular vine on n elements. Each edge $e(j,k) \in T_i$, $i = 1,\dots,n-1$, with conditioned set given by $\{j,k\}$ and conditioning set given by D_e, is associated with the corresponding copula $C_{j,k|D_e}$ and its density $c_{j,k|D_e}$. Then the distribution for the copula vine specification $(F_{X_1,X_2,\dots,X_n}, V)$ is uniquely determined and has a density given by:*

$$\begin{aligned} f_{X_1,\dots,X_n}(x_1,\dots,x_n) =& f_{X_1}(x_1)\cdots f_{X_n}(x_n) \\ & \cdot \prod_{i=1}^{n-1} \prod_{e(j,k)\in E_i} c_{j,k|D_e}(F_{X_j|\boldsymbol{X}_{D_e}}(x_{j|D_e}), F_{X_k|\boldsymbol{X}_{D_e}}(x_{j|D_e}); \boldsymbol{x}_{D_e}). \end{aligned} \tag{2.10}$$

Each of the bivariate copulas $c_{j,k|D_e}$ of the previous decomposition, describes the dependence between the random variables X_j and X_k conditional on $\boldsymbol{X}_{D_e}$.

One usually assumes that the bivariate conditional copula $c_{j,k|D_e}$ does not depend directly on the variables $\boldsymbol{X}_{D_e}$. This assumption is called the **simplifying assumption**(Hobæk Haff et al., 2010). This assumption greatly reduces the complexity of the models and will be used from now on. The reduction in the complexity is very useful but is also one of the reasons that different decompositions (different vine structures) differ in terms of performance in modeling a data set.

The conditionals CDFs $F_{X_j|\boldsymbol{X}_{D_e}}$ in (2.10) can be evaluated tree-by-tree using a recursive formula derived in Joe (1997), which says that for every $i \in N_1$, every $A \subset N_1 - i$ and an arbitrary $j \in A$:

$$F_{X_i|\boldsymbol{X}_A}(x_i|\boldsymbol{x}_A) = \frac{\partial C_{i,j;A\setminus\{j\}}\left(F_{X_i|\boldsymbol{X}_{A\setminus\{j\}}}(x_i|\boldsymbol{x}_{A\setminus\{j\}}), F_{X_j|\boldsymbol{X}_{A\setminus\{j\}}}(x_j|\boldsymbol{x}_{A\setminus\{j\}})\right)}{\partial F_{X_j|\boldsymbol{X}_{A\setminus\{j\}}}(x_j|\boldsymbol{x}_{A\setminus\{j\}})}. \tag{2.11}$$

Note that if $F_{X_i|\boldsymbol{X}_A}$ appears in (2.10), by the construction of the Vine, there is only one $j \in A$, such that the copula $C_{i,j|A\setminus\{j\}}$ belongs to the previous tree. Therefore, the only copulas needed in this computation are the ones already specified in the preceding trees. This result is fundamental, since it allows us to compute the joint density function in a closed form.

We can also consider the variables to be uniform such that (2.10) and (2.11) read as:

$$f_{U_1,\dots,U_n}(u_1,\dots,u_n) = \prod_{i=1}^{n-1} \prod_{e(j,k)\in E_i} c_{j,k|D_e}(u_{j|D_e}, u_{k|D_e}). \tag{2.12}$$

and

$$u_{i|A} = C_{i|A}(u_i|\boldsymbol{u}_A) = \frac{\partial C_{i,j;A\setminus\{j\}}\left(u_{i|A\setminus\{j\}}, u_{i|A\setminus\{j\}}\right)}{\partial u_{i|A\setminus\{j\}}}, \tag{2.13}$$

respectively [4].

Example 2.1 (Continued) *The joint probability density for the Vine Copula model in Figure 2.2 using simplifying assumptions is given by:*

$$\begin{aligned} f_{U_1,\dots,U_n}(u_1,\dots,u_4) =& c_{1,2}(u_1,u_2)\cdot c_{2,3}(u_2,u_3)\cdot c_{3,4}(u_3,u_4)\cdot \\ & c_{1,3|2}(u_{1|2},u_{3|2})\cdot c_{2,4|3}(u_{2|3},u_{4|3})\cdot c_{1,4|2,3}(u_{1|2,3},u_{4|2,3}). \end{aligned} \tag{2.14}$$

All the conditional samples can be computed, using the copulas defined in previous trees:

$$u_{1|2} = \frac{\partial C_{1,2}(u_1,u_2)}{\partial u_2}, u_{3|2} = \frac{\partial C_{2,3}(u_2,u_3)}{\partial u_2}, \; u_{1|2,3} = \frac{\partial C_{1,3|2}(u_{1|2},u_{3|2})}{\partial u_{3|2}}, \dots$$

CONDITIONAL INDEPENDENCES IN VINES

Some copulas in the decomposition (2.10) can be specified to be the independent copula [5]. This allows us to build models that include some conditional independencies. However, there might be more conditional independencies in the distribution that one constructs, except the ones directly specified by assigning the independence copula to some edges. Such a case will be later illustrated in Example 4.3.

TYPES OF VINES

Two different types of vines will be important later on in this thesis.

Definition 2.3 (C-Vine, D-Vine) *C and D-Vines are subclasses of the regular Vine, which are characterised by:*

- *C-Vine: In every tree there is a central node, which is connected to the rest of the nodes of the tree. So that all the trees are star-shaped, with the central node as the center. They will be denoted as* $CV_{1,\dots,n}$.
- *D-Vine: Each of the trees is chain-shaped, so that all nodes are connected with 2 nodes except the ends, which are only connected with 1 node. They will be denoted as* $DV_{1,\dots,n}$.

For more information about this type of vines, please see Brechmann and Schepsmeier (2013). For instance, the Vine Copula in Figure 2.2 is a D-Vine.

Other important types are the incomplete vines. These are vines from which some nodes have been removed, that is, the corresponding copula is set to be the independent copula. Within the incomplete vines the so-called m-saturated vines introduced by Kurowicka and Cooke, 2006a play a fundamental role in this project.

Definition 2.4 (m-saturated vine) *An incomplete vine is an m-saturated vine of a regular vine V if all descendants of a node in its node set belong to the node set of the incomplete vine.*[6]

[4]Notice that the derivatives of a copula are equal to conditional copulas
[5]This copula is used to model independence between random variables.
[6]Please see Kurowicka and Cooke (2006a) to understand the notation used.

Example 2.2 (m-saturated vine) *Let's illustrate the concept of m-saturated vine. We can observe in Figure 2.3 two different vines.*

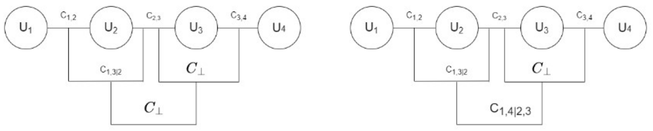


Figure 2.3: Illustration of m-saturated vine definition.

The vine on the left is m-saturated. Indeed, for the node $3,4$*, we have that its descendant nodes are* $2,4|3$ *and* $1,4|2,3$*. For these nodes, we have that* $C_{2,4|3} = C_{1,4|2,3} = C_{\perp\!\!\!\perp}$*. On the other hand, for the vine on the right* $C_{1,4|2,3} \neq C_{\perp\!\!\!\perp}$*. Therefore, it does not fulfil the previous definition.*

2.2.2. VINE SAMPLING

There are two strategies for sampling from a vine, the cumulative and density approaches. During this project, we only use the first one, which is illustrated by means of an example. The theory behind can be consulted in Kurowicka and Cooke (2006b). All computations shown over this section are obtained by using the `VineCopula` package.

Example 2.1 (Continued) *The sampling procedure for the vine copula model in Figure 2.2 is based on the fact that if* $v_1, v_2, v_3, v_4 \sim U(0,1)$ *are independent,* (u_1, u_2, u_3, u_4) *fulfilling:*

$$\begin{cases} u_1 = v_1, \\ u_2 = C^{-1}_{2|1;u_1}(v_2), \\ u_3 = C^{-1}_{3|1;u_1}\left(C^{-1}_{3|1,2;C_{1|2}(u_1|u_2)}(v_3)\right), \\ u_4 = C^{-1}_{4|3;u_3}\left(C^{-1}_{4|2,3;C_{2|3}(u_2|u_3)}\left(C^{-1}_{4|1,2,3;C_{1|2,3}(u_1|u_2,u_3)}(v_4)\right)\right). \end{cases} \tag{2.15}$$

are distributed as that Vine Copula. We choose the copula families and parameters to be:

	Family	*Kendall's* τ
$C_{1,2}$	*Gaussian: N*	*0.25*
$C_{2,3}$	*Clayton: C*	*0.25*
$C_{3,4}$	*Gumbell: G*	*0.75*
$C_{1,3\|2}$	*Frank: F*	*0.75*
$C_{2,4\|3}$	*Joe: J*	*0.75*
$C_{1,4\|2,3}$	*Independent: I*	-

```
tree      edge | family  cop    par  |  tau   utd   ltd
-------------------------------------------------------
  1        2,1 |      1    N   0.38  | 0.25     -     -
           3,2 |      3    C   0.67  | 0.25     -  0.35
           4,3 |      4    G   4.00  | 0.75  0.81     -
  2      3,1;2 |      5    F  14.14  | 0.75     -     -
         4,2;3 |      6    J   6.78  | 0.75  0.89     -
  3    4,1;3,2 |      0    I      -  | 0.00     -     -
---
type: D-vine  1 <-> U1, 2 <-> U2, 3 <-> U3, 4 <-> U4.
```

Taking $N = 2500$*, as a sample size and* $set.seed(1)$*, we carry out the sampling. The obtained results can be seen in Figure 2.4.*

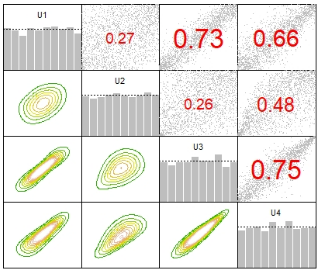


Figure 2.4: Bivariate contour plots, scatter plots, correlations and histograms of data simulated from our Vine Copula.

The computation time is:

$\boxed{\textit{Sampling time: 0.1999979 secs}}$

The logLikelihood of this sample with the true parameters is given by:

$\boxed{logLik = 7834.768}$

Applying the inverse of the CDFs, we obtain samples of the distribution: $x_i = F_i^{-1}(u_i)$.

2.2.3. VINE ESTIMATION

Estimation of the Vine copula model requires four steps.

1. Estimate the marginal distributions F_{X_i} [7] and transform the data $(x_1^m, \dots, x_n)$ into pseudo observations $(u_1, \dots, u_n)$ using the Probability Integral Transformation (PIT).
2. Select the Vine structure.
3. Choose copula families for each bivariate term.
4. Estimate copula parameters for each bivariate term[8].

$\left.\right\} \Rightarrow$ Dependence Structure

This estimation procedure is called the Inference Functions for Margins (IFM) method and was introduced by Joe (1997).

The problem of selecting the structure will be analysed in Subsection 2.2.4. Once the structure is selected, the Maximum Likelihood (ML) method is used to carry out steps 3 and 4. That is having the structure V, the pseudo observation $(u_1^m, \dots, u_n^m)$, with $m = 1, \dots, N$, the objective is to find the parameters $\boldsymbol{\theta}$ which maximise:

$$logLik(\boldsymbol{u},\boldsymbol{\theta}) = \sum_{m=1}^{N}\sum_{i=1}^{n-1}\sum_{e(j,k)\in E_i} log\left(c_{j,k|D_e}\left(C_{j|D_e}(u_j^m|\boldsymbol{u}_{D_e}^m, C_{k|D_e}(u_k^m|\boldsymbol{u}_{D_e}^m);\boldsymbol{\theta}\right)\right). \quad (2.16)$$

To compare different models with varying number of parameters we use Akaike information criterion (AIC) and Bayesian information criterion (BIC) given by:

$$\mathrm{AIC}(\boldsymbol{u},\boldsymbol{\theta}) = logLik(\boldsymbol{u},\boldsymbol{\theta}) - |\boldsymbol{\theta}|, \quad (2.17)$$

$$\mathrm{BIC}(\boldsymbol{u},\boldsymbol{\theta}) = \sum_{m=1}^{N}\sum_{i=1}^{n}\log\left(f(x_i^m|\boldsymbol{x}_{pa(v_i)}^m)\right) - \frac{|\boldsymbol{\theta}|}{2}\log(N). \quad (2.18)$$

[7] Either using parametric or non-parametric methods.

The model with the highest score will be the preferred one.

Moreover, to evaluate the fit of a Vine, some goodness of fit test can be used. Schepsmeier (2019) shows several tests which are implemented in the `VineCopula` package through the `RVineGofTest` command.

Finally, optimizing the Likelihood over the whole space of parameters is very expensive. Instead, the estimation is carried out tree by tree (Aas et al., 2009). First, the bivariate copulas in the first tree are estimated using the observations. We can then plug these estimated copulas and our observations in (2.13) to transform our data. These transformed data is then used to estimate the bivariate conditional copulas in the second tree, etc...

Example 2.1 (Continued) *Let's estimate the copula families and parameters for the simulated data from Figure 2.4 using the structure shown in Figure 2.2. The obtained results using the BIC as criterion are:*

```
tree     edge | family  cop    par  |  tau   utd   ltd
-----------------------------------------------------------
   1      2,1 |      1    N   0.40  | 0.26     -     -
          3,2 |      3    C   0.70  | 0.26     -  0.37
          4,3 |      4    G   4.01  | 0.75  0.81     -
   2    3,1;2 |      5    F  14.22  | 0.75     -     -
        4,2;3 |      6    J   6.62  | 0.74  0.89     -
   3  4,1;3,2 |      0    I      -  | 0.00     -     -
---
type: D-vine     logLik: 7827.79    AIC: 7822.79    BIC: 7808.23
---
1 <-> U1,  2 <-> U2,  3 <-> U3,  4 <-> U4, Computation time of 9.012099 secs.
```

We can see how well the estimation implementations work, since all the families are correctly recovered and the parameters are close to the real values. The computational times for this example are feasible, but as the dimension increases, they increase greatly.

2.2.4. VINE STRUCTURE SELECTION

Theoretically, any vine structure can be used to decompose a density (Zhu, 2022). In practice, however, because of the limited number of parametric bivariate copula families, the simplifying assumption and the tree by tree estimation procedure, the performance of different vine copula models vary.

Morales Napoles et al. (2010) showed that in n dimensions, the number of different regular vines structure r_n is given by:

$$r_n = \binom{n}{2}(n-2)!\,2^{\binom{n-2}{2}}. \tag{2.19}$$

As the number grows super exponentially with dimension, estimating all vine structures in high dimensions is computationally not feasible. Heuristic techniques are then needed for searching the best structure. The most popular one in the literature is the Dibbman's heuristic (Dißmann et al., 2013). It constructs the vine structure tree-wise. The first tree is determined by a maximum spanning tree with weights being the absolute Kendall's τ. The remaining trees are determined under the same premise, respecting the

Table 2.1: Number of possible regular vine structures depending on the dimension.

Dimension n	Number of vines r_n
2	1
3	3
4	24
5	480
6	23040

regularity condition in Definition 2.1. Other option proposed by Kurowicka (2011), consist on building a vine such that nodes of top trees correspond to the smallest absolute values of partial correlations. Nevertheless, these procedures are not optimal, and the challenge of structure selection remains as an open problem currently under research, see for instance Zhu (2022).

Example 2.1 (Continued) *Let's apply the Dibbman's algorithm to the simulated data showed in Figure 2.4 to find the structure. The obtained results are:*

```
tree     edge | family  cop    par  par2 |   tau   utd   ltd
------------------------------------------------------------
   1      3,1 |      5    F  13.54  0.00 |  0.74     -     -
          4,2 |     14   SG   1.89  0.00 |  0.47     -  0.56
          4,3 |      4    G   4.01  0.00 |  0.75  0.81     -
   2    4,1;3 |      6    J   1.19  0.00 |  0.10  0.21     -
        3,2;4 |     24  G90  -2.51  0.00 | -0.60     -     -
   3  2,1;4,3 |      0    I      -     - |  0.00     -     -
---
type: D-vine    logLik: 6755.18    AIC: 6750.18    BIC: 6735.62
---
1 <-> U1,   2 <-> U2,   3 <-> U3,   4 <-> U4, Comput. time: 0.6690509 secs
```

We can observe that the estimated structure is quite different from the true structure. The logLik of the obtained model is lower than the one for the true structure. Indeed, a relative error of 13.77% *is made. Finally, a goodness of fit test is performed to assess the resulting structure. Using the PIT method and the KS statistic, the obtained results are:*

```
Hypothesis testing:
$KS            $p.value
[1] 20.29188   [1] 0.575
```

Despite the large difference in logLik, this test does not reject the null hypothesis at significance level $\alpha = 0.05$.

This example illustrates how difficult it is to find the correct structure and that by using heuristics we cannot guarantee that the best model will be obtained. Nevertheless, heuristic methods can obtain acceptable results.

3
Background on Bayesian Networks

"You are smarter than your data.
Data do not understand causes and effects; humans do."

Judea Pearl

The objective of this chapter is to familiarize the reader with the ideas behind graphical models, particularly Bayesian Networks.

3.1. Concepts from Graph Theory

This section provides an introduction to graph theory that will be key to understanding graphical models in general, and BNs in particular.

Definition 3.1 (Graph) *A graph is a pair $\mathcal{G} = (\mathbb{V}, E)$ with:*

- $\mathbb{V} = \{v_1, \ldots, v_n\}$ *a finite set of nodes.*
- $E = \left\{\{v_i, v_j\}, v_i, v_j \in \mathbb{V} \text{ and } v_i \neq v_j\right\}$ *a set of edges between nodes.*
  Two types of edges are to be distinguished:
  1. *Undirected edges* $:= \left\{v_i, v_j\right\}$, *if they are unordered pairs. The graph is then called undirected graph.*
  2. *Directed edges* $:= \left(v_i, v_j\right)$ *if they are ordered pairs:* $v_i \rightarrow v_j$, *that is* $(v_i, v_j) \neq (v_j, v_i)$. *They are also called arcs and the graph is then called directed graph.*

  If a graph contains both directed and undirected edges, it is called partially directed graph.

Definition 3.2 (Parents, Children) *If there is an arc* $\left(v_i, v_j\right)$, *then* v_i *is a parent of* v_j *and* v_j *is a child of* v_i. *We also denote:*

- $pa(v_i) :=$ {*set of parents of* v_i}. *If* $pa(v_i) = \emptyset$, *then* v_i *is a root node.*
- $ch(v_i) :=$ {*set of children of* v_i}. *If* $ch(v_i) = \emptyset$, *then* v_i *is a leaf node.*

Definition 3.3 (Trail, Cycle) :

- *A trail going from node* a *to node* b *is a set of different edges* $\{\{v_{i-1}, v_i\} \in E,\ i = 2,\dots,k\}$, *such that* $a = v_1$ *and* $b = v_k$. *If the edges are undirected/directed, the trail is called undirected/directed trail, respectively.*
- *A non-empty trail, in which the first and last vertices are equal*: $v_1 = v_k$, *is a cycle. If there are no cycles, the graph is said to be acyclic. If the trail is undirected/directed, the cycle is called undirected/directed cycle, respectively.*

Definition 3.4 (Ancestors, Descendants) *If there is a directed trail from* v_i *to* v_j, *then* v_j *is a descendant of* v_i *and* v_i *is an ancestor of* v_j. *We also denote:*

- $an(v_i) :=$ {*set of ancestors of* v_i}.
- $de(v_i) :=$ {*set of descendant of* v_i}.

Example 3.1 (Graph) *Below we show an example of a Directed Acyclic Graph composed by 8 nodes and 11 arcs. Moreover, we illustrate definitions of sets of parents, children, ancestors and descendant presented above.*

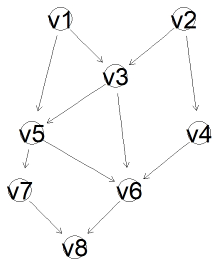


Figure 3.1: Example of a Directed Acyclic Graph $\mathcal{G}$.

$$\mathbb{V} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$

$$E = \{v_1 \to v_3, \dots, v_7 \to v_8\}$$

$$pa(v_6) = \{v_3, v_4, v_5\}$$

$$ch(v_5) = \{v_6, v_7\}$$

$$an(v_7) = \{v_1, v_2, v_3, v_5\}$$

$$de(v_3) = \{v_5, v_6, v_7, v_8\}$$

This network does not contain directed cycles, however, if we reversed the arc $v_3 \to v_6$, *we would obtain the directed cycle:*

$$\mathbb{C} = \{v_3 \to v_5 \to v_6 \to v_3\}.$$

3.1.1. DIRECTED ACYCLIC GRAPHS

Directed Acyclic Graphs (DAGs) play a fundamental role during this project. They allow ordering the vertices in such a way that the parents appear before the children in the ordering. This will enable us to factorise the joint probability function when using graphical models.

Definition 3.5 (Complete Order) *The vertices are completely ordered if there exists a relationship* " $<$ " *on elements of* $\mathbb{V} = \{v_1, \dots, v_n\}$ *such that* $\forall\ v_i, v_j, v_l \in \mathbb{V}$, *we have:*

1. *either* $v_i < v_j$, *or* $v_j < v_i$.
2. $v_i \nless v_i$.
3. *if* $v_i < v_j$ *and* $v_j < v_l$, *then* $v_i < v_l$.

Theorem 3.1 *In a directed graph, conditions 1 and 2 are equivalent.*

1. *There is no directed cycle.*
2. *There exists a complete ordering of the vertices such that parents are earlier in the ordering than children.*

The proof of this theorem can be seen in Appendix A.

Example 3.1 (continued) *The graph in Figure 3.1 is a Directed Acyclic Graph, and hence there exist a complete ordering. However, this ordering does not have to be unique. Indeed, two possible orderings for this example are:*

$$\{v_1 < v_2 < v_3 < v_4 < v_5 < v_6 < v_7 < v_8\},$$

$$\{v_2 < v_1 < v_4 < v_3 < v_5 < v_6 < v_7 < v_8\}.$$

3.1.2. MORALISED GRAPH

The moralised graph is an important concept that will be used in this dissertation. This graph gives a connection between directed and undirected graphs.

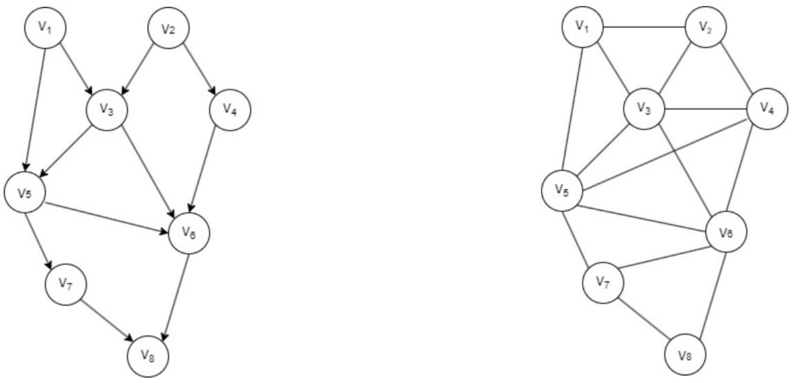
Definition 3.6 (Moralised Graph) *The moral graph* $M[\mathcal{G}]$ *of a Directed Acyclic Graph* $\mathcal{G}$ *is the undirected graph that contains an undirected edge between* v_i *and* v_j *if:*

- *There is a directed edge between* v_i *and* v_j *in* $\mathcal{G}$.
- v_i *and* v_j *are both parents of a node* $v_k \in \mathbb{V}$.

Example 3.1 (continued) *We can see in Figure 3.2 the moralised graph associated with the DAG shown above.*

3.1.3. SEPARATION IN GRAPHS

Separation is a key concept to study independence relationships in a set of variables corresponding to nodes of the graph.



(a) DAG $\mathscr{G}$. (b) Moralised graph $M[\mathscr{G}]$.

Figure 3.2: Example of a DAG $\mathscr{G}$ and its moralised graph $M[\mathscr{G}]$.

UNDIRECTED GRAPHS

Definition 3.7 (Blocked Trail in Undirected Graphs) *In an undirected graph, a trail between two different nodes v_i and v_j is said to be blocked by a set $\mathbf{V}_K$, if that trail contains a node v_k that is in $\mathbf{V}_K$.*

Definition 3.8 (Separation in Undirected Graphs) *If $\mathbf{V}_I$, $\mathbf{V}_J$ and $\mathbf{V}_K$ are three disjoint subsets of nodes in an undirected graph $\mathscr{G}$, then $\mathbf{V}_K$ is said to separate $\mathbf{V}_I$ from $\mathbf{V}_J$, denoted $sep_{\mathscr{G}}(\mathbf{V}_I;\mathbf{V}_J|\mathbf{V}_K)$, if all undirected trails between any nodes $v_i \in \mathbf{V_I}$ and $v_j \in \mathbf{V_J}$ contains a node $v_k \in \mathbf{V}_K$.*

We denote:

$$\mathscr{I}(\mathscr{G}) = \left\{\text{sep}_{\mathscr{G}}(\mathbf{V}_I;\mathbf{V}_J|\mathbf{V}_K)\ ,\ \mathbf{V}_I,\mathbf{V}_J,\mathbf{V}_K \in \mathbb{V}\right\}\ ,\ \text{if } \mathscr{G} \text{ is an undirected graph.} \tag{3.1}$$

We will refer to this set as the set of conditional independences encoded in the undirected graph $\mathscr{G}$.

Example 3.1 (continued) *Let's study the separation between v_1 and v_6 in the previously studied moralised graph. We identify in Figure 3.2b,* 10 *different undirected trails between these two nodes. It is easy to see that both $\mathbf{V} = \{v_2, v_3, v_5\}$ and $\mathbf{V} = \{v_3, v_4, v_5\}$ block all these previous trails. It can be concluded that:*

$$sep_{M[\mathscr{G}]}(v_1; v_6|v_2, v_3, v_5),$$

$$sep_{M[\mathscr{G}]}(v_1; v_6|v_3, v_4, v_5).$$

DIRECTED GRAPHS

In case of directed graphs, the separation criterion is extended to include one extra way how a trail can be blocked.

Definition 3.9 (Blocked trail in Directed Graphs) *An undirected trail between two different nodes v_i and v_j resulting from dropping directions on the directed graph, is said to be blocked by a set $\mathbf{V}_K$ if:*

- *Either that trail contains a node v_k that is in $\mathbf{V}_K$ and the connection at v_k is either serial or diverging, that is:*

$$(\rightarrow v_k \rightarrow, \leftarrow v_k \leftarrow, \leftarrow v_k \rightarrow)$$

- *Or that trail contains a node v_l such that v_l and its descendants are not in $\mathbf{V}_K$ and the connection at v_l is a converging connection, that is:*

$$(\rightarrow v_l \leftarrow)$$

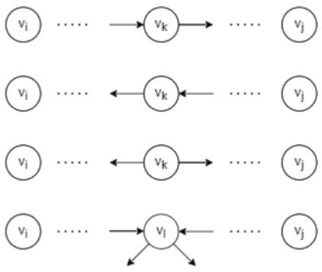


Figure 3.3: Serial, diverging and converging connections.

Definition 3.10 (Separation in Directed Graphs) *If $\mathbf{V}_I, \mathbf{V}_J$ and $\mathbf{V}_K$ are three disjoint subsets of nodes in a DAG $\mathscr{G}$, then $\mathbf{V}_K$ is said to d-separate $\mathbf{V}_I$ from $\mathbf{V}_J$, denoted d-sep$_{\mathscr{G}}(\mathbf{V}_I;\mathbf{V}_J|\mathbf{V}_K)$, if all undirected trails between any nodes $v_i \in \mathbf{V_I}$ and $v_j \in \mathbf{V_J}$ resulting from dropping directions in the directed graph, are blocked by nodes of $\mathbf{V}_K$.*

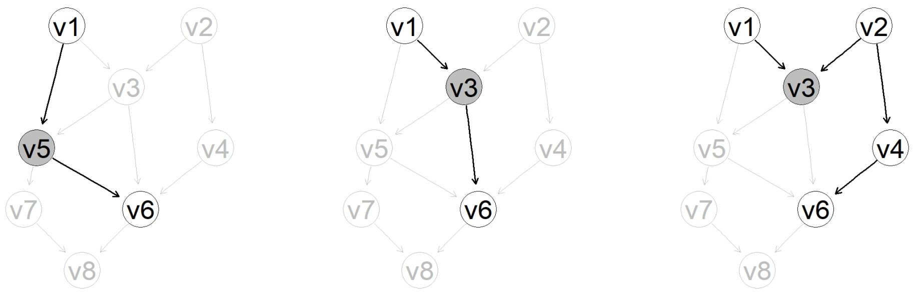
We denote

$$\mathscr{I}(\mathscr{G}) = \left\{\text{d-sep}_{\mathscr{G}}(\mathbf{V}_I;\mathbf{V}_J|\mathbf{V}_K)\ ,\ \mathbf{V}_I,\mathbf{V}_J,\mathbf{V}_K \in \mathbb{V}\right\}\ ,\ \text{if } \mathscr{G} \text{ is a DAG.} \tag{3.2}$$

We will refer to this set as the set of separations encoded in the DAG $\mathscr{G}$.

Example 3.1 (continued) *Let's examine now whether v_1 and v_6 are d-separated in the directed graph. When we disregard the directions in DAG, we can identify in Figure 3.4 three different trails between v_1 and v_6 (they are presented in figure below). Our objective is to find the minimal subset of nodes that block each one of them.*

1. *$v_1 \rightarrow v_5 \rightarrow v_6$ is blocked by $\mathbf{V} = v_5$, as there is a serial connection.*
2. *$v_1 \rightarrow v_3 \rightarrow v_6$ is blocked by $\mathbf{V} = v_3$, as there is a serial connection.*
3. *$v_1 \rightarrow v_3 \leftarrow v_2 \rightarrow v_4 \rightarrow v_6$ is blocked by $\mathbf{V} = \emptyset$. This follows from the fact that there is a diverging connection in $v_l = v_3$, and $v_3 \notin \emptyset$.*



(a) trail 1 (b) trail 2 (c) trail 3

Figure 3.4: Trails between v_1 and v_6 obtained by disregarding directions in the BN.

Note that the minimal subset of nodes that block all the trails is not the union of subsets blocking each trail separately. We see that the set $\mathbf{V} = \{v_3, v_5\}$ *does not block trail 3, as* v_3 *has a converging connection. Indeed, either* v_2 *or* v_4 *have to be included in* $\mathbf{V}$. *It can be concluded that:*

$$d\text{-}sep_{\mathcal{G}}(v_1; v_6 | v_2, v_3, v_5),$$

$$d\text{-}sep_{\mathcal{G}}(v_1; v_6 | v_3, v_4, v_5).$$

In this case, the sets of nodes separating and d-separating v_1 *and* v_6 *, in the moralised graph and in the DAG respectively, are the same.*

This does not always hold. For instance, we identify in Figure 3.2a four different trails between modes v_3 *and* v_4. *The trail that goes through* v_2 *has a diverging connection. The rest of the trails contain converging connections. As a consequence:*

$$d\text{-}sep_{\mathcal{G}}(v_3; v_4 | v_2).$$

On the other hand, the nodes v_3 *and* v_4 *are connected in the moralised graph, and therefore there is no possible separation between them. As a consequence:*

$$\cancel{sep_{M[\mathcal{G}]}(v_3; v_4 | v_2)}$$

Link between Separation in Undirected and Directed Graphs

Motivated by this previous example, the connection between separation in the moralised graph and d-separation in the DAG is shown

Proposition 3.1 *Let* $\mathcal{G}$ *be a DAG. Then* $\mathcal{I}(M[\mathcal{G}]) \subseteq \mathcal{I}(\mathcal{G})$.

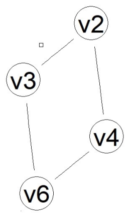
The proof can be found in Koller and Friedman (2009), Proposition 4.8. This proposition implies that all separations encoded in the moralised graph $M[\mathcal{G}]$ will also be present in the DAG $\mathcal{G}$.

3.1.4. Chordal Graphs

Lastly, concepts of chordal and strongly chordal graphs are introduced. These concepts will be important when dealing with PCBNs.

Definition 3.11 (Chordal and Strongly Chordal Graphs) *An undirected graph $\mathcal{G} = (\mathbb{V}, E)$ is said to be chordal if all its cycles of four or more vertices have a chord*[1]*. Moreover, a graph is strongly chordal if it is a chordal graph and every cycle of even length (≥ 6) in $\mathcal{G}$ has an odd chord*[2]*.*

Example 3.1 (continued) *The undirected graph resulting from replacing directed edges by undirected ones (removing directionality) in the DAG in Figure 3.1 is not a chordal graph.*



Indeed, if we remove directionality, there is a cycle of length 4 with no chord.

$$\{v_2, v_3, v_6, v_4\}$$

To make the undirected graph chordal, we have to include the edges: $\{v_3, v_4\}$ or $\{v_2, v_6\}$ and $\{v_6, v_7\}$ or $\{v_5, v_8\}$. When $\{v_3, v_4\}$ and $\{v_6, v_7\}$ are added, the resulting graph is not strongly chordal, as then there is a cycle $\mathbb{C} = \{v_1, v_3, v_4, v_6, v_7, v_5\}$ without odd chord. If e.g. edge $\{v_1, v_4\}$ is added, the obtained graph is strongly chordal.

3.2. Graphical Models

Graphical models are a popular class of dependency modeling techniques (Lauritzen, 1996, Spirtes et al., 2000). They utilize graphs $\mathcal{G}$ to depict the relationships between elements of the random vector $\boldsymbol{X}$ with distribution $\mathcal{P}$.

- Each node $v_i \in \mathbb{V}$ represents a random variable $X_i \in \boldsymbol{X}$.
- Each edge $\{v_i, v_j\} \in E$ represents the direct dependence between X_i and X_j.

We start the presentation with basic probabilistic concepts. They are presented for absolutely continuous random variables rather then in general set up, due to our focus on this case in the thesis.

3.2.1. Probability Concepts

Definition 3.12 (Independence) *Two random variables X_i and X_j are independent if and only if ($\iff$) their joint density is equal to the product of marginal densities*

$$f_{X_i,X_j}(x_i, x_j) = f_{X_i}(x_i) \cdot f_{X_j}(x_j).$$

[1]A chord is an edge that is not part of the cycle but connects two vertices of the cycle.

[2]An odd chord is an edge that connects two vertices that are an odd distance (> 1) apart from each other in the cycle.

In this thesis We will use the following notation when X_i and X_j are independent: $X_i \perp\!\!\!\perp X_j$. Naturally the alternative definition of independence involving the conditional density of X_i given X_j can be used. X_i and X_j are independent if and only if the conditional distribution:

$$f_{X_i|X_j}(x_i|x_j) = f_{X_i}(x_i).$$

Definition 3.13 (Conditional Independence) *Two random variables X_i and X_j are conditionally independent given a random variable X_k $\iff$*

$$f_{X_i,X_j|X_k}(x_i,x_j|x_k) = f_{X_i|X_k}(x_i|x_k)\cdot f_{X_j|X_k}(x_j|x_k)$$

We denote it as $X_i \perp\!\!\!\perp X_j|X_k$. If X_i and X_j are conditionally independent given X_k then:

$$f_{X_i|X_j,X_k} = \frac{f_{X_i,X_j|X_k}(x_i,x_j|x_k)}{f_{X_j|X_k}(x_j|x_k)} = \frac{f_{X_i|X_k}(x_i|x_k)\cdot f_{X_j|X_k}(x_j|x_k)}{f_{X_j|X_k}(x_j|x_k)} = f_{X_i|X_k}(x_i|x_k).$$

3.2.2. Bayesian Networks

In this project, we focus on graphical models called Bayesian Networks (BNs).

Definition 3.14 (Bayesian Networks) *Bayesian Networks $\mathscr{B}$ are a class of graphical models specified by*

- *A Directed Acyclic Graph $\mathscr{G} = (\mathbb{V}, E)$ in which the absence of the arc $v_i \to v_j$, for $v_i < v_j$, represents conditional independence between X_i and X_j given the variables corresponding to parents of X_j.*
- *A set of Probability Density Functions (PDFs)*

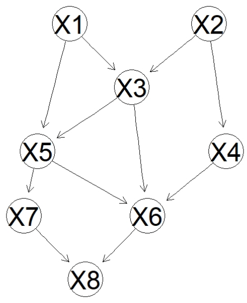
$$f_{X_i|\boldsymbol{X}_{pa(v_i)}}(x_i|\boldsymbol{x}_{pa(v_i)}) \tag{3.3}$$

It is easy to see using standard factorisation of a density and given the conditional independence that follow from the definition above that the density represented by BN is:

$$f_{X_1,\ldots,X_n}(x_1,\ldots,x_n) = \prod_{i=1}^{n} f_{X_i|\mathbf{X}_{pa(v_i)}}(x_i|\boldsymbol{x}_{pa(v_i)}). \tag{3.4}$$

The derivation of (3.4) can be seen in Appendix A.

Example 3.2 (Bayesian Network) *Taking the DAG studied in Example 3.1 and ordering of nodes {1,2,3,4,5,6,7,8}, we get using the definition above that, the quantitative part of the Bayesian Network requires specification of conditional densities given next to the figure of the DAG. These densities are presented below in parametric form where vectors of parameters, that need to be specified, are denoted as $\boldsymbol{\theta_i}$, $i = 1,...,8$.*



The cpds that need to be specified are:

$$\{f_{X_1}(x_1;\boldsymbol{\theta_1}),\ f_{X_2}(x_2;\boldsymbol{\theta_2}),$$
$$f_{X_3|X_1,X_2}(x_3|x_1,x_2;\boldsymbol{\theta_3}),\ f_{X_4|X_2}(x_4|x_2;\boldsymbol{\theta_4}),$$
$$f_{X_5|X_1,X_3}(x_5|x_1,x_3;\boldsymbol{\theta_5}),\ f_{X_6|X_3,X_4,X_5}(x_6|x_3,x_4,x_5;\boldsymbol{\theta_6}),$$
$$f_{X_7|X_5}(x_7|x_5;\boldsymbol{\theta_7}),\ f_{X_8|X_6,X_7}(x_8|x_6,x_7;\boldsymbol{\theta_8})\}$$

Figure 3.5: Example of a Bayesian Network.

The density can then be expressed as:

$$\begin{aligned} f_{X_1,\dots,X_n}(x_1,\dots,x_n) = & f_{X_1}(x_1;\boldsymbol{\theta_1})\cdot f_{X_2}(x_2;\boldsymbol{\theta_2})\cdot f_{X_3|X_1,X_2}(x_3|x_1,x_2;\boldsymbol{\theta_3})\cdot f_{X_4|X_2}(x_4|x_2;\boldsymbol{\theta_4}) \\ & \cdot f_{X_5|X_1,X_3}(x_5|x_1,x_3;\boldsymbol{\theta_5})\cdot f_{X_6|X_3,X_4,X_5}(x_6|x_3,x_4,x_5;\boldsymbol{\theta_6}) \\ & \cdot f_{X_7|X_5}(x_7|x_5;\boldsymbol{\theta_7})\cdot f_{X_8|X_6,X_7}(x_8|x_6,x_7;\boldsymbol{\theta_8}). \end{aligned} \tag{3.5}$$

3.3. CONDITIONAL INDEPENDENCIES IN GRAPHICAL MODELS

There can be more conditional independencies in the distribution represented by BN than the ones given in its definition. The main idea is to show that the concept of separation of nodes in the graph is equivalent to conditional independencies in the distribution corresponding to this graph. The notation introduced by Koller and Friedman (2009) is used in this section. Let us denote:

- The conditioanl independencies encoded in the graph $\mathscr{G}$, denoted as $\mathscr{I}(\mathscr{G})$.
- The conditional independencies that hold in the distribution $\mathscr{P}$, denoted as $\mathscr{I}(\mathscr{P})$. This corresponds to the set:

$$\mathscr{I}(\mathscr{P}) = \{\mathbf{X}_I \perp\!\!\!\perp_{\mathscr{P}} \mathbf{X}_J|\mathbf{X}_K\ ,\ \mathbf{X}_I,\mathbf{X}_J,\mathbf{X}_K \in \boldsymbol{X}\}. \tag{3.6}$$

Theorem 3.2 (d-separation ⇒ Conditional Independence) *Let $\mathscr{P}$ be the probability distribution of the random vector $\boldsymbol{X}$. Let $\mathscr{G}$ be a DAG, such that $\mathscr{P}$ admits a recurring factorisation according to $\mathscr{G}$ of the type (3.4). Then $\mathscr{I}(\mathscr{G}) \subseteq \mathscr{I}(\mathscr{P})$.*

The proof of this theorem can be found either in Koller and Friedman (2009), Subsection 4.5.1.1, or in Lauritzen (1996), Theorem 3.27.

It is vital to underline the following points from this theorem:

1. Any conditional independence that appears in the network is also satisfied in the distribution. That is:

$$\text{d-sep}_{\mathscr{G}}(\mathbf{V}_I;\mathbf{V}_J|\mathbf{V}_K) \Rightarrow \mathbf{X}_I \perp\!\!\!\perp_{\mathscr{P}} \mathbf{X}_J|\mathbf{X}_K. \tag{3.7}$$

2. The reciprocal does not hold. In fact, there may be conditional independencies in the distribution that cannot be read in the network. This will be later illustrated in Example 4.3.

3.3.1. Moralised Graph of Bayesian Networks

We could use a moralized graph corresponding to DAG to read conditional independencies present in the distribution. Indeed, from Proposition 3.1 and Theorem 3.2, it can be deduced that the conditional independencies encoded in the moralised are satisfied in the distribution:

$$\mathrm{sep}_{M[\mathscr{G}]}(\mathbf{V}_I;\mathbf{V}_J|\mathbf{V}_K) \Rightarrow \mathbf{X}_I \perp\!\!\!\perp_{\mathscr{P}} \mathbf{X}_J|\mathbf{X}_K. \tag{3.8}$$

However, there may be more conditional independencies in $\mathscr{G}$, which cannot be read from $M[\mathscr{G}]$. Even so, undirected graphs can provide us with useful information to obtain the structure of directed graphs.

3.4. Properties of Bayesian Networks

There are two fundamental properties of BNs that are examined in this section.

3.4.1. Local Markov Property

Definition 3.15 (Local Markov Property) *Each X_i is conditionally independent of its non descendants (e.g., X_j for which there is no path from v_i to v_j) given its parents.*

$$X_i \perp\!\!\!\perp \boldsymbol{X}_{non\text{-}de(v_i)}|\boldsymbol{X}_{pa(v_i)} \in \mathscr{I}(\mathscr{G}). \tag{3.9}$$

The structure of the Bayesian Network can be then viewed in two very different ways: (Koller and Friedman, 2009):

- First, as a compact representation of a set of conditional independence assumptions about a distribution.
- Second, as a structure that allows to compactly represent a joint distribution in a factorized manner.

3.4.2. Equivalence Classes

Serial and divergent connections result in equivalent factorisations of the distribution. This is illustrated in Figure 3.6.

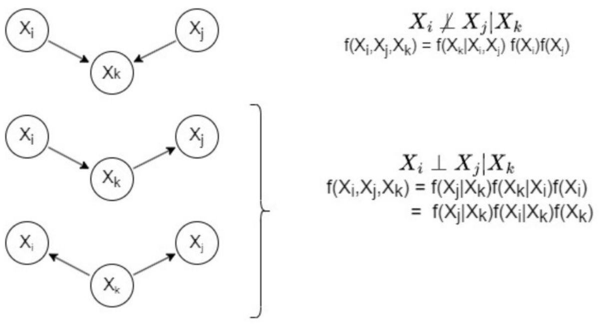


Figure 3.6: Different probability factorisations for serial, converging and diverging connections.

Then $X_i \to X_k \to X_j$ and $X_i \leftarrow X_k \to X_j$ are equivalent. As a consequence, the probability distribution $\mathscr{P}$ can be represented using DAGs with different arc configurations. Such DAGs are said to belong to the same equivalence class. Furthermore, there is no intrinsic property of $\mathscr{P}$ that would allow us to associate it with one graph over an equivalent one.

Definition 3.16 (Equivalence classes) *Two graph structures $\mathscr{G}_1$ and $\mathscr{G}_2$ over $\mathscr{P}$ belong to the same equivalence class $\Leftrightarrow \mathscr{I}(\mathscr{G}_1) = \mathscr{I}(\mathscr{G}_2)$, that is, they encode the same set of conditional independences.*

These equivalence classes are completely characterised by:

- Skeleton: undirected graph obtained as the result of removing directionality from $\mathscr{G}$.
- v-structures: convergent connections of the type $X_i \to X_k \leftarrow X_j$, where there is no arc connecting X_i and X_j.

The following theorem holds.

Theorem 3.3 *Two DAGs $\mathscr{G}_1$ and $\mathscr{G}_2$ over the distribution $\mathscr{P}$ belong to the same equivalence class $\Longleftrightarrow \mathscr{G}_1$ and $\mathscr{G}_2$ have the same skeleton and the same set of v-structures.*

This theorem can be found in Koller and Friedman (2009), Theorem 3.7, and the ideas behind the proof are included in Appendix B.

Equivalence classes are represented by a partially directed graph [3]. Directed edges represent v-structures. Undirected edges represent edges that admit both directions. By directing them we give rise to different graphs belonging to the same equivalence class. We include in Appendix B an example of how to obtain the equivalence class of a DAG.

When estimating structures in later chapters, we will be interested in obtaining the equivalence class of the DAG used to generate the data. The `cpdag` command from the `bnlearn` (Marco Scutari, 2022) package allow us to determine whether two DAGs belong to the same equivalence class. This will facilitate the assessment of structure estimation results.

[3] see Definition 3.1

4

GBNs and PCBNs

The aim of this chapter is to provide an overview of the two types of BNs that will be studied in this dissertation. First, Gaussian Bayesian Networks (GBNs) are presented. Later, Pair Copula Bayesian Networks (PCBNs) are introduced. The qualitative part of both GBNs and PCBNs are the same. They differ with respect to the quantitative part, hence how the conditional probability functions are modelled.

4.1. Types of Bayesian Networks

The PDFs of the form $f_{X_i|\boldsymbol{X}_{pa(v_i)}}(x_i|\boldsymbol{x}_{pa(v_i)})$ must be specified.

- GBNs: The conditional distributions are assumed to be Gaussian with constant variance. This assumption asserts that the joint distribution of the random variables is multivariate Gaussian.
- PCBNs: the conditional distributions can be expressed as the product of bivariate copulas. The study of these distributions is one of the main current challenges in the field of BN research.

4.2. Gaussian Bayesian Networks

Gaussian Bayesian Networks (GBNs) are discussed below. The notation used in this section is that of Koller and Friedman (2009).

Definition 4.1 (Gaussian Bayesian Network) *GBNs are BNs whose variables are continuous, and in which all PDFs are Gaussian with constant variance. That is, if X_{n+1} is a continuous variable with continuous parents $X_1, \dots, X_n$, then there exist some parameters $\beta_0, \beta_1, \dots, \beta_n$ and σ^2 such that the CPDs are of the form:*

$$f_{X_{n+1}|X_1,\dots,X_n}(x_{n+1}|x_1,\dots,x_n) = \mathcal{N}\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n; \sigma^2\right).$$

A significant result is that GBNs can be used to represent the class of multivariate Gaussian distributions and vice versa. Indeed, we show in Appendix C that:

- If $\mathscr{B}$ is a GBN, then it defines a joint distribution that is jointly Gaussian.
- The result of conditioning a multivariate Gaussian, is a Gaussian distribution where the mean is a linear function of the conditioning variables.

From the above we have:

1. Gaussian distributions are completely characterized by its mean vector $\boldsymbol{\mu}$ and its covariance matrix $\boldsymbol{\Sigma}$. This makes the distributions computationally tractable via simple algebra. Exact inference is hence possible.
2. Sampling from Gaussian distribution is computationally cheap and fast.
3. The independencies and conditionally independencies of the Gaussian distribution can be read from the covariance matrix $\boldsymbol{\Sigma}$, and the precision matrix $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$. Indeed:
    - X_i and X_j are independent $\iff \Sigma_{i,j} = 0$.
    - X_i and X_j are conditionally independent given $\boldsymbol{X} \setminus \{X_i, X_j\} \iff \Omega_{i,j} = 0$.
    - X_i and X_j are conditionally independent given $X_{k_1}, \dots, X_{k_n} \iff \tilde{\Sigma}^{-1}_{i,j} = 0$, where $\tilde{\Sigma}$ denote the covariance sub-matrix formed by: $(X_i, X_j, X_{k_1}, \dots, X_{k_n})$.

    The proofs of these properties are shown in Appendix C. These properties also hold for the correlation matrix (result of normalizing the covariance matrix). Moreover, we show that these properties do not hold in general if the distribution is not multivariate Gaussian (see an example in Appendix C).

Example 4.1 (Gaussian Bayesian Network) *Let's study the GBN given by Figure 4.1.*

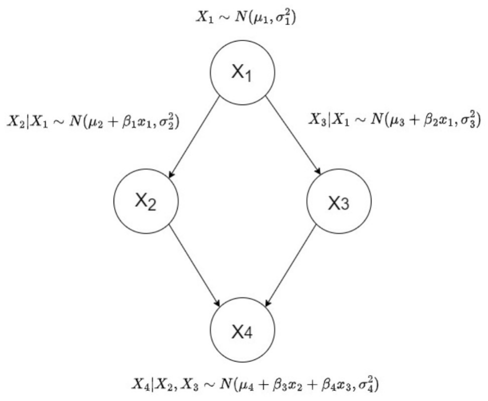

$X_1 \sim N(\mu_1, \sigma_1^2)$
X1
$X_2|X_1 \sim N(\mu_2 + \beta_1 x_1, \sigma_2^2)$
$X_3|X_1 \sim N(\mu_3 + \beta_2 x_1, \sigma_3^2)$
X2
X3
X4
$X_4|X_2, X_3 \sim N(\mu_4 + \beta_3 x_2 + \beta_4 x_3, \sigma_4^2)$

Figure 4.1: Diamond-shape Gaussian Bayesian Network.

1. *Simplicity of computations:*

To compute the distribution of $f_{X_2,X_3}(x_2,x_3)$ we need to carry out the following integral[1]*:*

$$\begin{aligned} f_{X_2,X_3}(x_2,x_3) &= \int_{-\infty}^{\infty} f_{X_1,X_2,X_3}(x_1,x_2,x_3)\cdot dx_1 \\ &= \int_{-\infty}^{\infty} f_{X_1}(x_1)\cdot f_{X_2|X_1}(x_2|x_1)\cdot f_{X_3|X_1}(x_3|x_1)\cdot dx_1. \end{aligned} \tag{4.1}$$

In general, if the distributions are not Gaussian, the above integral does not have a closed form. For the multivariate Gaussian to obtain the marginal distribution it is sufficient to remove the irrelevant variables (the variables to be marginalized) from the mean vector and covariance matrix[2].

Example 4.1 ***(Continued)***

1. *The distribution of (X_2,X_3) is given by:*

$$(X_2,X_3) \sim N\left(\begin{pmatrix}\mu_2+\beta_1\mu_1\\ \mu_3+\beta_2\mu_1\end{pmatrix}, \begin{pmatrix}\sigma_2^2+\beta_1^2\sigma_1^2 & \beta_1\beta_2\sigma_1^2\\ \beta_1\beta_2\sigma_1^2 & \sigma_3^2+\beta_2^2\sigma_1^2\end{pmatrix}\right). \tag{4.2}$$

   The derivation can be examined in Appendix C.

2. *Fast simulations: We use the `bnlearn` package. The parameters have been chosen such that:*

$$\mu_1=\mu_2=\mu_3=\mu_4=\beta_1=\beta_2=\beta_3=\beta_4=\sigma_1^2=\sigma_2^2=\sigma_3^2=\sigma_4^2=1.$$

   Taking $N=2500$, and $set.seed(1)$ for reproducible results, the computation time taken by the `R` software is: Sampling time: 0.01843095 secs This time is relatively small considering the large number of samples. We will compare these times with ones needed in the PCBNs.

3. *Can the conditional independencies be easily detected?: We show in Appendix C that the theoretical covariance matrix $\boldsymbol{\Sigma}$ fulfils:*

$$\Sigma^{-1}_{1,4} = \tilde{\Sigma}^{-1}_{2,3} = 0,$$

   and therefore $X_1 \perp\!\!\!\perp X_4|X_2,X_3, X_2 \perp\!\!\!\perp X_3|X_1$. In practice, these conditions are checked from the empirical covariance and empirical correlation matrices, we have:

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1.0516 & 1.064 & 1.0244 & 2.0772\\ 1.0643 & 2.1027 & 1.0363 & 3.0970\\ 1.0244 & 1.0363 & 1.9666 & 2.9867\\ 2.0772 & 3.0970 & 2.9867 & 6.9532 \end{pmatrix}, \quad \tilde{\boldsymbol{\Sigma}} = \begin{pmatrix} 1.0516 & 1.0643 & 1.0244\\ 1.0643 & 2.1027 & 1.0363\\ 1.0244 & 1.0363 & 1.9666 \end{pmatrix}$$

[1] We use the fact that $X_2 \perp\!\!\!\perp X_3|X_1$ holds in our BN, to factorise the joint density.
[2] https://web.archive.org/web/20100117200722/http://fourier.eng.hmc.edu/e161/lectures/gaussianprocess/node7.html

Computing the inverse:

$$\mathbf{\Sigma}^{-1} = \begin{pmatrix} 2.871 & -0.925 & -0.897 & -0.062 \\ -0.925 & 1.971 & 1.000 & -1.002 \\ -0.897 & 1.000 & 1.986 & -1.020 \\ -0.062 & -1.002 & -1.020 & 1.022 \end{pmatrix}, \quad \tilde{\mathbf{\Sigma}}^{-1} = \begin{pmatrix} 2.867 & -0.987 & -0.959 \\ -0.987 & 0.988 & 1.94 \cdot 10^{-6} \\ -0.959 & 1.94 \cdot 10^{-6} & 0.968 \end{pmatrix}$$

We obtain $\Sigma^{-1}_{1,4} = -0.0259$ and $\tilde{\Sigma}^{-1}_{2,3} = 0.0004$, values really close to 0. Finally, partial correlations can be directly computed from these matrices[3]*. In this case, we obtain:*

$$\begin{pmatrix} 1.000 & 0.389 & 0.375 & 0.036 \\ 0.389 & 1.000 & -0.505 & 0.706 \\ 0.375 & -0.505 & 1.000 & 0.716 \\ 0.036 & 0.706 & 0.716 & 1.000 \end{pmatrix}, \quad \begin{pmatrix} 1.000 & 0.586 & 0.576 \\ 0.586 & 1.000 & -1.98 \cdot 10^{-6} \\ 0.576 & -1.98 \cdot 10^{-6} & 1.000 \end{pmatrix}$$

Therefore $\rho_{X_1,X_4|X_2,X_3} = 0.036$ and $\rho_{X_2,X_3|X_1} = -1.98 \cdot 10^{-6}$. In practice, conditional independencies in GBNs will be checked by testing whether partial correlations are equal to zero.

These three studied properties allow us to efficiently learn the structure of GBNs and make these networks pretty attractive.

However, it is also the case that GBNs are very restrictive and often cannot be used in practice as the Gaussianity assumption is frequently violated.

Example 4.1 (Continued) *The previously simulated Gaussian data can be seen in Figure 4.2.*

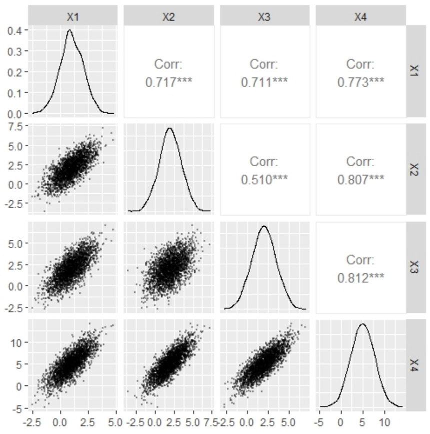


Figure 4.2: Density and Scatter Plots for the different bivariate margins of the Gaussian BN, $N = 2500$.

This type of distribution cannot take into account features such as asymmetries, tail dependencies or non-linear dependencies. Recall that copulas were able to take account for these properties. Highlighting then the limitations of GBNs. limitations.

[3]By just normalising these matrices and taking negative values of the off-diagonal terms.

To evaluate multivariate Gaussian assumptions we will use the so-called Mardia test (Mardia, 1970), which assess skewness and kurtosis of the distribution. This test is implemented in the `MVN` package [4] in `R` (Korkmaz et al., 2014), and will be used during this project.

Example 4.1 (Continued) *Applying the Mardia test[5] to the previous Gaussian data, we obtain:*

```
                Test          Statistic           p value Result
1 Mardia Skewness  15.3319426173024 0.757109152690467    YES
2 Mardia Kurtosis -1.37411557685134 0.169405804326776    YES
3             MVN              <NA>              <NA>    YES
```

For the simulated GBN data, at significance level $\alpha = 0.05$, *we obtain that the test does not reject the null hypothesis that the data follow a multivariate Gaussian distribution, as expected.*

4.3. PAIR COPULA BAYESIAN NETWORKS

Pair Copula Bayesian Networks are introduced in this section.

4.3.1. PCBN CONSTRUCTION

In PCBNs the conditional distributions of variables given their parents in DAG are modelled using copulas and conditional copulas. Let's see then how to decompose PDFs of the form $f_{X_v|\boldsymbol{X}_K}(x_v|\boldsymbol{x}_K)$, in terms of bivariate copulas.

Let $v \in \mathbb{V}$ be a vertex, with parental set $K = \{\text{pa}(v)\}$. Let's assume that $n := |K|$, and K can be written as an ordered subset $K = \{\omega_1, \dots, \omega_n\}$, such that $\omega_i \neq \omega_j$, for $i \neq j$. We define $K_{-i} = \{\omega_{i+1}, \dots, \omega_n\}$ for every $i \in \{1, \dots, n\}$. Using the same reasoning for the PCC in (2.7), it follows that:

$$f_{X_v|\boldsymbol{X}_K}(x_v|\boldsymbol{x}_K) = c_{v,\omega_1|K_{-1}}\left(F_{X_v|\boldsymbol{X}_{K_{-1}}}(x_v|\boldsymbol{x}_{K_{-1}}), F_{X_{\omega_1}|\boldsymbol{X}_{K_{-1}}}(x_{\omega_1}|\boldsymbol{x}_{K_{-1}}); \boldsymbol{x}_{K_{-1}}\right) \cdot f_{X_v|\boldsymbol{X}_{K_{-1}}}(x_v|\boldsymbol{x}_{K_{-1}}).$$

By using the previously defined order and repeating the process recursively, the desired factorisation can be read as:

$$f_{X_v|\boldsymbol{X}_K}(x_v|\boldsymbol{x}_K) = f_{X_v}(x_v)\prod_{i=1}^{n} c_{v,\omega_i|K_{-i}}\left(F_{X_v|\boldsymbol{X}_{K_{-i}}}(x_v|\boldsymbol{x}_{K_{-i}}), F_{X_{\omega_i}|\boldsymbol{X}_{K_{-i}}}(x_{\omega_i}|\boldsymbol{x}_{K_{-i}}); \boldsymbol{x}_{K_{-i}}\right). \quad (4.3)$$

This decomposition is characterized by the order of the parental set. For the n-dimensional case, there are $n!$ possible permutations of the order in K. This give rise to $n!$ different pair copula factorisations.

[4] This package also allow us to generate box plots, Q-Q plots, histograms to assess Gaussian assumptions.
[5] Yes in the MVN column means that the test does not reject the null hypothesis that the data are multivariate Gaussian.

Formal Definition

We then proceed to show a more formal definition. Let $\mathcal{G} = (\mathbb{V}, E)$ be a DAG. Let $\mathscr{P}$ be an absolutely continuous Markovian probability measure [6] on $\mathbb{R}^n$, $n := |\mathbb{V}|$, with strictly increasing univariate marginal CDFs F_{X_i}. Let $<_v$ be a total order in $pa(v)$ for every $v \in \mathbb{V}$, and denote $\mathcal{O} := \{<_v | v \in \mathbb{V}\}$ the set of parental orderings for $\mathcal{G}$. For every $v \in \mathbb{V}$ and $\omega \in pa(v)$, we set:

$$pa(v;\omega) := \{u \in pa(v) | u <_v \omega\} \tag{4.4}$$

Using Sklar's Theorem, first Kurowicka and Cooke (2004), and later Bauer et al. (2012), showed that the joint probability density $f_{X_1,\ldots,X_n}$ can be factorised using the conditional pair copulas $C_{v,\omega|pa(v;\omega)}$, $v \in \mathbb{V}$ and $\omega \in pa(v)$, each of them corresponding to exactly one edge $\omega \to v$ in $\mathcal{G}$. Therefore, the joint distribution can be decomposed as:

$$f_{X_1,\ldots,X_n} = \prod_{v \in \mathbb{V}} f_{X_v}(x_v) \cdot \prod_{\omega \in pa(v)} c_{v,\omega|pa(v;\omega)} \left(F_{X_v|\boldsymbol{X}_{pa(v;\omega)}}(x_v|\boldsymbol{x}_{pa(v;\omega)}), F_{X_\omega|\boldsymbol{X}_{pa(v;\omega)}}(x_\omega|\boldsymbol{x}_{pa(v;\omega)}); \boldsymbol{x}_{pa(v;\omega)} \right). \tag{4.5}$$

Theoretically, every multivariate distribution can be represented using (4.5). In practice, however, as we saw when studying vine models, simplifying assumptions[7] (Hobæk Haff et al., 2010) are assumed. This greatly simplifies the inference process, but leads to the fact that different orders result in different, non-equivalent decompositions. From now on, we work under these assumptions.

Examples

This factorization can be better understood by means of an example.

Example 4.2 (Pair Copula Bayesian Network) *Let's consider the Diamond-shaped BN with uniform margins, given by Figure 4.3a.*

The only node with more than one parent is 4*, that has precisely two parents* $pa(4) = \{2,3\}$*. Hence, there are two possible orders of the variables for the decomposition:*

- *The first order is given by* $3 <_4 2$*, which means* $pa(4;3) = \emptyset$ *and* $pa(4;2) = \{3\}$*. This order is illustrated in Figure 4.3b and the decomposition can be read as:*

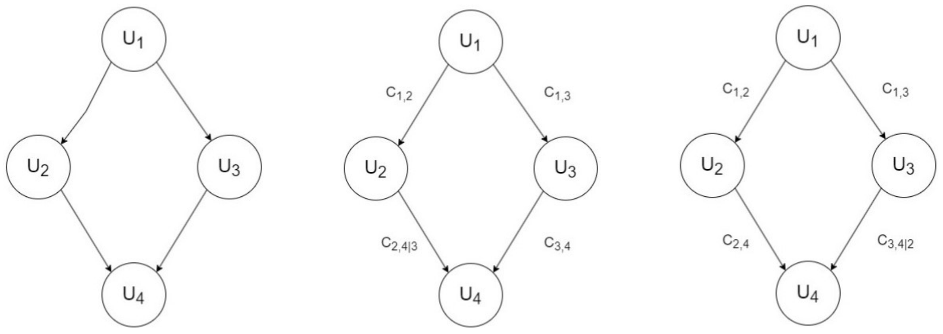
$$f_{U_1,\ldots,U_4}(u_1,\ldots,u_4) = c_{1,2}(u_1,u_2) \cdot c_{1,3}(u_1,u_3) \cdot c_{3,4}(u_3,u_4) \cdot c_{2,4|3}(u_{2|3},u_{4|3}). \tag{4.6}$$

- *The second order is given by* $2 <_4 3$*, which means* $pa(4;2) = \emptyset$ *and* $pa(4;3) = \{2\}$*. This order is illustrated in Figure 4.3c and the decomposition can be read as:*

$$f_{U_1,\ldots,U_4}(u_1,\ldots,u_4) = c_{1,2}(u_1,u_2) \cdot c_{1,3}(u_1,u_3) \cdot c_{2,4}(u_2,u_4) \cdot c_{3,4|2}(u_{3|2},u_{4|2}). \tag{4.7}$$

[6]This means that the probability measure has the local Markov property defined in Definition 3.15

[7]It is then not possible to represent all multivariate distributions by a pair-copula decomposition of the simplified form. Any distribution might however be approximated by a simplified PCC.



(a) Diamond-shape BN, U-margins (b) Order 1. (c) Order 2.

Figure 4.3: Diamond-shape BN uniform margins and its possible orders

Let's analyse (4.6) *in detail* [8]. *To evaluate the conditional copula* $c_{2,4|3}$*, the conditional data* $u_{2|3}$ *and* $u_{4|3}$ *must be computed. The latter can be easily obtained using the copula* $C_{3,4}$ *specified during the decomposition and* (2.13)*, such that:*

$$u_{4|3} = \frac{\partial C_{3,4}(u_3, u_4)}{\partial u_3} = C_{3|4}(u_3|u_4).$$

However, to obtain the value $u_{2|3}$*, we need to know the value of* $C_{2|3}$*. The copula* $C_{2,3}$ *is not specified in the decomposition given by* (4.6)*. Therefore, integrations are necessary to compute* $C_{2|3}$*. Exploiting the conditional independence* $U_2 \perp\!\!\!\perp U_3 | U_1$ *readable in the previous DAG, we get:*

$$u_{2|3} = C_{2|3}(u_2|u_3) = \int_0^{u_2} c_{2,3}(\omega_2, u_3) \cdot d\omega_2 = \int_0^1 \int_0^{u_2} c_{1,2,3}(\omega_1, \omega_2, u_3) \cdot d\omega_1 \cdot d\omega_2 =$$
$$\int_0^1 \int_0^{u_2} c_{1,2}(\omega_1, \omega_2) \cdot c_{1,3}(\omega_1, u_3) \cdot d\omega_1 \cdot d\omega_2 = \int_0^1 \left(\int_0^{u_2} c_{1,2}(\omega_1, \omega_2) \cdot d\omega_2 \right) \cdot c_{1,3}(\omega_1, u_3) \cdot d\omega_1 =$$
$$\int_0^1 \frac{\partial C_{1,2}(\omega_1, u_2)}{\partial \omega_1} \cdot c_{1,3}(\omega_1, u_3) \cdot d\omega_1. \tag{4.8}$$

If we have $m = 1, \ldots, N$ *different samples, the logLikelihood function becomes:*

$$logLik(\boldsymbol{u}, \boldsymbol{\theta}) = \sum_{m=1}^{N} log\left(c_{1,2}(u_1^m, u_2^m)\right) + log\left(c_{1,3}(u_1^m, u_3^m)\right) + log\left(c_{3,4}(u_3^m, u_4^m)\right) +$$
$$log\left(c_{2,4|3}\left(\int_0^1 \frac{\partial C_{1,2}(\omega_1, u_2^m)}{\partial \omega_1} \cdot c_{1,3}(\omega_1, u_3^m) \cdot d\omega_1, C_{4|3}(u_4^m | u_3^m)\right)\right). \tag{4.9}$$

The previous integral has no general closed-form solution. Hence, Monte Carlo methods (Hammersley, 1964) must be used for its evaluation.On the other hand, we saw in Section 4.2 that for the Gaussian case, the distribution of (X_2, X_3) was computable via simple algebra, without the need for integrations.

[8]The case of (4.7) is analogous.

Bauer and Czado (2016) proposed a general algorithm for evaluating conditional PDFs in PCBNs. For high-dimensional cases, the integrals involved in computing the PDFs are usually multidimensional integrals. The main problem arises from the long computational time required to determine these integrals.

This subsection brings to light the aspects of PCBNs.

- On the one hand, we have seen how probability densities can be factorised using bivariate copulas as in (4.5). This fact, together with the good properties of the copulas studied in Section 2.1 shows the great flexibility offered by these PCBNs.
- On the other hand, we have seen in this example that integrations with no-closed solutions are usually present on PCBNs. Its computation implies an immense computational cost and is therefore one of the main disadvantages of PCBNs.

In the case where all copulas involved in the decomposition are Gaussian, the integrals can be avoided by using the covariance matrix (A. M. Hanea et al., 2006) and the partial correlation vines (Kurowicka and Cooke, 2003). However, the use of Gaussian families alone does not result in flexible models.

4.3.2. PCBNs vs Vines

PCBNs theory is closely connected to that of regular vines and has benefited from developments in the latter. In addition, it also provides an alternative way of modeling dependencies, different from undirected graphical models. This subsection discusses the differences and similarities between PCBNs and Vines. These will be illustrated via examples. The work of A. M. Hanea (2011), shows a more general and detailed overview of this topic

Differences

PCBNs are directed graphical models, whereas Vines are undirected graphical models. Perhaps the most important difference between directed and undirected graphs, in general, is that they make different statements of conditional independence. The lack of an edge between two nodes in both graph structures means that the dependency structure between these two variables is controlled by some other variables. However, these structures have a different nature. The absence of an arc in PCBNs encodes conditional independence statements. In contrast, regular vines can be viewed as fully connected graphs representing conditional dependency statements. One may then wonder which model is better. This question is ill-posed, since each model has its advantages and disadvantages over the other.

- In regular vines the concept of conditional independence is weakened to allow for various forms of conditional dependence (A. M. Hanea, 2011). This can be seen in the example above.

  Example 4.2 (Continued) *The involved copulas and the conditional independence statements readable from Figure 4.3b are respectively:*

$$C_{1,2},\ C_{1,3},\ C_{3,4},\ C_{2,4|3}$$

$$U_2 \perp\!\!\!\perp U_3 | U_1 \ , \ U_1 \perp\!\!\!\perp U_4 | U_2, U_3.$$

Let's try to construct a vine that specifies all these copulas and the conditional independence statements. Figure 4.4 shows the different steps (for 2,3 and 4 variables) in the construction of such a vine.

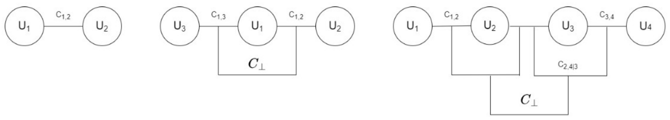


Figure 4.4: Vines D_2, D_3 and D_4 to reconstruct the Diamond shape BN.

The variable orders in the lower tree of the vines in 3 *and* 4 *dimensions do not coincide. We therefore cannot specify both conditional independence statements at the same time with a single vine copula. The same is true for the specification of all copulas.*

This is closely related to the fact that PCBNs enable the specification of conditional copulas, which cannot be computed by simply plugging in results from preceding levels as in (2.13). Instead, integrations are needed for the computation of these copulas, as viewed in (4.8).

- We have seen that in BNs that d-separation implies conditional independence. However, if two nodes are not d-separated it does not imply that the corresponding variables are dependent. Let's illustrate this with an example.

Example 4.3 (Non-d-separation $\not\Rightarrow$ conditional dependence) *Let's study the triangle shape BN. This BN can also be expressed in terms of a regular vine, as can be seen in Figure 4.5.*

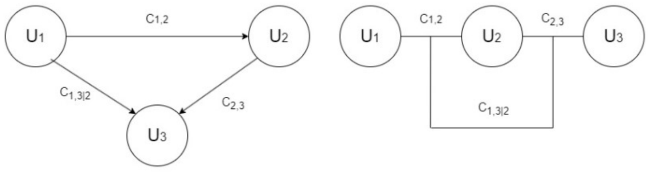


Figure 4.5: Triangle-shape BN and its equivalent Vine representation

There may be independencies that are not encoded in the graph but are present in the distribution. Taking $C_{1,2}$, $C_{2,3}$ and $C_{1,3|2}$ to be Gaussian. Then the copula describing dependence between U_1, U_2 and U_3 is Gaussian, with parameters described in the correlation matrix. The relationship between the partial correlations of these copulas is given by the following equation (Kurowicka and Cooke, 2003)

$$\rho_{1,3|2} = \frac{\rho_{1,3} - \rho_{1,2}\rho_{2,3}}{\sqrt{\left(1-\rho_{1,2}^2\right)\cdot\left(1-\rho_{2,3}^2\right)}}.$$

Choosing $C_{1,3|2}$ such that its partial correlation is:

$$\rho_{1,3|2} = \frac{-\rho_{1,2}\rho_{2,3}}{\sqrt{\left(1-\rho_{1,2}^2\right)\cdot\left(1-\rho_{2,3}^2\right)}},$$

we then get $\rho_{1,3} = 0$. This plus the fact of $C_{1,3}$ is Gaussian, implies that $U_1 \perp\!\!\!\perp U_3$. Thus, U_1 and U_3 are not d-separated and are not dependent.

This example shows how there may be conditional independencies present in our data that are not encoded in the DAG.

We have just seen how if vines fail to represent independencies (see Example 4.2), PCBNs fail to represent dependencies (see Example 4.3). Some statement combinations are better represented using a regular vine, whereas others benefit from DAG representation. The saturated nature of regular vines is a disadvantage in modeling and learning relationships when hundreds of variables are involved. Furthermore, the directed structure of PCBNs provides a more intuitive representation in terms of the flow of influences between variables (A. M. Hanea, 2011).

Similarities

There are situations in which PCBNs and regular vines represent the same set of conditional independence statements and can be characterized by the same bivariate copulas. In the work of Hobæk Haff et al. (2016) and Müller and Czado (2018), the conditions under which high-dimensional DAG models can be linked to regular vines are presented. Motivated by previous work, Zhu and Kurowicka (2022) show that the set of conditional independence encoded in a DAG with strongly chordal skeleton (see Definition 3.11) can be represented by an m-saturated vine (see Definition 2.4). Moreover, one can choose orders in these DAGs, so that the set of bivariate copulas involved in the factorization is the same as in the equivalent m-saturated Vine.

Example 4.4 (Continued) *The vine given by Figure 2.2, together with the copula families taken for sampling ($C_{1,4|2,3} = C_{\perp\!\!\!\perp}$) is an m-saturated vine. Therefore, there is a strongly chordal graph, which represents the same conditional independence statements: $U_1 \perp\!\!\!\perp U_4|U_2,U_3$ as the Vine. This m-saturated vine and its equivalent strongly chordal graph can be seen in Figure 4.6.*

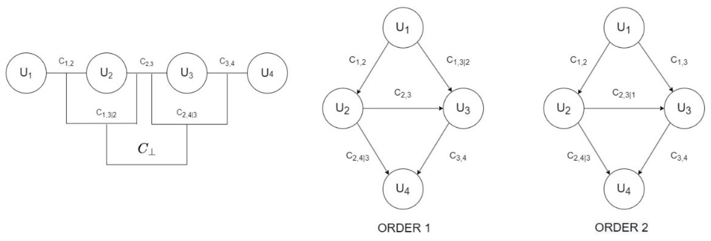


Figure 4.6: m-saturated vine and its equivalent strongly chordal

Furthermore, choosing order 1, the specified copulas are exactly the same. This means that (2.13) *can be used, and hence integrations are not needed. In this case DAG inherits all the properties/procedures studied in Section 2.2. On the other hand, choosing order 2, the copula $C_{2,3}$ is not specified. As a result, integrations are still necessary. Although in this case:*

$$c_{2,3}(u_2,u_3) = \int_0^1 c_{1,2}(\omega_1,u_2)\cdot c_{1,3}(\omega_1,u_3)\cdot c_{2,3|1}(u_2,u_3)\cdot d\omega_1.$$

The avoidance of integral is a powerful result, as it greatly simplifies the computations. The only problem is that strongly chordal assumptions are restrictive and not every graph fulfils them.

Finally, the nature of both methods of factorising the density function as the product of bivariate copulas is the same. As a consequence the sampling procedure for PCBNs uses the one for regular vines. This will be illustrated in Subsection 4.3.3. In addition, the idea behind learning the DAG of an PCBNs together with its parameters from data coincides with the one for learning regular vines (A. M. Hanea, 2011).

4.3.3. PCBN Sampling

The principles behind sampling for PCBNs are the same as that of vines, illustrated in Subsection 2.2.2. The main differences are that PCBNs tend to specify more independent copulas and that integrations are usually required. Let's look at an example to clarify this procedure. The command `cuhre` from the `cubature` package (Narasimhan et al., 2022) in `R` is used to perform the integrations.

Example 4.3 (Continued) *Let's sample from the PCBN specified by Figure 4.3b.*

Sampling Procedure

The sampling procedure (Kurowicka and Cooke, 2006b) is then based on the fact that if $v_1, v_2, v_3, v_4 \sim U(0,1)$ are independent, (u_1,u_2,u_3,u_4) fulfilling:

$$\begin{cases} u_1 = v_1, \\ u_2 = C^{-1}_{2|1;u_1}(v_2), \\ u_3 = C^{-1}_{3|1;u_1}(v_3), \\ u_4 = C^{-1}_{4|3;u_3}\left(C^{-1}_{4|2,3;\int_0^1 \frac{\partial C_{1,2}(\omega_1,u_2)}{\partial \omega_1}\cdot c_{1,3}(\omega_1,u_3)\cdot d\omega_1}(v_4)\right). \end{cases} \tag{4.10}$$

are distributed as the diamond-shape PCBN. Moreover, (4.10) *differs from* (2.15) *in:*

- *$C_{2,3|1} = C_{\perp\!\!\!\perp}$ and therefore the expression to obtain u_3 is simplified.*
- *The copula $C_{2|3}$ need to be computed through integrations and therefore the expression to obtain u_4 becomes significantly more complicated.*

Simulation Setup

The copula families and parameters chosen for sampling are presented in Table 4.1.
The families have been chosen to obtain a dependence structure with tail dependencies and asymmetries, far away from the Gaussian case. The parameters have been chosen to

Table 4.1: Families and parameters of the copulas chosen for the diamond-shaped PCBN.

	Family	Kendall's τ
$C_{1,2}$	Clayton	0.5
$C_{1,3}$	Gumbell	0.75
$C_{3,4}$	Frank	0.5
$C_{2,4\|3}$	Joe	0.75

have highly/medium correlated copulas so that they can be visually identified by scatter plots. In addition, $N = 2500$ is chosen as a sample size large enough to be representative and not too computationally expensive. Finally, we $\texttt{set.seed}(123)$ for reproducibility of our results.

The computation time taken by the R *software for $N = 2500$ is:*

$$\boxed{\textit{Sampling time: 103.85 secs}}$$

This time is more than 1000 times larger than the times taken in the vine in Example 2.1. This reveals how computationally expensive it is to perform integrations.

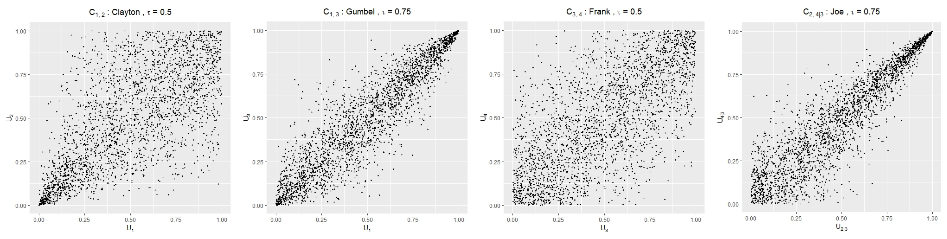
Plugging the true parameters specified in Table 4.1 into (4.9), and evaluating the obtained samples, we get:

$$\boxed{True\ logLik = 7029.19,\quad Comp.time = 93.76 secs}$$

Even the evaluation of the logLikelihood function with true parameters in the simplest non-strongly chordal graph is computationally very expensive.

VISUAL ANALYSIS

After carrying out the sampling procedure explained in (4.10), we present scatter plots to make sure that we are simulating correctly. These plots can be examined in Figure 4.7:



(a) U_1 vs U_2. (b) U_1 vs U_3. (c) U_3 vs U_4. (d) $U_{2|3}$ vs $U_{4|3}$.

Figure 4.7: Scatter plots of the bivariate margins specified by the copulas in Table 4.1

The first three subfigures were obtained by simply plotting the corresponding non-conditional margins. For the fourth one the pseudo-observations $u_{2|3}$ were computed using (4.8) and $u_{4|3}$ using differentiation of $C_{3,4}$.

We visually verify that the simulation results correspond to the expected results of the copulas and parameters selection. In addition, the plots of the remaining bivariate margins are included in Figure 4.8 to gain a broader understanding of the dependency structure.

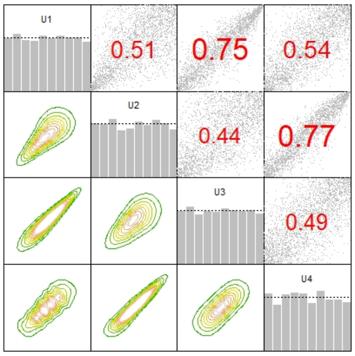


Figure 4.8: Bivariate contour plots, scatter plots, correlations and histograms of our data simulated from the Diamond-shape PCBN.

4.3.4. PCBN Estimation

The PCBN estimation procedure, as in the case of vines, follows the IFM method explained in Subsection 2.2.3. There are, however, two fundamental differences:

- In the case of DAG, not only do we have to choose the structure, but we must also choose the order of the variables. Different orders give rise to different results. This is mainly due to the limited number of copulas we consider and the simplified assumptions.
- If the graph is not strongly chordal, integrations will be necessary to compute the conditional margins and fit the corresponding conditional copulas.

The problem of selecting the DAG and the order will be analysed later. Haff (2013) explores different methods used for parameter estimation. We only show the ML estimation method tree by tree (Aas et al., 2009). That is having the DAG $\mathscr{G}$, the order $\mathscr{O}$ and the pseudo observations $(u_1^m,\dots,u_n^m)$ the ML method finds the parameters $\boldsymbol{\theta}$ which maximise:

$$logLik(\boldsymbol{u},\boldsymbol{\theta}) = \sum_{m=1}^{N}\sum_{v\in\mathbb{V}}\sum_{\omega\in pa(v)} log\left(c_{v,\omega|pa(v;\omega)}\left(C_{v|pa(v;\omega)}(u_v^m|\boldsymbol{u}_{pa(v;\omega)}^m), C_{\omega|pa(v;\omega)}(u_\omega^m|\boldsymbol{u}_{pa(v;\omega)}^m)\right)\right) \tag{4.11}$$

For more insights, we refer to Bauer et al. (2012), who carried out a simulation study to investigate the tractability of likelihood inference in PCBN models. Bauer and Czado (2016) and Bauer and Czado (2016) also deal with ML estimation.

Example 4.3 (Continued) *Let's use the ML method to estimate parameters, copula families and to compute the logLikelihood for the true DAG using the true order. The obtained results are:*

$$\boxed{\textit{Estimated logLik, True DAG, True order} = 7029.76, \quad \textit{Comp.time} = 113.83\,secs}$$

```
> C12.estim
Bivariate copula: Clayton (par = 1.95, tau = 0.49)
> C13.estim
Bivariate copula: Gumbel (par = 4.07, tau = 0.75)
> C34.estim
Bivariate copula: Frank (par = 5.66, tau = 0.5)
> C24.3.estim
Bivariate copula: Joe (par = 6.88, tau = 0.75)
```

We can observe how the ML techniques are able to obtain the correct families and almost all the parameters as well. Therefore, the estimated logLik is practically the same as the real one, and the computation time is similar as well. It can be seen how the estimation process for non-strongly chordal PCBNs is computationally expensive with respect to that of the vines in Subsection 2.2.3.

4.3.5. PCBN Structure Selection

The problem of selecting the structure in PCBN lies in between the problems of:

- Learning the structure in BNs, will be studied in later chapters.
- Learning the structure in Vines, already studied in Subsection 2.2.4.

It involves two steps.

1. Estimation of the DAG $\mathcal{G}$.
2. Selection of the set $\mathcal{O}$ of parent orderings. Remember that the number of possible parent ordering was $\prod_{v\in\mathbb{V}} pa(v)!$. Theoretically, given a DAG $\mathcal{G}$ any order can be used to decompose the density. In practice, however, because of the limited number of parametric bivariate copula families, the simplifying assumption and the tree by tree estimation procedure, the performance of different orders vary.

Example 4.3 (Continued) *Let's illustrate this last statement. Previously we simulated data from Figure 4.3b and we obtained that the estimated logLik using the true order* $3 <_4 2$ *was given by:*

$$\boxed{\textit{Estimated logLik, True DAG, True order} = 7029.76, \quad \textit{Comp.time} = 113.83\textit{secs.}}$$

Let's now estimate the logLikelihood for these data using the order specified by Figure 4.3c, that is the wrong order $2 <_4 3$*. In this case the logLikelihood can be read as:*

$$\begin{aligned} logLik(\boldsymbol{u},\boldsymbol{\theta}) = \sum_{m=1}^{N} & log\left(c_{1,2}(u_1^m, u_2^m)\right) + log\left(c_{1,3}(u_1^m, u_3^m)\right) + log\left(c_{2,4}(u_2^m, u_4^m)\right) + \\ & log\left(c_{3,4|2}\left(\left(\int_0^1 \frac{\partial C_{1,3}(\omega_1, u_3^m)}{\partial \omega_1} \cdot c_{1,2}(\omega_1, u_2^m) \cdot d\omega_1, C_{4|2}(u_4^m | u_2^m)\right)\right)\right) \end{aligned} \tag{4.12}$$

Using the procedure explained in Subsection 4.3.4, we obtain that the estimated logLikelihood of the true DAG using the wrong order is given by:

$$\boxed{\textit{Estimated logLik, True DAG, Wrong order} = 6334.587, \quad \textit{Comp.time} = 89.74\textit{secs.}}$$

Even knowing the structure, the simple fact of choosing an incorrect order results in a relative error of 9.88%.

These are the main reasons why the problem of PCBN structure selection from data is NP-hard with no solution yet found.

This section serves as an introduction to one of the most important topics that will be addressed in this dissertation. Methods attempting to find structures of PCBNs will be discussed in the next chapters.

5

Structure Learning in BNs

"It is impossible to be a mathematician without being a poet in soul."

Sofia Kovalevskaya

In this chapter we study the problem of learning the structure and the parameters of BNs. In particular Structure Learning is the main focus of this dissertation.

We give an overview of the different Structure Learning algorithms available in the literature. Moreover, we introduce the measures that will be used in this thesis to assess the performance of studied algorithms.

5.1. Learning Bayesian Networks

Model Selection and Estimation are collectively known as learning, and are usually performed as a two-step process:

- Structure Learning: learning the graph structure from the data.
- Parameters Learning: learning the local distributions implied by the graph structure learned in the previous step.

In fact learning is a two-step process:

$$P(\boldsymbol{\Theta}, \mathscr{G} | \mathscr{D}) = \underbrace{P(\mathscr{G}|\mathscr{D})}_{\text{Structure Learning}} \cdot \underbrace{P(\boldsymbol{\Theta}|\mathscr{G}, \mathscr{D})}_{\text{Parameter Learning}},$$

where $\mathscr{D}$ is the data.

5.1.1. LEARNING METRICS

Te evaluate performance of different methods in learning the structure of BNs from data the Structural Hamming Distance (SHD) is used in this thesis.

Definition 5.1 (Structural Hamming Distance) *The SHD between two partially directed graphs* [1] *counts the number of edges that need to be added to, removed from, directed in, or reversed to transform one graph to another graph. The SHD comprises:*

- *False Positive edges (FP): edges that have to be removed from, directed in, or reversed from the estimated graph.*
- *False Negative edges (FN): edges that have to be added to the estimated graph.*

$$SHD = FP + FN \tag{5.1}$$

This distance is implemented in the `bnlearn` package by means of the command `shd`. Furthermore, a visual comparison of the graphs can be carried out using the command `graphviz.compare` of the `bnlearn` package. This command displays the original graph and next to it the estimated graph including:

- True positive edges in **black**.
- False positive edges in red.
- False negative edges in blue.

In addition to SHD, Hamming distance will also be used as an evaluation metric.

Definition 5.2 (Hamming Distance) *The Hamming distance between two graphs counts the number of distinct edges between the two networks' skeletons.*

This measure provides complementary information to that obtained from the SHD. Indeed, it is useful to assess whether the algorithms can recover the true directions of the arcs in the BN. It is also used to compare moralised graph of BNs.

Finally, we will also compare the number of edges between the true and the estimated network. This measure will give us information about whether overestimation or underestimation is occurring.

5.2. STRUCTURE LEARNING IN BAYESIAN NETWORK

This section presents the theory behind the different types of structure learning algorithms. First, the Graphical Lasso is studied. Second, the Constraint-Based and the Score-Based algorithms are discussed. For each of these 3 algorithms we show the good properties of the Gaussian case, and the problems that arise in when these algorithms are used when data is generated from PCBNs.

5.2.1. GRAPHICAL LASSO

First, the Graphical Lasso is discussed. This algorithm results in an undirected graph, which corresponds to the moralized graph $M[\mathcal{G}]$ of the BN.

[1] Remember that the equivalence classes were defined by this type of graphs

Graphical Lasso

It can be deduced from Definition 3.8 that if there is no edge in $M[\mathscr{G}]$ between nodes v_i and v_j, the remaining set of nodes separate them. We saw that all the conditional independences encoded in $M[\mathscr{G}]$, also hold in the distribution $\mathscr{P}$. As a consequence, the variables X_i and X_j would be conditionally independent given the remaining variables. We saw in Appendix C, that for the Gaussian distribution it holds that:

$$X_i \perp\!\!\!\perp X_j | X_1,\dots,X_n \backslash \{X_i, X_j\} \iff \Omega_{i,j} = 0, \tag{5.2}$$

where $\boldsymbol{\Omega}$ is the precision matrix. Therefore, estimating the sparsity pattern of $\boldsymbol{\Omega}$ is equivalent to determining which edges are missing in $M[\mathscr{G}]$. This is precisely the reasoning behind the Graphical Lasso (Friedman et al., 2008) to obtain the graph structure.

There are different methods to estimate the sparsity pattern of $\boldsymbol{\Omega}$, but most of them rely on the use of the L^1 penalty (Tibshirani, 1996). In this project, we only focus on the method developed by Meinshausen and Bühlmann (2006), using the implementations carried out by Giraud et al. (2012). This method fits a regression for each of the variables, using the remaining variables as predictors and imposing a L^1 penalty:

$$\begin{cases} \min(X_1 - \theta_{1,2}X_2 - \cdots - \theta_{1,n}X_n)^2 + \lambda\left(|\theta_{1,2}| + \cdots + |\theta_{1,n}|\right) \\ \quad\vdots \\ \min(X_n - \theta_{n,1}X_1 - \cdots - \theta_{n,n-1}X_{n-1})^2 + \lambda\left(|\theta_{n,1}| + \cdots + |\theta_{n,n-1}|\right) \end{cases} \tag{5.3}$$

More rigorously, the problem can be expressed as finding the matrix $\hat{\boldsymbol{\theta}}^{\boldsymbol{\lambda}}$ for a given regularisation parameter λ, such that:

$$\hat{\boldsymbol{\theta}}^{\boldsymbol{\lambda}} = \operatorname{argmin}\{\|\boldsymbol{X}_{\mathscr{D}} - \boldsymbol{X}_{\mathscr{D}}\boldsymbol{\theta}'\|^2_{N\times n} + \lambda\|\boldsymbol{\theta}'\|_1 : \theta' \in \boldsymbol{\Theta}\}, \tag{5.4}$$

where $\boldsymbol{X}_{\mathscr{D}}$ is the $N \times n$ data matrix [2], $\boldsymbol{\Theta}$ is the set of $n \times n$ matrices with 0 on the diagonal and $\|\boldsymbol{\theta}'\|_1 = \sum_{i\neq j}\theta_{i,j}$. The procedure then starts with the fully connected undirected graph. Once the solution $\hat{\theta}^{\lambda}$ is found, the edges of the type $v_i - v_j$ are removed, if either the estimated coefficient $\theta_{i,j}$ or $\theta_{j,i}$ are zero. This results in the desired moralised graph $M[\mathscr{G}_\lambda]$.

The performance of this method heavily depends on the regularisation parameter λ. Letting $\lambda \to 0$, the moralised graph $M[\mathscr{G}_\lambda]$ will become more and more dense. For $\lambda \to \infty$, the moralised graph $M[\mathscr{G}_\lambda]$ will become more and more sparse. This parameter's optimal value is frequently unknown. To cope with this issue, many authors propose to apply cross-validation or the BIC criterion. The `GGMselect` package instead uses the criterion [3] proposed by Giraud (2008) to solve this problem [4].

Examples and Remarks

The most important remark to be made is that (5.2) only holds for the Gaussian case. Hence, if the model is misspecified, (e.g. when data from PCBNs is used), this algorithm might lead to incorrect results.

[2]Each column corresponds to an independent realization of the random vector $\boldsymbol{X}^T = (X_1,\dots,X_n)$

[3]This criterion offers good statistical accuracy and strong theoretical results.

[4]This criterion is not included here due to the complexity of the formulas, but it can be found in Giraud et al. (2012)

Example 5.1 (Graphical Lasso algorithm for Gaussian data) *Let's apply the Graphical Lasso algorithm to the Gaussian data generated in Example 4.1 and plotted in Figure 4.2. To do so, the `GGMselect` package (Bouvier et al., 2022) in `R` is used. The LA family is chosen in Giraud et al. (2012), to apply the procedure previously explained. The criterion proposed in Giraud (2008) is used to determine λ's optimal value. The resulted undirected graph is showed in Figure 5.1.*

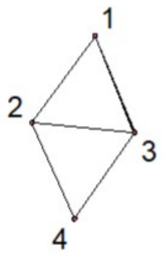


Figure 5.1: Graphical Lasso applied to Gaussian data.

The algorithm is able to recover the true moralised graph of the DAG and the Hamming distance is 0. *It can be read from Figure 5.1 that* $X_1 \perp\!\!\!\perp X_4 | X_2, X_3$. *On the other hand, there is an edge between* X_2 *and* X_3.*This is because* $X_2 \not\perp\!\!\!\perp X_3 | X_1, X_4$, *even though* $X_2 \perp\!\!\!\perp X_3 | X_1$ *holds.*

In the Gaussian setting, this method has been proven to be asymptotically consistent in estimating the set of non-zero elements of $\boldsymbol{\Omega}$ (Friedman et al., 2008). This algorithm can be used as a preliminary step to learn the BN structure. Indeed, edges not presented in the moralized graph, will not be in DAG. These edges can then be blacklisted to give prior extra information to the score and constraint algorithms.

Finally, this algorithm has also been applied to areas outside the Gaussian assumptions, with useful and interesting results (Müller and Czado, 2019, Zhu and Kurowicka, 2022). Therefore, we will study their performance for PCBNs.

5.2.2. Constraint-Based algorithms

Constraint-based algorithms address the problem of learning BN structures as a representation of conditional independencies. They attempt to test for conditional dependencies and independencies in the data and then reconstruct a map that best captures the test results. The outcome of these algorithms is an equivalent class, rather than a particular graph. The most important decisions to be made in this context are the choice of:

Conditional Independence Tests

The methodology used in this project is hypothesis testing. To test whether the variables X_i and X_j are conditionally independent given the set $\boldsymbol{X}_K$, the framework used is:

$$H_0: \; X_i \perp X_j | \boldsymbol{X}_K \quad \text{vs} \quad H_1: \; X_i \not\perp X_j | \boldsymbol{X}_K$$

In this context, we need to define a statistic, to measure the discrepancy with the null hypothesis. Based on this statistic and a significance level $\alpha \in (0,1)$, we define a decision rule $T_\alpha(x_i, x_j; \boldsymbol{x}_K) \in \{H_0, H_1\}$, that will be used to accept or reject the null hypothesis.

- In GBNs, we studied that 0 correlation implies independence. Hence, the so-called 0-correlation independence tests are used [5]. During this project, we use the exact t test for Pearson's correlation coefficient, defined as:

  $$t(X_i, X_j | \boldsymbol{X}_K) = \rho_{X_i, X_j | \boldsymbol{X}_K} \sqrt{\frac{n - |K| - 2}{1 - \rho^2_{X_i, X_j | \boldsymbol{X}_K}}}, \tag{5.5}$$

  where $\rho_{X_i, X_j | \boldsymbol{X}_K}$ is the conditional correlation. This test statistics under null hypothesis is distributed as a Student's t with $n - |K| - 2$ degrees of freedom. The null-hypothesis is rejected when: $|t(X_i, X_j | \boldsymbol{X}_K)| > t_{1-\frac{\alpha}{2}; n - |\boldsymbol{X}_K| - 2}$ [6].

  In this Gaussian case, the conditional and partial correlations are equal. These can be calculated directly using the covariance matrix, so that these tests can be performed very quickly. Moreover, these tests also work when working with Gaussian copulas.

- Outside the Gaussian world the equality of partial and conditional correlations is not valid and zero correlation does not mean independence Another type of test is then needed to test for conditional independence. One could test whether some distance of an empirical copula and independence copula is very small, but these type of tests are computationally expensive. Therefore, it is not recommended to use them to learn the structure of PCBNs.

[5] Tests that use conditional correlations as deviance measures

[6] This threshold value is set according to the probability of false rejection.

CONSTRUCTION PROCEDURES

Several algorithms have been proposed to construct the network using the above tests. In this project we only discuss the PC algorithm (Spirtes et al., 1993), which is schematized Algorithm 1 and Algorithm 2. The application of this algorithm will be illustrated with an example below.

Algorithm 1 PC algorithm: finding the skeleton.

```
**Input** Data set X; significance level $\alpha \in (0,1)$; conditional independence test with decision rule $T_\alpha(x_i, x_j; x_K)$ for the null hypothesis $H_0 : X_i \perp X_j | \boldsymbol{X}_K \;\; v_i \neq v_j \in \mathbb{V}, \boldsymbol{V}_K \subseteq \mathbb{V} \backslash \{v_i, v_j\}$.
**Output** Skeleton $\mathcal{G} = (\mathbb{V}, E)$, separation sets $S_{v_i,v_j}$, $v_i \neq v_j \in \mathbb{V}$, $(v_i, v_j) \neq E$, $(v_j, v_i) \neq E$.
 1: $\mathcal{G} \leftarrow$ fully connected undirected graph on $\mathbb{V}$;
 2: $k \leftarrow 0$;
 3: **repeat**
 4:     **for** $v_i \in \mathbb{V}$ and $v_j \in ad(v_i)$ do $v_i$ and $v_j$ are adjacent in $\mathcal{G}$ **do**
 5:         **if** if $T_\alpha(x_i, x_j; x_K) = H_0$ for any $\boldsymbol{V}_K \subseteq ad(v_i) \backslash \{v_j\}$ with $|\boldsymbol{V}_K| = k$ **then**
 6:             delete $v_i - v_j$ from $\mathcal{G}$
 7:             $S_{v_i,v_j} \leftarrow \boldsymbol{V}_K$;
 8:             $S_{v_j,v_i} \leftarrow \boldsymbol{V}_K$;
 9:         **end if**
10:     **end for**
11:     $k \leftarrow k + 1$.
12: **until** $|ad(v_i)| \leq k$ for all $v_i \in \mathbb{V}$.
```

Algorithm 2 PC algorithm: introducing edge directions.

```
**Input** Skeleton $\mathcal{G} = (\mathbb{V}, E)$, separation sets $S_{v_i,v_j}$, $v_i \neq v_j \in \mathbb{V}$, $(v_i, v_j) \neq E$, $(v_j, v_i) \neq E$.
**Output** Partially directed graph representing the equivalence class.
 1: % Introduce the v-structures
 2: **for** $v_i \in \mathbb{V}$ and $v_j \notin ad(v_i)$ and $v_k \in ad(v_i) \cup ad(v_j)$ **do**
 3:     **if** $v_k \notin S_{v_i,v_j}$ **then**
 4:         replace $v_i - v_k - v_j$ by $v_i \rightarrow v_k \leftarrow v_j$ in $\mathcal{G}$;
 5:     **end if**
 6: **end for**
 7: % Orient as many undirected edges as possible by application of the rules:
 8: **repeat**
 9:     **R1** orient $v_j - v_k$ into $v_j \rightarrow v_k$ whenever $\mathcal{G}$ contains $v_i \rightarrow v_j$ and $v_k \notin ad(v_i)$;
10:     **R2** orient $v_i - v_j$ into $v_i \rightarrow v_j$ whenever $\mathcal{G}$ contains $v_i \rightarrow v_h \rightarrow v_j$;
11:     **R3** orient $v_i - v_j$ into $v_i \rightarrow v_j$ whenever $\mathcal{G}$ contains $v_i - v_j \rightarrow v_j$ and $v_i - v_l \rightarrow v_j$ and $v_l \notin ad(v_k)$;
12: **until** no more edges can be added.
```

The basic idea is to start with a fully connected undirected graph, and removing edges whenever the null hypothesis H_0 is not rejected. The final step is to direct the edges to prevent new v-structures and directed cycles until no more edges require direction. As a result, the equivalence class is obtained rather than a specific network.

EXAMPLES AND REMARKS

If all the conditional independences presented in $\mathscr{P}$ hold in $\mathscr{G}$ and if all statistical test decisions are correct, then the PC algorithm returns the correct equivalence class (Meek, 2013). For the low-dimensional Gaussian case these algorithms seem to work well.

Example 5.2 (PC algorithm for Gaussian data) *Let's apply the PC algorithm to the Gaussian data generated in Example 4.1 and plotted in Figure 4.2. To do so, the command* `pc.stable` *of the* `bnlearn` *package is used. The conditional independence test is the linear correlation test for continuous variables showed in (5.5) with $\alpha = 0.05$. The resulting equivalent class can be seen in Figure 5.2:*

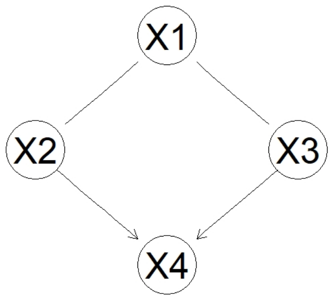


Figure 5.2: PC algorithm applied to Gaussian data.

In this case, the PC algorithm recovers the true equivalent class and hence the SHD and the Hamming distance are 0.

Unfortunately, some problems may arise, especially for high dimensional cases:

1. Constraint-based algorithm assumes that $\mathscr{I}(\mathscr{G}) = \mathscr{I}(\mathscr{P})$ [7]. The main problem is that there might be some conditional independencies in the data to exist that are not represented in the graph [8].

2. Small conditional correlation does not imply independence.

3. Independence tests have always certain probability of false rejection or false acceptance. When several tests are carried out the multiple testing problem might occur. To cope with this multiple testing problem, one may lower the value of $alpha$. This is not implemented in PC algorithm.

Constraint-based algorithms are sensitive to false acceptance and rejections. Hence, it is often better to opt for the score-based algorithms (Scutari et al., 2019).

[7] Recall that is not always true since $\mathscr{I}(\mathscr{G}) \subseteq \mathscr{I}(\mathscr{P})$, see Theorem 3.2.
[8] See for instance Example 4.3

5.2.3. SCORED-BASED ALGORITHMS

Score-based algorithms approach the problem of structure learning as an optimization problem (Koller and Friedman, 2009). The idea is to define a score function that measures how well the model fits the observed data[9]. An initial graph is then modified based on improvements in the score, until a DAG corresponding to a local maximum score is reached (Theodoridis, 2020). This part of the procedure requires specification of a search process through the set of all possible DAGs. Unlike the Constraint-based algorithms where an equivalence class was obtained, the Score-based algorithms output a particular graph. The most important decisions to be made in this context are the choice of:

SCORE FUNCTION

There are different types of score functions, but we will only study likelihood based score functions. Desirable properties of score functions include that:

1. The score functions include a penalisation term. The use of the mere Likelihood function as a score might lead to overfitting. Indeed, adding one extra edge to a network will usually not decrease the maximum Likelihood score (Koller and Friedman, 2009). It is then reasonable to add penalisation terms that favour simpler structures.

2. The score functions are equivalent. This means that the scores associated with all DAGs belonging to an equivalent class are the same.

3. It is desirable that a score function is decomposable, in the sense that the total score is the sum of the scores of all the nodes. Decomposability allows a significant reduction of computational time required during the search of structures. Local changes in the network result in local changes in the score, while most of the score components remain the same, speeding up the procedure.

Example of widely used score functions (Scutari and Denis, 2021) are the AIC-score and the BIC-score:

$$AIC(\mathscr{G},\mathscr{D}) = \sum_{m=1}^{N}\sum_{i=1}^{n}\log\left(f(x_i^m|\boldsymbol{x}^m_{pa(v_i)})\right) - |\boldsymbol{\theta}|, \tag{5.6}$$

$$BIG(\mathscr{G},\mathscr{D}) = \sum_{m=1}^{N}\sum_{i=1}^{n}\log\left(f(x_i^m|\boldsymbol{x}^m_{pa(v_i)})\right) - \frac{|\boldsymbol{\theta}|}{2}\log(N), \tag{5.7}$$

where $|\boldsymbol{\theta}|$ is the total number of parameters and $m = 1,\dots,N$ are the different samples.

- The Gaussian case exhibits very good properties. Indeed, the logLikelihood scores are equivalent. Moreover, the value of the logLikelihood associated with a GBN can be quickly computed using simple algebra.

- The non-Gaussian case is quite different. LogLikelihood scores are not equivalent in this case. Indeed, we saw that even for the same DAG, the resulting scores from taking different parental orderings were different. Lastly, the logLikelihood computation might involve integrals which are extremely computationally expensive. These reasons make, learning structure of PCBNs using score-based algorithms, much more challenging.

[9]The higher the score, the better model we have for our data.

Search Procedure

The search through the set of possible DAG structures is very difficult. The pool of potential candidate structures is immense. Indeed, Robinson (1977) showed that on $n := |\mathbb{V}|$ the number of DAGs p_n is given by the recurrence equation:

$$p_0 = 1, \quad p_n = \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k} 2^{k(n-k)} p_{n-k}. \tag{5.8}$$

Table 5.1: Number of possible DAG structures.

nodes number: n	DAGs number: p_n
1	1
2	3
3	25
4	543
10	4.2×10^{18}

We see that p_n grows super-exponentially in n, hence a systematic trial of all potential DAGs is impossible, necessitating the use of efficient searching techniques. Many heuristic approaches (Koller and Friedman, 2009) have been presented in recent decades to try to identify global maxima in a computationally efficient manner. In this project we only discuss the Hill Climbing algorithm presented in Algorithm 3, which belongs to the so-called Greedy search algorithms.

Algorithm 3 Hill Climbing algorithm: finding the DAG.

```
    Input Score function used and Initial network structure $\mathcal{G}$, usually (but not necessarily) with 0 edges.
    Output Directed Acyclic Graph $\mathcal{G}_{\text{final}}$ representing the BN.
 1: % Compute the score of $\mathcal{G}$, $\text{Score}_{\mathcal{G}} = \text{Score}(\mathcal{G})$;
 2: Set max.score = $\text{Score}_{\mathcal{G}}$.
 3: repeat
 4:     for every possible arc addition, deletion or reversal resulting in a DAG: do
 5:         Compute the score of the modified network $\text{Score}_{\mathcal{G}^*} = \text{Score}(\mathcal{G}^*)$
 6:         if $\text{Score}_{\mathcal{G}^*} > \text{Score}_{\mathcal{G}}$, set $\mathcal{G} = \mathcal{G}^*$ and $\text{Score}_{\mathcal{G}} = \text{Score}_{\mathcal{G}^*}$. then
 7:             update max.score = $\text{Score}_{\mathcal{G}}$.
 8:         end if
 9:     end for
10: until max.score does not increase.
11: $\mathcal{G}_{\text{final}} = \mathcal{G}$.
```

The algorithm starts with an initial graph, and explores the search space by adding, removing, or reversing one arc at a time. The scores function is computed for each proposed graph and the one with the highest score is kept. The procedure is repeated until the score cannot be improved any further. The three operations: adding, removing and reversing posses good properties. The space of graphs is search locally but also explored efficiently at the same time. The algorithm does not guarantee that the best structure will be found and it is important that the good initial structure is chosen. It is also recommended to run the algorithm with random restarts.

Example 5.3 (Hill-Climbing algorithm for Gaussian data) *Let's apply the Hill Climbing algorithm to the Gaussian data generated in Example 4.1 and plotted in Figure 4.2. To do so, the command* `hc` *of the* `bnlearn` *package is used. We choose the BIC-score defined in (5.7) and the empty graph as initial one. The Hill-Climbing algorithm steps with its corresponding scores are displayed then in Figure 5.3:*

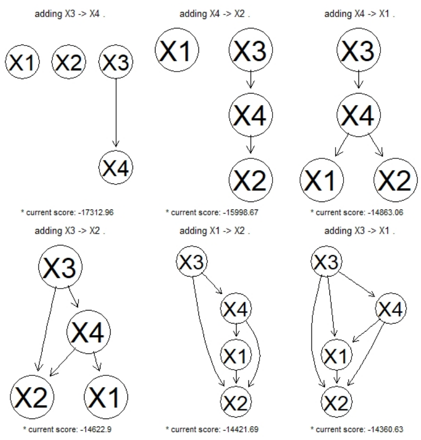

adding X3 -> X4 .
* current score: -17312.96
adding X4 -> X2 .
* current score: -15998.67
adding X4 -> X1 .
* current score: -14863.06
adding X3 -> X2 .
* current score: -14622.9
adding X1 -> X2 .
* current score: -14421.69
adding X3 -> X1 .
* current score: -14360.63

Figure 5.3: Hill Climbing algorithm with BIC-score and no random restarts applied to Gaussian data.

Running the algorithm did not lead to finding the true structure. Hence $l = 20$ *random restarts* [10] *are included. The graphs obtained with their respective scores are shown in Figure 5.4* [11].

The distance between the graphs are given by:

$$\boxed{\textit{No random restarts: } TP = 2,\ SHD = 4:\ FP = 4,\ FN = 0,\ \textit{Ham=1}}$$

$$\boxed{\textit{20 random restarts: } TP = 4,\ SHD = 0:\ FP = 0,\ FN = 0,\ \textit{Ham=0}}$$

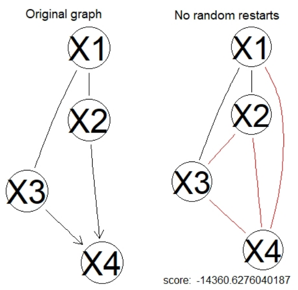
The score for the original network is found to be higher than the score initially obtained.

Finally, the command `score` *of the* `bnlearn` *package is applied to the 3 different DAGs belonging to the same equivalence class to show that the score is equivalent.*

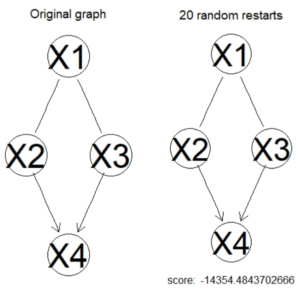
```
> score(model2network("[X1][X2|X1][X3|X1][X4|X2:X3]"),
+        data = gaussian.data, type = "bic-g")
[1] -14354.48
> score(model2network("[X2][X1|X2][X3|X1][X4|X2:X3]"),
+        data = gaussian.data, type = "bic-g")
[1] -14354.48
> score(model2network("[X3][X1|X3][X2|X1][X4|X2:X3]"),
+        data = gaussian.data, type = "bic-g")
[1] -14354.48
```

[10] Initialising the Hill Climbing algorithm using different initial graphs

[11] See Subsection 5.1.1 to interpret the graph.



(a) No random restarts.



(b) $l = 20$ random restarts.

Figure 5.4: Hill Climbing algorithm with BIC-score and different number of random restarts applied to Gaussian data.

This example highlights some of the drawbacks of the Hill Climbing algorithm, such as getting stuck at a local maximum. For this reason, it is really important to choose a favourable DAG as the starting point of the algorithm. In this task, the Graphical Lasso could be of great help. Indeed, if the Graphical Lasso recovers the true moralized graph, this graph could be used as the initial graph, thus incorporating previous knowledge. By simply removing the necessary edges [12], we would recover the original graph.

Finally, we mention that score-based algorithms are more stable than constraint-based algorithms according to Scutari et al. (2019).

[12]The edges that connect common parents in the original network.

6
Gaussian Methods applied to PCBNs data

In chapter Chapter 5 we saw the difficulties of structure learning algorithms for PCBNs data. Hence, we will now explore whether the algorithms for the Gaussian case could be still applied to estimate the structure of PCBNs. The goal of this chapter is thus to assess how Gaussian Learning algorithms work for PCBNS. First, the Graphical Lasso, Constraint and Score-based algorithms are analysed for a particular example. Finally, a larger simulation study is carried out to gain more insights about how appropriate are these learning procedures.

6.1. Overview

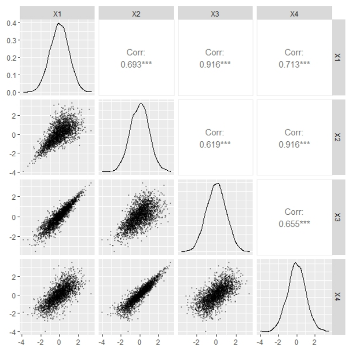
We generated data from PCBNs with standard Gaussian margins. This is not restrictive as margins can always be transformed to desired distributions and this choice of margins 'helps' the tested algorithms. Even if the marginal distributions are Gaussian the joint distribution is far from joint Gaussian. This difference can be explored below.

Example 6.1 (Limitations of Gaussian assumptions) *The data simulated from the Diamond-shape PCBN in Example 4.2 with Gaussian margins and the best fitting joint Gaussian distributions are plotted in Figure 6.1.*

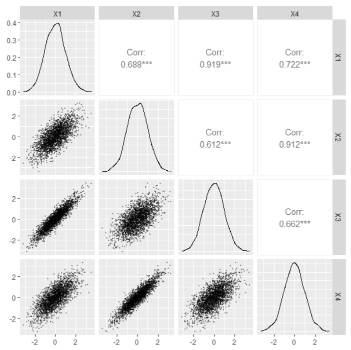
It can be seen at a glance that these scatter plots are very different. Indeed, we can observe lower tail dependency between X_1 and X_2 in the copula data, whereas these dependencies are not present in the multivariate Gaussian. Similarly, X_1 and X_3 exhibit upper tail dependencies in the copula data that are obviously not visible in the multivariate Gaussian.

One can also apply the Mardia's test[1] *to see that the multivariate Gaussian distribution is not a good choice to model the PCBN data.*

[1] See Section 4.2)



(a) Transformed data.



(b) MLE multivariate Gaussian.

Figure 6.1: Bivariate Scatter Plots for PCBN Data with Gaussian margins vs Data simulated from the MLE Multivariate Gaussian.

```
             $multivariateGaussianity
             Test         Statistic p value Result
1 Mardia Skewness  1848.0922252244       0     NO
2 Mardia Kurtosis 35.1589144379238       0     NO
3             MVN             <NA>    <NA>     NO
```

The test rejects the multivariate Gaussianity hypothesis at significance level $\alpha = 0.05$. *So the multivariate Gaussian distribution is not appropriate, but we could still obtain reasonable results when applying methods designed for Gaussian data to find the PCBN structure.*

The rest of the chapter continues as follows:

1. First, Graphical Lasso, Constraint-based and Score-based algorithms are applied to data simulated from the Diamond-shape PCBN with standard Gaussian margins. The main objective is to illustrate the performance of these algorithms

2. Finally, a more in-depth simulation study on the suitability of these Gaussian strucuture learning algorithms for GBN and PCBN data is carried out.

6.2. GRAPHICAL LASSO

First, the Graphical Lasso is analysed. The obtained results, applying this algorithm and using the criterion proposed by Giraud (2008) to determine the optimal value of λ, can be seen in Figure 6.2.

In this case we obtain a fully connected network. However, the moralized graph associated with the true graph does not have the edge $(X_1 - X_4)$. The Graphical Lasso fails to spot that conditional independence in this case. As we said before, the equation (5.2) is no longer valid. As a consequence, this algorithm fails to recover the moralised graph for non-gaussian data.

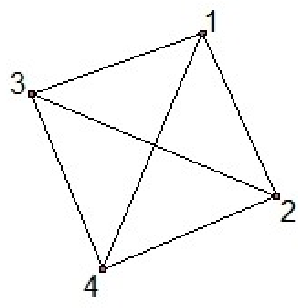
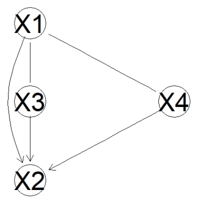


Figure 6.2: Moralised Graph obtained using Graphical Lasso for our PCBN Data with Gaussian margins.

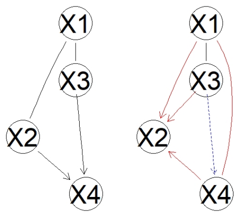
It might be interesting to check the graph obtained for different values of λ [2]. The main problem is that the `GGMselect` package is coded in such a way that the above criteria is implemented directly. Therefore, we can not choose the desired value of λ.

6.3. CONSTRAINT-BASED ALGORITHMS

Next, Constraint-based algorithms are analysed. The obtained results, applying the PC algorithm and using the 0-correlation independence test in (5.5) with $\alpha = 0.05$, can be seen in Figure 6.3.



(a) Equivalence class result of the PC algorithm.



(b) Comparison between the obtained and true equivalence classes using PC algorithm.

Figure 6.3: Gaussian PC algorithm applied to our PCBN Data with Gaussian margins.

The distances between both graphs are given by:

$$\boxed{\text{TP} = 1,\ \text{SHD} = 5:\ \text{FP} = 4,\ \text{FN} = 1,\ \text{Ham} = 3}$$

The results obtained are pretty poor, in a 4 edges network the SHD is 5. Let's try to identify the reasons behind it. The only conditional independence readable from the estimated network is $X_3 \perp\!\!\!\perp X_4 | X_1$. However, this conditional independence in not presented in the Diamond-shape PCBN, nor in the data since:

$$c_{3,4|1}(u_3, u_4|u_1) = \int_0^1 c_{1,2}(u_1,\omega_2)\cdot c_{1,3}(\omega_1, u_3)\cdot c_{3,4}(u_3, u_4)\cdot c_{2,4|3}(C_{2|3}(\omega_2|u_3), C_{4|3}(u_4|u_3)\cdot d\omega_2 \neq 1$$

[2]This example may suggest that by penalizing more, we would obtain a graph with fewer edges, and who knows if we would recover the graph.

On the other hand, these algorithms are not able to detect: $X_2 \perp\!\!\!\perp X_3|X_1$ nor $X_1 \perp\!\!\!\perp X_4|X_2, X_3$, even though these conditional independences are encoded in the Diamond-based PCBN and thus are present in our data. The performed Gaussian conditional independence tests are:

```
> ci.test("X2", "X3", "X1", test = "cor", data = gdatabbn1)
Pearson's Correlation
data:  X2 ~ X3 | X1
cor = -0.055426, df = 2497, p-value = 0.00558
alternative hypothesis: true value is not equal to 0

> ci.test("X1", "X4", c("X2","X3"), test = "cor", data = gdatabbn1)
Pearson's Correlation
data:  X1 ~ X4 | X2 + X3
cor = 0.065404, df = 2496, p-value = 0.001073
alternative hypothesis: true value is not equal to 0

> ci.test("X3", "X4", "X1", test = "cor", data = gdatabbn1)
Pearson's Correlation
data:  X3 ~ X4 | X1
cor = 0.0042779, df = 2497, p-value = 0.8307
alternative hypothesis: true value is not equal to 0
```

We can see that the only test not rejecting the null hypothesis H_0 [3] at significance level $\alpha = 0.05$, is the test: $X_3 \perp\!\!\!\perp X_4|X_1$. The other two tests yield small correlations, but do not have sufficient statistical evidence to reject the null hypothesis.

This example illustrates that the correlation tests for the Gaussian case might not detect the conditional independences presented in our data. Secondly, these tests might not reject the null hypothesis when low correlation is in the data but variables are not independent.

6.4. Score-Based algorithms

Lastly, the Score-based algorithms are analysed. The results obtained, applying the Hill-Climbing algorithm, using BIC score (5.7) and $l = 15$ random restarts, can be seen in Figure 6.4.

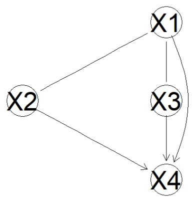
The distances between both graphs are given by:

$$\boxed{\text{TP} = 4,\ \text{SHD} = 1:\ \text{FP} = 1,\ \text{FN} = 0,\ \text{Ham} = 1}$$

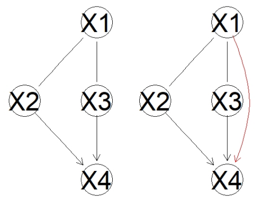
In this case, the results are not so bad. Only the edge $X_1 \to X_4$ was not present in the original network. Thus, Hill Climbing fails to detect $X_1 \perp\!\!\!\perp X_4|X_2, X_3$, but is able to recover $X_2 \perp\!\!\!\perp X_3|X_1$. Let's examine the scores for the estimated graph and the true one.

$$\boxed{\text{BIC}_{\text{Estim.DAG}} = -8629.624,\ \text{BIC}_{\text{True.DAG, estim.param.}} = -8631.07}$$

[3]The test use, H_0: variables are conditionally independent.



(a) Equivalence class result of the Hill Climbing algorithm.



(b) Comparison between the obtained and true equivalence classes using Hill Climbing algorithm.

Figure 6.4: Gaussian Hill Climbing applied to our PCBN Data with Gaussian margins.

The score gives priority to the estimated DAG over the true DAG. The true DAG is hence no longer the global maximum of the score function. We are using a score function based on the Gaussian logLikelihood. However, since our data are not Gaussian, this score function will no longer be valid to represent the likelihood of these data. As a consequence, these algorithms may fail to recover the true DAG. Despite this, the obtained results seem more promising than in the Constraint-based case.

6.5. SIMULATION STUDY

Finally, a further simulation study on the suitability of these procedures for learning the structure of PCBNs is carried out. We are insterested in getting insights that can help us in the process of finding the true structure of PCBNs.

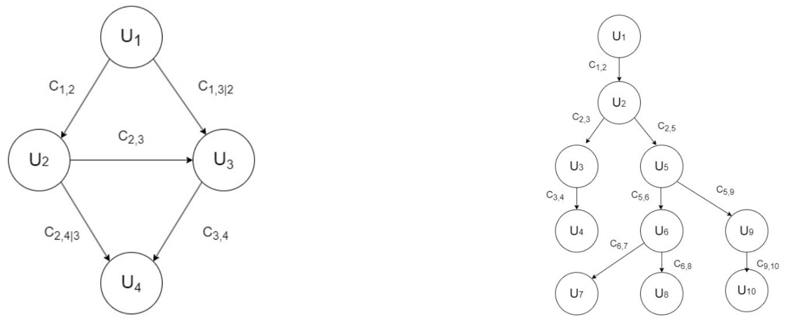
6.5.1. SIMULATION OBJECTIVES

The main objective of this simulation study is to compare the efficiency of the Gaussian structure learning algorithms for GBNs and PCBNs. To do so, we study the following points:

1. Assess whether the Graphical Lasso is able to recover the true moralised graphs.
2. Examine which of the 2 algorithms: PC and Hill Climbing yield better results.
3. Analyze the results obtained for SHD and Hamming distance, to see if our algorithms are able to recover equivalence classes and skeletons.
4. Study the performance of these algorithms for graphs of different nature, i.e. with different number of nodes and edges.
5. Inspect the efficiency of the algorithms with respect to the sample size.
6. Explore the computation time required for each of these algorithms.
7. In case the algorithms do not recover the true structure:
    - whether there is an overestimation or underestimation of the number of edges.
    - are there any patterns with respect to copula families or parameters.

6.5.2. SIMULATION SETUP

We have selected two DAGs of different nature to carry out this simulation study. These DAGs can be seen in Figure 6.5. Both graphs are strongly chordal, so vine copula models are used to sample from them[4].



(a) Small Network. (b) Big Network.

Figure 6.5: Simulation Study: GBNs vs PCBNs.

The idea of this study is, first, to sample from these DAGs using only Gaussian copulas, so that by transforming the margins to Gaussian, GBNs are obtained. Second, to sample from these DAGs choosing other copula families, so that PCBNs are obtained.

SMALL NETWORK

The small network can be seen in Figure 6.5a. It has $n = 4$ nodes and $|E| = 5$ edges, i.e. we are dealing with a low-dimensional but dense network. We simulate from $m = 60$ different scenarios, results of the combination of:

- 10 family configurations.
  - For GBNs, the chosen families are Gaussian copulas.
  - For PCBNs, these configurations are shown in Table 6.1.

Table 6.1: Different Family Configuration for PCBNs, Small Network.

	Family Configuration									
	1	2	3	4	5	6	7	8	9	10
$C_{1,2}$	t	C	G	F	J	t	G	J	t	C
$C_{2,3}$	t	C	G	F	J	t	G	J	C	F
$C_{3,4}$	t	C	G	F	J	t	G	J	G	J
$C_{1,3\|2}$	t	C	G	F	J	C	F	t	F	t
$C_{2,4\|3}$	t	C	G	F	J	C	F	t	J	C

- 6 different parameter configurations [5], shown in Table 6.2.

  The choice of these configurations has been such that we have scenarios where:

[4] See Subsection 2.2.2
[5] Same parameter configuration for GBNs and PCBNs

Table 6.2: Different Parameter Configurations, Small Network.

	Parameter Configuration					
	1	2	3	4	5	6
$\tau_{1,2}$	0.25	0.75	0.75	0.25	0.25	0.75
$\tau_{2,3}$	0.25	0.75	0.75	0.75	0.25	0.75
$\tau_{3,4}$	0.25	0.75	0.25	0.75	0.25	0.75
$\tau_{1,3\|2}$	0.75	0.25	0.75	0.25	0.25	0.75
$\tau_{2,4\|3}$	0.75	0.25	0.25	0.75	0.25	0.75

  - Copulas in 1^{st} tree are low correlated and Copulas in 2^{nd} tree are highly correlated, and vice versa.
  - Mix between highly and low correlated copulas.
  - All copulas are low correlated.
  - All copulas are highly correlated.

Big Network

The larger network can be seen in Figure 6.5b. It has $n = 10$ nodes and $|E| = 9$ edges, i.e. we are dealing with a higher dimensional, but sparse network. We simulate from $m = 64$ different scenarios, results of the combination of:

- 8 family configurations.
  - For GBNs, the chosen families are Gaussian copulas.
  - For PCBNs, these configurations are shown in Table 6.3.

Table 6.3: Different Family Configuration for PCBNs, Big Network.

	Family Configuration							
	1	2	3	4	5	6	7	8
$C_{1,2}$	t	C	G	F	J	t	G	J
$C_{2,3}$	t	C	G	F	J	t	G	J
$C_{3,4}$	t	C	G	F	J	C	F	t
$C_{2,5}$	t	C	G	F	J	C	F	t
$C_{5,6}$	t	C	G	F	J	G	J	C
$C_{6,7}$	t	C	G	F	J	G	J	C
$C_{6,8}$	t	C	G	F	J	F	t	G
$C_{5,9}$	t	C	G	F	J	F	t	G
$C_{9,10}$	t	C	G	F	J	J	C	F

- 8 different parameter configurations [6], shown in Table 6.4.

  The choice of these configurations has been such that we have scenarios where:
  - All copulas are low correlated.
  - All copulas are high correlated.
  - Mix between highly and low correlated copulas.

[6]Same parameter configuration for GBNs and PCBNs

Table 6.4: Different Parameter Configurations, Big Network.

	Parameter Configuration							
	1	2	3	4	5	6	7	8
$\tau_{1,2}$	0.25	0.75	0.25	0.75	0.25	0.75	0.25	0.75
$\tau_{2,3}$	0.25	0.75	0.25	0.75	0.25	0.75	0.75	0.25
$\tau_{3,4}$	0.25	0.75	0.25	0.75	0.75	0.25	0.25	0.75
$\tau_{2,5}$	0.25	0.75	0.25	0.75	0.75	0.25	0.75	0.25
$\tau_{5,6}$	0.25	0.75	0.75	0.25	0.25	0.75	0.25	0.75
$\tau_{6,7}$	0.25	0.75	0.75	0.25	0.25	0.75	0.75	0.25
$\tau_{6,8}$	0.25	0.75	0.75	0.25	0.75	0.25	0.25	0.75
$\tau_{5,9}$	0.25	0.75	0.75	0.25	0.75	0.25	0.75	0.25
$\tau_{9,10}$	0.25	0.75	0.75	0.25	0.75	0.25	0.25	0.75

Additional Remarks

- Since neither of these two networks has a v-structure, the moralized graph and the DAG skeleton will coincide. This enables a direct comparison of the three investigated methods.
- The fact that GBNs and PCBNs have the same parameter configurations makes the scenarios comparable and therefore also their results.

For each of the previous scenarios, $M = 30$ simulations are performed using different seeds [7]. During the whole simulation study we set the seeds to get reproducible results. Moreover, 4 different sample sizes are taken:

$$N = 500,\ N = 2500,\ N = 7500,\ N = 15000.$$

This will allow us to determine how the efficiency of these algorithms varies with respect to the sample size.

6.5.3. Simulation Results

For each of the simulations:

1. the Graphical Lasso, using the criterion proposed by Giraud (2008) to determine the optimal value of λ,
2. the PC algorithm, using the 0-correlation test showed in (5.5) with $\alpha = 0.05$,
3. the Hill Climbing algorithm, using BIC score (5.7) and $l = 15$ random restarts,

are applied to the simulate data. To compare the true and the estimated structures we compute for each of the previous algorithms:

- SHD distance [8].
- Hamming distance.

[7]To ensure that outliers are smoothed out
[8]only for the PC and Hill Climbing algorithm
[9]We also store an adjacency matrix for each of the scenarios, to better understand the edges that are presented.

- Number of edges [9].
- Running times.

The quantities of interest are averaged over the $M = 30$ different simulations. These values are stored for all different scenarios into data frames.

For a better visual analysis of the results, density plots are created for: SHD distances, Hamming distances and number of edges. In addition, the medians of these results are computed and represented with a vertical line in the density plots. We also include the values of the quantiles: $Q_{0.05}$ and $Q_{0.95}$ to have a better quantitative understanding of the spread in the distribution. The computational times for each of the algorithms are stored. The obtained results can be seen in Table 6.5.

Table 6.5: Simulation Results Chapter 6.

	Small Network	**Big Network**
$N = 500$	Table D.1 Figure D.1	Table D.4 Figure D.5
$N = 2500$	Table D.2 Figure D.2	Table D.5 Figure D.6
$N = 7500$	Table D.3 Figure D.3	Table D.6 Figure D.7
$N = 15000$	Table 6.6 Figure D.4	Table 6.7 Figure D.8
Number of edges	Table D.7	Table D.8
Running times	Table D.9	

Notice that we will have in total:

$$\underbrace{2}_{\text{Small and Big}} \times \underbrace{4}_{\text{Different Sample Sizes}} \times \underbrace{2}_{\text{Gaussian and Non-gaussian}} = 16 \text{ Data Frames}$$

$$\underbrace{2}_{\text{Small and Big}} \times \underbrace{4}_{\text{Different Sample Sizes}} \times \underbrace{2}_{\text{Gaussian and Non-gaussian}} \times \underbrace{3}_{\text{SHD, Hamming, edges}} = 48 \text{ figures}$$

For a better follow-up of the discussion of the results, we include Table 6.6 and Table 6.7 corresponding to $N = 15000$.

Once these results have been obtained, we are ready to address the first 6 simulation objectives. Let's analyse in which cases the studied algorithms fail, in order to address point number 7.

Table 6.6: Simulation Results for $N = 15000$, Small Network.

		Gaussian			**Non-Gaussian**		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	0.033	0.000	1.033	1.000	0.000	1.440
Constraint	SHD	1.016	0.031	4.667	1.000	0.067	4.467
	Hamming	0.900	0.031	1.102	1.000	0.067	1.803
Score	SHD	0.000	0.000	1.003	0.833	0.000	3.500
	Hamming	0.000	0.000	1.003	0.833	0.000	1.745

Table 6.7: Simulation Results for $N = 15000$, Big Network.

		Gaussian			**Non-Gaussian**		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	0.000	0.000	0.033	3.733	0.000	20.995
Constraint	SHD	0.467	0.177	1.147	6.633	0.458	16.118
	Hamming	0.267	0.067	0.662	3.933	0.267	15.953
Score	SHD	0.167	0.000	0.390	7.733	0.238	22.958
	Hamming	0.067	0.000	0.167	5.550	0.100	19.500

6.5.4. SIMULATION FAILURE ANALYSIS

The most significant results we have found during the analysis are included below.

SMALL NETWORK

1. **Graphical Lasso**

The performance of Graphical Lasso for the Gaussia case is very efficient, especially the larger the sample size. This algorithm never place the edge $(1,4)$, so overestimation never occurs. Moreover, it is able to recover the true moralised graph in most scenarios. However, it encounters problems with parameter configurations 1 and 6 [10] in Table 6.2. For these cases, all the scenarios fail to recover the edge $(2,3)$.

For the non-Guassian case the performance worsens notably. In fact, for this case, as we increase the sample size, the results are farther away from the real ones. No failure patterns are observed. But it always tends to overestimate the number of edges, for instance including the $(1,4)$ edge in 35 out of the 60 scenarios we study.

2. **PC Algorithm**

The PC algorithm does not perform well for the Gaussian case, compared to the other algorithms. Even for $N = 15000$, it fails to recover the true equivalence class. It mainly fails for:

- Parameter configuration 3 in Table 6.2, characterised by $\tau_{3,4} = 0.25$. It is precisely the edge $(3,4)$ the one that it is not recovered.

[10]These correspond to cases where copulas in the second tree are highly correlated.

- Parameter configuration 4 in Table 6.2, characterised by $\tau_{1,2} = 0.25$. It is precisely the edge $(1,2)$ the one that is not recovered.

These copulas with low correlation are probably the cause of the failure of the tests. On the other hand, the edge $(1,4)$ is almost never included, so the number of edges tends to be underestimated.

For the non-Gaussian case, the results are even poorer. There are no apparent patterns in the failed scenarios, but they always tend to overestimate the number of edges.

3. HC Algorithm

The HC algorithm gives the best results for the Gaussian case. Like the Graphical Lasso, it only fails to recover the true structure for scenarios with highly correlated copulas in the second tree.

The results obtained for the non-Gaussian case are worse than for the Gaussian case. Even so, it is the one that gives the best results of the 3 algorithms for PCBNs. It is the least overestimated and has a median hamming distance that is closest to zero. This algorithm gives good results for the scenarios where the copulas in the second tree have all low correlations.

BIG NETWORK

1. Graphical Lasso

Graphical Lasso for the Gaussian case is the best performing algorithm. For $N = 15000$, it is able to recover the true moralized graph in 58 out of 64 scenarios.

The performance for the non-Guassian case worsens considerably, reaching even 20 wrong edges. In this case, the scenarios furthest away from the true network were those corresponding to configuration families 2 and 5 in Table 6.3. These scenarios correspond to the fully Clayton and fully Joe copula cases. On the other hand, configuration families 1: t copula, gives good results. It was also interesting to find that the fully low correlated case had a lower error than the fully highly correlated case. Finally, this algorithm always overestimates the number of edges, but it recovers the edges of the true network.

2. PC Algorithm

The PC algorithm performs considerably better than it did for the small network. In fact, the median of SHD for the big network is lower in all cases than the small one, even though the number of edges is higher. In the few cases where it fails, it is because it incorporates some extra edges, but it always recovers the true edges.

As expected, the outcomes for the PC algorithm for the non-Gaussian case are much poorer. They get even worse with growing sample size. It is interesting to see how the scenarios in which PC fails coincide exactly with those in which Graphical Lasso failed [11]. As before, these algorithms tend to overestimate the number of edges (although a little less than Graphical Lasso) but always recover the true edges.

[11] Fully Clayton and fully Joe copula scenarios

3. HC Algorithm

As in the case of the small network, the HC algorithm is relatively good for the Gaussian case. Moreover, it always recovers the true edges.

For the non-Gaussian case exactly the same happens as in case of the Graphical Lasso and the PC algorithm. HC also fails mainly when the copula families are Clayton and Joe, and even more so when all copulas have high correlation. It always recovers true edges, but adds even more edges than the previous algorithms. The quantile 95 reaches a value of 28 wrong edges.

6.5.5. SIMULATION DISCUSSION

For this two different network structure we can draw the following conclusions from this simulation study.

The results obtained for GBNs are as expected significantly better than those for PCBNs. Indeed, all the methods we have used work under Gaussian assumption, so when the data is not Gaussian likelihoods and tests are no longer valid .

1. The Graphical Lasso is able to recover the moralized graph for the Gaussian case. In fact for the large network, it is the best performing algorithm. On the other hand, for the non-Gaussian case it tends to overestimate the number of edges, but the true edges are usually recovered.

2. The Hill Climbing algorithm outperforms the PC algorithm for the Gaussian setting in both networks. This is in line with the results discussed in Scutari et al. (2019). For the non-Gaussian case this is no longer true.

3. There are no major differences when SHD or Hamming distance are used.

4. The results are quite different between networks. For example the PC algorithm did not have a good performance for the small and dense network, however for the large network it seems to work much better. This suggests that we should not expect any of these algorithms to have the best performance in all cases.

5. For the Gaussian case we can observe that the efficiency of all algorithms increases with respect to the sample size, as expected. For the non-Gaussian case this is no longer true. For the large network, the larger the sample size, the more overestimation occurs.

6. The computation times are longer for the Graphical Lasso than for the other two algorithms, however all three algorithms are quite fast considering the large sample size. We have tested only 4 and 10 dimensional networks. Higher-dimensional examples should be still considered.

7. Finally, it is interesting to see the results analysed at Subsection 6.5.4. The main conclusion we can draw is that for the large network, non-Gaussian case, the three algorithms overestimate, but they all recover true edges. It might then be a good idea:

(a) Use higher values of λ to induce more sparsity on the moralised graph.

(b) Lower the significance level α in the conditional independence tests. By doing so, we would reject less the null hypothesis. As a consequence, more edges would be removed from the network.

(c) Using scores that penalize more than the BIC, to remove edges.

However, these procedures may also cause true edges not to be recovered.

Since the true edges are always recovered, we can use these Gaussian methods to obtain prior information about the structure of PCBNs.

6.5.6. SIMULATION CONCLUSION

The above results suggest that these methods are not entirely efficient at learning the structure of PCBNs. Even if these algorithms were efficient in recovering the structure, we would still have to choose a parental ordering for the nodes in the PCBNs and then estimate the relevant conditional copulas. So we would run into additional questions and problems. It might be a better idea to tailor algorithms for PCBNs:

- For Constraint-based algorithms we saw many problems arise on how to test independencies for the non-Gaussian case.
- We therefore opt for exploring algorithms with scores based on the logLikelihood of PCBNs. This will be the topic to discuss in the following chapter.

In any case, we can benefit from the procedures discussed in this chapter. Indeed, the three algorithms were able to recover all the true edges. Thus, we could use these results to obtain an initial graph. This graph would be used to initialize a score-based algorithm with a suitable score for the PCBNs.

A more in-depth simulation study is needed to obtain more meaningful results or to support the findings in this simulation study.

7

SCORE-BASED ALGORITHMS FOR PCBNS

Gaussian methods were not completely efficient for learning the structure from PCBN data. Score-based algorithms with PCBN logLikelihood scores are then explored in this chapter. Two main issues arise when dealing with these scores:

1. The logLikelihood computation might involve integration which is computationally expensive [1].

2. The scores computed for different ordering of parents are not the same even if the structure is the same [2].

For these reasons, it is computationally unfeasible to implement search algorithms based on the exact logLikelihood expressions and to check all the possible parental orderings. Instead, we can:

1. Make logLikelihood approximations that avoid integral computations. In this way, the procedure is speed up and computationally practical algorithms can be obtained.

2. Choose some criteria to select the set of parental orderings.

The former point was first addressed by Pircalabelu et al. (2017). They proposed approximating the logLikelihood using quotient of regular vines and using it as a score. This algorithm is analysed in this chapter, showing its shortcoming.

Motivated by these works, another PCBN logLikelihood approximation is proposed. Moreover, a heuristic criterion for the selection of the parental ordering is included to tackle the second point. A particular example is presented to motivate the choice of our proposed approximation and a simulation study is carried out to compare these methods.

[1] Recall for instance (4.8) when dealing with the Diamond shape PCBN.

[2] Recall the large logLikelihood difference when choosing different parental ordering in the Diamond shape PCBN from Example 4.2.

7.1. VINE DAG

Pircalabelu et al. (2017) were the first to propose a logLikelihood approximation using regular vines. The score is based on the fact that if X_n has parents $X_1, \dots, X_{n-1}$, then the PDF can be expressed as the quotient of:

$$f_{X_n|X_1,\dots,X_{n-1}}(x_n|x_1,\dots,x_{n-1}) = \frac{f_{X_1,\dots,X_n}(x_1,\dots,x_n)}{f_{X_1,\dots,X_{n-1}}(x_1,\dots,x_{n-1})} \tag{7.1}$$

Moreover, they proposed to model

- $f_{X_1,\dots,X_n}(x_1,\dots,x_n)$ using a C-Vine, with central node X_n [3].
- $f_{X_1,\dots,X_{n-1}}(x_1,\dots,x_{n-1})$ using a D-Vine [4].

Since theoretically all vine structures can be used to decompose the density, the assumptions included so far are valid. However, at this stage (7.1) propose:

$$f_{X_n|X_1,\dots,X_{n-1}}(x_n|x_1,\dots,x_{n-1}) = \frac{\text{CV}_{1,\dots,n}(x_1,\dots,x_n)}{\text{DV}_{1,\dots,n-1}(x_1,\dots,x_{n-1})}. \tag{7.2}$$

Indeed, the above is correct if the density of D-vine is computed from the density of C-vine by integrating out the first variable. Such computations are prohibitive.

The authors proposed to represent each of the conditional densities in the density factorization separately and designed an interesting penalty to create their approximation of the likelihood based score:

$$\begin{aligned} \text{IC} = & \underbrace{\sum_{m=1}^{N}\sum_{i=1}^{n}\left(\log\left(\text{CV}_{pa(i),i}(\boldsymbol{x}^m_{pa(i)}, x^m_i)\right) - \log\left(\text{DV}_{pa(i)}(\boldsymbol{x}^m_{pa(i)})\right)\right)}_{\text{logLikelihood}} \\ & \underbrace{-\sum_{m=1}^{N}\sum_{i=1}^{n}\frac{\log\left(\text{DV}_{pa(i)}(\boldsymbol{x}^m_{pa(i)})\right)}{|\text{pa}(i)|\log(n)}}_{\text{Penalty}}, \end{aligned} \tag{7.3}$$

where N is the sample size and n is the number of nodes.

In practice the authors fit the C-Vine and D-Vine separately and they fit each conditional density separately.

Few more choices are quite arbitrary in our opinion:

- The choice of the vine copula structures in the decomposition. In the C-Vine the child node is always included in the conditioned set, whereas we saw in Subsection 4.3.1 that the child node was always in the conditioning set. Furthermore, the type of vines are specified, but not the order of the nodes belonging to them.

[3] linking the central node: the child, to all of its parents.
[4] Linking all the parents, giving them equal importance.

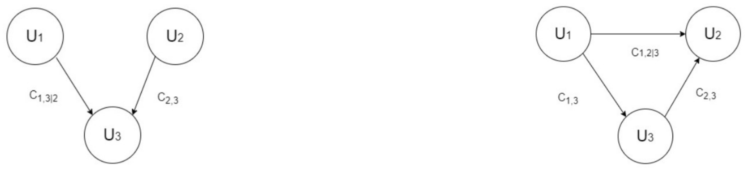
- The penalty is very difficult to motivate. It is based on part of the logLikelihood and that the number of parents is in the denominator.

To run the algorithm presented above the order of variables in the decomposition is needed. This point has not been addressed in the paper. Moreover, the testing of the proposed procedure included in the paper was minimal.

To gain more experience with the proposed method, several practical experiments were performed. Below we include a small example of the shortcomings of the proposed score function.

7.1.1. SHORTCOMINGS VINE DAG

Let's examine the shortcomings of this result. To do so we investigate two different PCBNs, BN_1 and BN_2 that can be seen in Figure 7.1. Both graphs are strongly chordal, so vine copula models can be used to simulate from them and also to estimate its logLikelihoods.



(a) BN_1. (b) BN_2.

Figure 7.1: Shortcoming of the CD-Vine decomposition's logLikelihood approximation.

Imagine that we simulate from BN_1 in Figure 7.1a. For this network we have:

- BIC (see (5.7)), estimating the logLikelihood [5] using the correct order $2 <_3 1$, and assigning a copula to each edge, we obtain:

$$\text{BIC } BN_1 = \underbrace{\sum_{m=1}^{N} \log\Big(\hat{c}_{2,3}(u_2^m, u_3^m) \cdot \hat{c}_{1,3|2}(u_{1|2}^m, u_{3|2}^m)\Big)}_{\text{Estimated logLikelihood}} - \underbrace{\frac{2}{2}\log(N)}_{\text{Penalisation}} . \tag{7.4}$$

- IC (see (7.3)), estimating the approximated logLikelihood. Since U_3 has two parents U_1 and U_2, the C-Vine will have as its central node U_3, while the D-Vine will consist of the copula joining the parents, i.e. $c_{1,2}$.

$$\text{IC } BN_1 = \underbrace{\sum_{m=1}^{N} \log\left(\frac{\hat{c}_{1,3}(u_1^m, u_3^m) \cdot \hat{c}_{2,3}(u_2^m, u_3^m) \cdot \hat{c}_{1,2|3}(u_{1|3}^m, u_{2|3}^m)}{\hat{c}_{1,2}(u_1^m, u_2^m)}\right)}_{\text{Approximated logLikelihood}}$$
$$- \underbrace{\sum_{m=1}^{N} \frac{\log(\hat{c}_{1,2}(u_1^m, u_2^m))}{2\log(N)}}_{\text{Penalisation}} . \tag{7.5}$$

[5] From now on the hat is used to denote that the copula is estimated (both family and parameters) from data.

Moreover, we also analyse BN_2 represented in Figure 7.1b. For this network we have:

- BIC (see (5.7)), estimating the logLikelihood using the order $3 <_2 1$, and assigning a copula to each edge, we obtain:

$$\text{BIC BN}_2 = \underbrace{\sum_{m=1}^{N} \log\left(\hat{c}_{1,3}(u_1^m, u_3^m) \cdot \hat{c}_{2,3}(u_2^m, u_3^m) \cdot \hat{c}_{1,2|3}(u_{1|3}^m, u_{2|3}^m)\right)}_{\text{Estimated logLikelihood}} - \underbrace{\frac{3}{2}\log(N)}_{\text{Penalisation}} . \tag{7.6}$$

- IC (see (7.3)), estimating the approximated logLikelihood. Since U_2 has two parents U_1 and U_3, the C-Vine will have as its central node U_2, while the D-Vine will consist of the copula joining the parents, i.e. $C_{1,3}$. In addition, U_3 has only one parent: U_1, so the D-Vine consist just of $c_{1,3}$. As the margins are uniform the C-Vine corresponding to node U_3 will be 1.

$$\text{IC BN}_2 = \underbrace{\sum_{m=1}^{N} \log\left(\frac{\hat{c}_{1,2}(u_1^m, u_2^m) \cdot \hat{c}_{2,3}(u_2^m, u_3^m) \cdot \hat{c}_{1,3|2}(u_{1|2}^m, u_{3|2}^m)}{\hat{c}_{1,3}(u_1^m, u_3^m)}\right) + \log\left(\hat{c}_{1,3}(u_1^m, u_3^m)\right)}_{\text{Approximated logLikelihood}}$$
$$- \underbrace{\sum_{m=1}^{N} \frac{\log(\hat{c}_{1,3}(u_1^m, u_3^m))}{2\log(N)}}_{\text{Penalisation}} . \tag{7.7}$$

Let's look at the expression (7.7). If we simulated from BN_1, in theory $C_{1,2} = C_{\perp}$, such that $c_{1,2}(u_1, u_2) = 1$. In practice $\hat{c}_{1,2}(u_1^m, u_2^m)$ will be close to 1, so when taking logarithms these terms will be negligible compared to the other terms in the logLikelihood. Moreover, the term $\hat{c}_{1,3}(u_1^m, u_3^m)$ vanishes as it is in numerator and denominator. Consequently:

$$\boxed{\text{Approximated logLikelihood of BN}_2 \approx \text{Estimated logLikelihood of BN}_1}$$

On the other hand, and again as $\log\left(\hat{c}_{1,2}(u_1^m, u_2^m)\right)$ will be negligible in (7.6). We then have that:

$$\boxed{\text{Approximated logLikelihood of BN}_1 \approx \text{Estimated logLikelihood of BN}_2}$$

Thus, a problem then arises since the CD-Vine decomposition approximation associates the logLikelihood of some networks with the true logLikelihood of others. We therefore expect the IC criterion to give priority to incorrect networks in most cases. Let's confirm our presumptions.

We simulate from BN_1 given by Figure 7.1a, using $m = 40$ different scenarios, results of the combination of:

- 10 family configurations shown in Table 7.1.

Table 7.1: Different Family Configurations, CD-Vine shortcomings.

	Family Configuration									
	1	2	3	4	5	6	7	8	9	10
$C_{1,3\|2}$	C	G	F	J	C	G	F	C	G	C
$C_{1,2}$	C	G	F	J	G	F	J	F	J	J

- 4 different parameter configurations shown in Table 7.2.

Table 7.2: Different Parameter Configurations, CD Vine shortcomings.

	Parameter Configuration			
	1	2	3	4
$\tau_{1,3\|2}$	0.25	0.25	0.75	0.75
$\tau_{2,3}$	0.25	0.75	0.25	0.75

For each of the above scenarios, we set the seeds to get reproducible results and $N = 2500$ is taken as a sample size during this example. Moreover, we compute:

- BIC BN_1, given by (7.4).
- BIC BN_2, given by (7.6).
- IC BN_1, given by (7.5).
- IC BN_1, given by (7.5).

The results obtained can be summarized as follows

1. In 39 out of 40 scenarios BIC $\text{BN}_1 >$ BIC BN_2, and hence the BIC score prioritise the true network over the other one.

2. In 40 out of 40 scenarios IC $\text{BN}_2 >$ IC BN_1, and hence the IC score always prioritise the other network over the true one.

Taking into account that the SHD between the two networks is SHD = 3, we show the significant errors that are made if we choose the IC criterion as the score. This example illustrate the limitations of this approach.

On the other hand, Pircalabelu et al. (2017) demonstrate the consistency of the penalty term in their proposed score. The proof can be seen in Lemma 4 in the previous paper. This leads us to believe that what fails in the score function is the approximation of the logLikelihood. Therefore, in the remainder of this chapter we will only focus on the study of logLikelihood approximations.

7.2. ESTIMATING EXTRA COPULAS

To simplify computation of likelihoods of PCBNs we propose to instead of integrating, we will estimate all the necessary copulas from data. The proposed score function will be then taken as an approximation of the likelihood and BIC penalty.

Example 7.1 (Approximation logLikelihood Diamond shape PCBN) *This procedure might become more clear with an example. Let's go back to the Diamond shape PCBN studied in Example 4.2.*

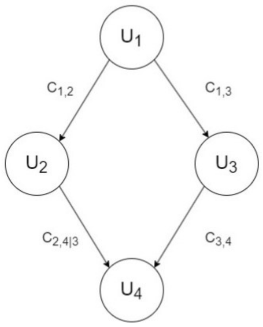


Figure 7.2: Diamond shape PCBN.

To evaluate the conditional copula $c_{2,4|3}$, the pseudo-observations $u_{2|3}$ and $u_{4|3}$ must be computed. This could be done applying (2.13) to the copulas $C_{2,3}$ and $C_{3,4}$, i.e:

$$u_{2|3} = \frac{\partial C_{2,3}(u_2,u_3)}{\partial u_3} \;,\; u_{4,3} = \frac{\partial C_{3,4}(u_3,u_4)}{\partial u_4}.$$

The PCBN decomposition specifies the copula $C_{3,4}$, allowing for the direct computation of $u_{4|3}$. In contrast, the decomposition does not directly include $C_{2,3}$. This copula has to be calculated using integrals:

$$C_{2,3}(u_2,u_3) = \int_0^{u_3}\int_0^1 \frac{\partial C_{1,2}(\omega_1,u_2)}{\partial \omega_1}\cdot c_{1,3}(\omega_1,\omega_3)\cdot d\omega_1\cdot d\omega_3. \tag{7.8}$$

Instead of computing $C_{2,3}$ using (7.8), our proposal is to estimate $\hat{C}_{2,3}$ directly from data. Differentiating this copula [6]*, we obtain an approximation $\hat{u}_{2|3}$ of $u_{2|3}$. The final step is to estimate $\hat{C}_{2,4|3}$ using the approximation $\hat{u}_{2|3}$ and the observation $u_{4|3}$. The approximated logLikelihood becomes:*

$$Approx.logLik(\boldsymbol{u},\boldsymbol{\theta}) = \sum_{m=1}^{N} log\big(c_{1,2}(u_1^m,u_2^m)\big) + log\big(c_{1,3}(u_1^m,u_3^m)\big) + \\ log\left(\underbrace{\hat{c}_{2,3}(u_2^m,u_3^m)\cdot c_{3,4}(u_3^m,u_4^m)\cdot \hat{c}_{2,4|3}(\hat{u}_{2|3}^m,u_{4|3}^m)}_{D\text{-}Vine}\right) - log\big(\hat{c}_{2,3}(u_2^m,u_3^m)\big). \tag{7.9}$$

Hence, no integration is required to compute this approximated logLikelihood. A motivational example showing good qualitative and quantitative properties of these procedures will be included later in Section 7.3.

7.2.1. GENERAL PROCEDURE

In a more general framework, if the node $v \in \mathbb{V}$, has an ordered parental set given by $K = \{\omega_1,\dots,\omega_n\}$, we saw in Subsection 4.3.1, that the PDF could be expressed as:

$$f_{X_v|\boldsymbol{X}_K}(x_v|\boldsymbol{x}_K) = f_{X_v}(x_v)\prod_{i=1}^{n} c_{v,\omega_i|K_{-i}}\left(F_{X_v|\boldsymbol{X}_{K_{-i}}}(x_v|\boldsymbol{x}_{K_{-i}}), F_{X_{\omega_i}|\boldsymbol{X}_{K_{-i}}}(x_{\omega_i}|\boldsymbol{x}_{K_{-i}});\boldsymbol{x}_{K_{-i}}\right), \tag{7.10}$$

[6] as in (2.13)

where $K_{-i} = \{\omega_{i+1}, \dots, \omega_n\}$, for every $i \in \{1, \dots, n\}$. The copulas in this prior decomposition are $C_{v,\omega_n}, C_{v,\omega_{n-1}|\omega_n}, \dots, C_{v,\omega_2|\omega_3,\dots,\omega_n}, C_{v,\omega_1|\omega_2,\dots,\omega_n}$.

Our proposal is then to construct a vine containing all the previous copulas, allowing us to estimate them without the need of integrals. This can be done by choosing a D-Vine with order in the 1^{st} tree given by $D = \{\omega_1, \omega_2 \dots, \omega_n, v\}$. By doing this, a regular vine is obtained in which the last node of each tree corresponds to each of the preceding copulas. This is better illustrated in Figure 7.3.

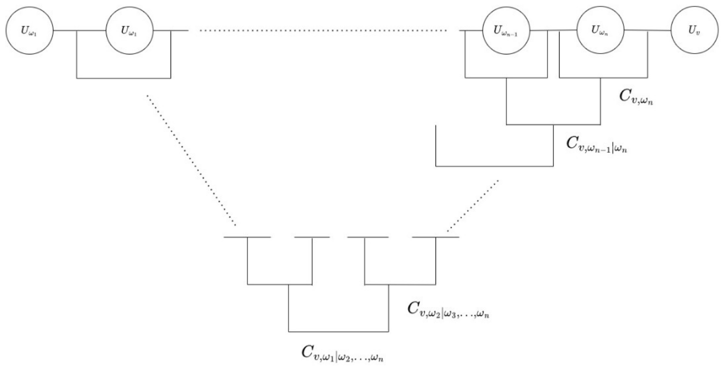

U_{ω_1}
U_{ω_1}
$U_{\omega_{n-1}}$
U_{ω_n}
U_v
C_{v,ω_n}
$C_{v,\omega_{n-1}|\omega_n}$
$C_{v,\omega_2|\omega_3,...,\omega_n}$
$C_{v,\omega_1|\omega_2,...,\omega_n}$

Figure 7.3: D-Vine, estimating extra copulas to avoid integrals.

By using the `RVineCopSelect` command of the `VineCopula` package, this D-Vine can be fitted from data. The output includes the logLikelihood of each of the copulas that compose the D-Vine. By selecting just the terms that appear in (7.10) and adding them up, we obtain the desired logLikelihood approximation of $\log\left(f_{X_v|\boldsymbol{X}_K}(x_v|\boldsymbol{x}_K)\right)$.

7.2.2. PARENTAL ORDERING

To select the set of parental orderings $\mathscr{O}$, the approach suggested by Bauer and Czado (2016) is employed. The foundation of this approach is the application of a greedy-type procedure inspired by the structure selection algorithm for vine copula models [7].

Imagine that the node $v \in \mathbb{V}$, has a parental set given by $K = \{\omega_1, \dots, \omega_n\}$, and assume that we have already selected the $n-i$ smallest parents of v, denoted by the ordered subset $K_{-i} = \{\omega_n <_v \dots <_v \omega_{i+1}\}$. This implies that the pair-copula families for

$$C_{v,\omega_n}, \dots, C_{v,\omega_{i+1}|\omega_{i+2},\dots,\omega_n},$$

have already been chosen, and its parameters have been estimated. The selection of the $n-i+1$ smallest parent is performed in three steps.

1. A truncated C-Vine up to level $n-i+1$, with order $C = \{\omega_n, \dots, \omega_{i+1}, v\}$ is constructed. The copulas in the last tree will be given by $C_{v,\omega|K_{-i}}$, for all $\omega \in K \backslash K_{-i} = \{\omega_1, \dots, \omega_i\}$.

[7] See Subsection 2.2.4

2. The previous C-Vine is fitted and the Kendall's τ for each of the copulas in the last tree are computed.

3. We select $\omega \in \{\omega_1, \dots, \omega_i\}$ such that it maximizes the absolute value of $\tau_{v,\omega|\omega_{i+1},\dots,\omega_n}$. For such a node, we select the copula family and we estimate its parameters.

Repeating this process $pa(v)$ times for each $v \in \mathbb{V}$, the set of parental orderings $\mathcal{O}$ is obtained. It is worth mentioning that this procedure is heuristic, and in many cases it will not recover the true order.

7.3. Motivational Example

The proposed approach is then explored for the particular example of the diamond-shaped PCBN. The data simulated in Example 4.2, with parameters given by Table 4.1 and plotted in Figure 4.7 is used in this section.

7.3.1. LogLikelihood Comparison

First, the true logLikelihood and its different approximations are computed.

- True logLikelihood [8]:

$$\begin{aligned}\text{logLik}(\boldsymbol{u},\boldsymbol{\theta}) = \sum_{m=1}^{N} & \log\left(c_{1,2}(u_1^m, u_2^m)\right) + \log\left(c_{1,3}(u_1^m, u_3^m)\right) + \log\left(c_{3,4}(u_3^m, u_4^m)\right) \\ & + \log\left(c_{2,4|3}\left(\int_0^1 \frac{\partial C_{1,2}(\omega_1, u_2^m)}{\partial \omega_1} \cdot c_{1,3}(\omega_1, u_3^m) \cdot d\omega_1, C_{4|3}(u_4^m | u_3^m)\right)\right)\end{aligned} \tag{7.11}$$

- Estimated LogLikelihood using CD-Vine approximation.

$$\begin{aligned}\text{logLik}(\boldsymbol{u},\hat{\boldsymbol{\theta}}) = \sum_{m=1}^{N} & \log\left(\hat{c}_{1,2}(u_1^m, u_2^m)\right) + \log\left(\hat{c}_{1,3}(u_1^m, u_3^m)\right) \\ & + \log\left(\frac{\hat{c}_{2,4}(u_2^m, u_4^m) \cdot \hat{c}_{3,4}(u_3^m, u_4^m) \cdot \hat{c}_{2,3|4}(\hat{C}_{2|4}(u_2^m|u_4^m), \hat{C}_{3|4}(u_4^m|u_2^m)}{\hat{c}_{2,3}(u_2^m, u_3^m)}\right)\end{aligned} \tag{7.12}$$

- Estimated logLikelihood computing extra margins using the correct order.

$$\begin{aligned}\text{logLik}(\boldsymbol{u},\hat{\boldsymbol{\theta}}) = \sum_{m=1}^{N} & \log\left(\hat{c}_{1,2}(u_1^m, u_2^m)\right) + \log\left(\hat{c}_{1,3}(u_1^m, u_3^m)\right) \\ & + \log\left(\hat{c}_{2,3}(u_2^m, u_3^m) \cdot \hat{c}_{3,4}(u_3^m, u_4^m) \cdot \hat{c}_{2,4|3}(\hat{C}_{2|3}(u_2^m|u_3^m), \hat{C}_{4|3}(u_4^m|u_3^m)\right) \\ & - \log\left(\hat{c}_{2,3}(u_2^m, u_3^m)\right)\end{aligned} \tag{7.13}$$

- Estimated logLikelihood computing extra margins using the wrong order.

[8] Parameters and copula families specified by Table 4.1

$$\begin{aligned}\text{logLik}(\boldsymbol{u},\hat{\boldsymbol{\theta}}) = \sum_{m=1}^{N} & \log\left(\hat{c}_{1,2}(u_1^m, u_2^m)\right) + \log\left(\hat{c}_{1,3}(u_1^m, u_3^m)\right) \\ & + \log\left(\hat{c}_{2,3}(u_2^m, u_3^m) \cdot \hat{c}_{2,4}(u_2^m, u_4^m) \cdot \hat{c}_{3,4|2}(\hat{C}_{3|2}(u_3^m|u_2^m), \hat{C}_{4|2}(u_4^m|u_2^m)\right) \\ & - \log\left(\hat{c}_{2,3}(u_2^m, u_3^m)\right)\end{aligned} \quad (7.14)$$

The obtained results can be seen in Table 7.3.

Table 7.3: Values and Computational times of the different logLikelihood approximations for the Diamond shape PCBN.

	Value	Computational Time
True logLik	7029.19	93.76
Estim. logLik CD-Vine	6176.758	8.102
Estim. logLik, extra copulas, true order	6773.572	8.245
Estim. logLik, extra copulas, wrong order	6357.682	8.634

Regardless of the chosen order, our proposed method yields better approximations than the CD-Vine approach. The relative error of the CD-Vine decomposition is 12.12%, while for our procedure with the true order is only 3.64%. This once again highlights the shortcomings of the CD-Vine approach. It also suggests that using our method might be a good idea. Indeed, a small error is still made, but this procedure is more than 10 times faster than computing the integrals.

Regarding the parental ordering and as we have $\tau_{2,4} = 0.77 > \tau_{3,4} = 0.50$, the previous criteria choose the wrong order $2 <_4 3$. As a consequence, a worse estimate of the log-Likelihood is obtained. This example illustrate how heuristic methods can sometimes fail. A more intensive study of this heuristic method is needed.

7.3.2. VISUAL ANALYSIS AND GOODNESS OF FIT TEST

We examine now how the estimation of extra copulas affects the data. To do so, we compare true margins (obtained by integration) with the copulas estimated using our method.

COMPARISON OF TRUE MARGINS $U_2 - U_3$ VS ESTIMATED COPULA $\hat{C}_{2,3}$

This comparison can be seen by means of scatter plots in Figure 7.4, where we include the values of τ and the copula families.

At first glance these scatter plots are very similar. Both have a lower tail dependence and the values of Kendall's τ are close Nevertheless, we need a quantitative study to confirm our findings. To do so a goodness-of-fit test [9] is performed to assess whether the estimated copula fits the true margins:

```
$p.value.CvM    $p.value.KS    $statistic.CvM   $statistic.KS
[1] 0           [1] 0          [1] 0.2724597    [1] 1.112736
```

[9]See the goodness of fit test discussed in Section 2.1

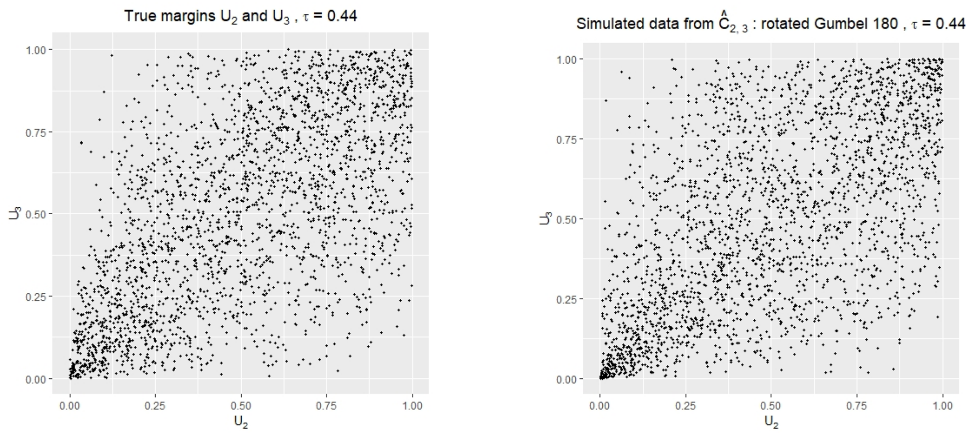
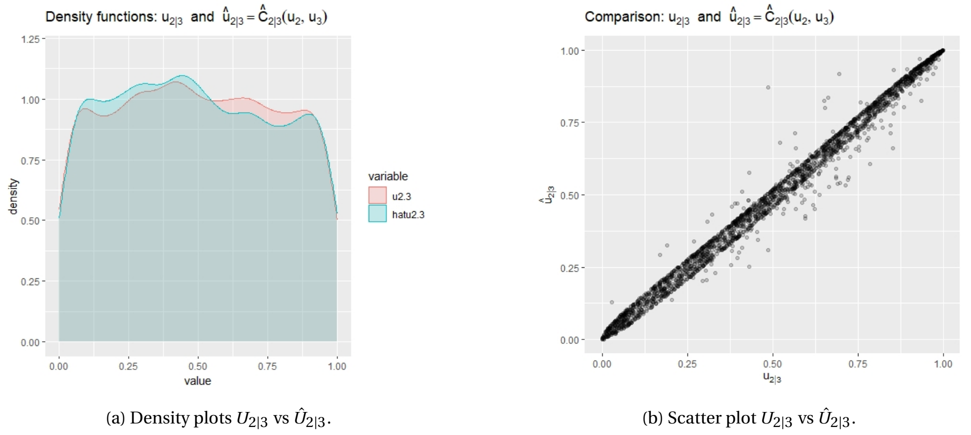


Figure 7.4: True margins U_2 and U_3 vs Data simulated from $\hat{C}_{2,3}$.

These tests reject the null hypothesis H_0 that margins U_2, U_3 come from the copula $\hat{C}_{2,3}$. This highlights how in order to simulate from this PCBN, integrations are required.

Comparison of True Margin $U_{2|3}$ vs Estimated Margin $\hat{U}_{2|3} = \hat{C}_{2|3}(U_2, U_3)$

In the approximation of likelihood we are not using the density of copula C_{23}. This copula is used to compute pseudo-observations $U_{2|3}$. We thus compare the true margins $U_{2|3}$ with the estimated ones $\hat{U}_{2|3}$ generated using the copula $\hat{C}_{2|3}$. This comparison can be seen in Figure 7.5.



(a) Density plots $U_{2|3}$ vs $\hat{U}_{2|3}$.

(b) Scatter plot $U_{2|3}$ vs $\hat{U}_{2|3}$.

Figure 7.5: Comparison of True Margin $U_{2|3}$ vs Estimated Margin $\hat{U}_{2|3} = \hat{C}_{2|3}(U_2, U_3)$.

In the figure on the left we can see how the distribution of $\hat{U}_{2|3}$ is approximately uniform in the interval $[0, 1]$. In the figure on the right we can observe how most of the points lie on the diagonal. Therefore, for most of the observations, its approximation is very close to the true value. In addition, we perform a goodness of fit test to check whether $\hat{U}_{2|3}$ is uniform.

```
One-sample Kolmogorov-Smirnov test
data:  hatu2.3, D = 0.027141, p-value = 0.05029
```

The p-value is greater than $\alpha = 0.05$, so there is not enough statistical evidence to reject the null hypothesis that $\hat{U}_{2|3}$ is uniform.

COMPARISON OF TRUE MARGINS $U_{2|3} - U_{4|3}$ VS ESTIMATED COPULA $\hat{C}_{2,4|3}$

Finally, including the margin $U_{4|3}$, we study the differences between the true copula $C_{2,4|3}$ and the estimated one $\hat{C}_{2,4|3}$ using the approximations. This comparison can be seen by means of scatter plots in Figure 7.6.

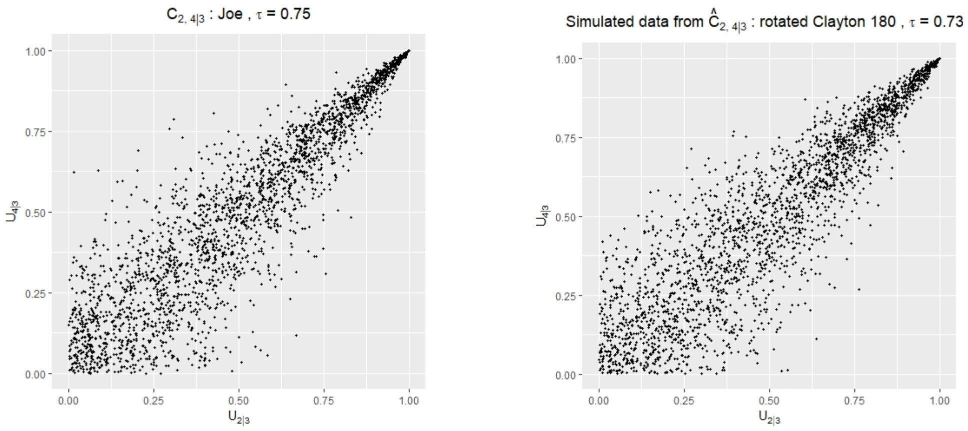


Figure 7.6: True conditional margins $U_{2|3}$ and $U_{4|3}$ vs Data simulated from $\hat{C}_{2,4|3}$.

In general terms the approximation made does not affect the shape of the data. Both distributions have upper tail dependence. There are differences between the families, but the values of both Kendall's τ are close. Moreover, a goodness-of-fit test is performed to obtain more quantitative insights:

```
$p.value.CvM     $p.value.KS     $statistic.CvM     $statistic.KS
[1] 0.6666667    [1] 0.7         [1] 0.02784511     [1] 0.4738971
```

There is not enough statistical evidence to reject the null hypothesis at level $\alpha = 0.05$. This shows that actually estimating copulas instead of carrying out integration is not a bad approximation. Although, there is still a difference in logLikelihoods:

$$\boxed{log(C_{2,4|3}) = 2734.239, \;\; log(\hat{C}_{2,4|3}) = 2478.053}$$

7.4. SIMULATION STUDY

Finally, a small simulation study is carried out on the suitability of the approximations to obtain the PCBN logLikelihoods.

7.4.1. SIMULATION OBJECTIVES

The main objectives of this simulation study are:

1. To analyse how good the previously explained logLikelihood approximations are.
2. To compare the performance of the approximations using the CD-Vine decomposition vs. estimating extra copulas.
3. To investigate the trade-off between the logLikelihood error incurred in these approximations vs the reduction in computational time involved in their use.
4. To inspect how appropriate is the proposed parental ordering.

7.4.2. SIMULATION SETUP

We have selected only one DAG to conduct this simulation study. This DAG is precisely the diamond-shape one. The choice of this DAG is based on the fact that:

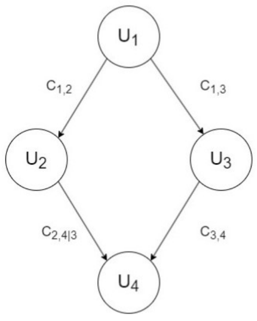


Figure 7.7: Diamond shape PCBN Simulation Study Chapter 7.

- We need a network where there is at least one node with more than one parent. This implies that the CD Vine approximation and the approximation by estimating additional copulas lead to different results. This fact rules out all tree-shaped DAGs.
- We need a DAG with a non-strongly chordal skeleton, so that integrations are necessary for the computation of its logLikelihood. On the other hand, these integrations are often extremely costly. In our case only a one-dimensional integration will be required.

We simulate from $m = 80$ different scenarios, results of the combination of:

- 10 family configurations shown in Table 7.4.

Table 7.4: Different Family Configurations, Simulation Study Diamond-Shape PCBN.

	Family Configuration									
	1	2	3	4	5	6	7	8	9	10
$C_{1,2}$	t	C	G	F	J	t	G	t	G	J
$C_{1,3}$	t	C	G	F	J	t	G	C	F	t
$C_{3,4}$	t	C	G	F	J	C	F	G	J	C
$C_{2,4\|3}$	t	C	G	F	J	C	F	F	t	G

- 8 different parameter configurations shown in Table 7.5.

Table 7.5: Different Parameter Configurations, Simulation Study Diamond-Shape PCBN.

	Parameter Configuration							
	1	2	3	4	5	6	7	8
$\tau_{1,2}$	0.25	0.75	0.25	0.75	0.25	0.75	0.25	0.75
$\tau_{1,3}$	0.25	0.75	0.25	0.75	0.25	0.75	0.75	0.25
$\tau_{3,4}$	0.25	0.75	0.75	0.25	0.25	0.75	0.25	0.75
$\tau_{2,4\|3}$	0.25	0.75	0.75	0.25	0.75	0.25	0.75	0.25

For each of the above scenarios we only simulated $M = 1$ time due to the computational burden of the simulations. We set the seeds to get reproducible results. Moreover, $N = 2500$ is taken as a sample size during the whole study.

7.4.3. SIMULATION RESULTS

For each of the scenarios, we compute:

1. True logLik, given by (7.11).
2. LogLik using CD-Vine decomposition, given by (7.12).
3. LogLik computing extra margins using the true order, given by (7.13).
4. LogLik computing extra margins using the wrong order, given by (7.14).
5. LogLik computing extra margins, using the previously described parental ordering heuristic.

The logLikelihood values vary greatly between scenarios, depending on whether the copulas are highly correlated or low correlated. Therefore, we compute the relative errors between the true logLik and the approximations made. This makes it easier for us to compare different scenarios. In addition, the respective computational times are stored.

For a better visual analysis of the results, density plots are created for both relative errors and computational times. Furthermore, the medians of these quantities are computed and represented with a vertical line in the above plots. We also include the values of the quantiles: $Q_{0.05}$, $Q_{0.25}$, $Q_{0.75}$ and $Q_{0.95}$, to have a better quantitative understanding of the spread of the distribution. The obtained results can be seen in:

- Relative Errors: Table 7.6, Figure 7.8.
- Computational Times: Table 7.7, Figure 7.9.

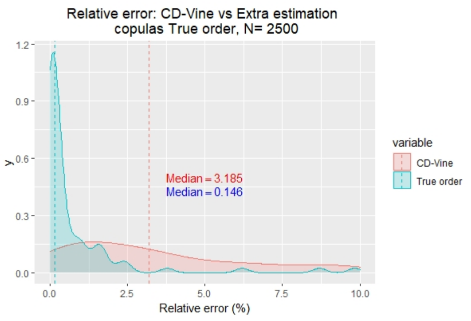
We now analyse the results obtained.

Table 7.6: LogLikelihood Relative Errors, Simulation Study Diamond-Shape PCBN.

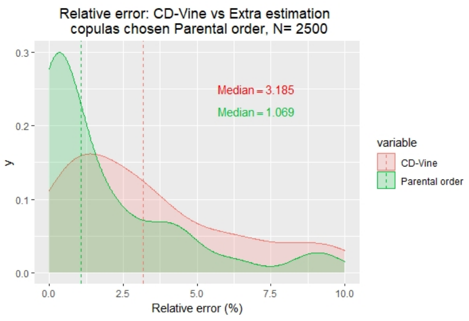
	$Q_{0.05}$	$Q_{0.25}$	Median	$Q_{0.75}$	$Q_{0.95}$
CD-Vine	0.167	1.307	3.185	7.358	13.951
True order	0.006	0.039	0.146	0.797	2.514
Wrong order	0.093	1.252	4.084	9.577	23.258
Parental order	0.025	0.095	1.069	4.239	14.322

Table 7.7: Computational Times, Simulation Study Diamond-Shape PCBN.

	$Q_{0.05}$	$Q_{0.25}$	Median	$Q_{0.75}$	$Q_{0.95}$
True logLik	139.214	156.717	190.453	247.769	388.397
CD-Vine	6.096	7.729	8.634	9.225	10.737
True order	7.233	8.188	8.818	9.612	11.377
Wrong order	7.131	8.017	8.550	9.263	11.278

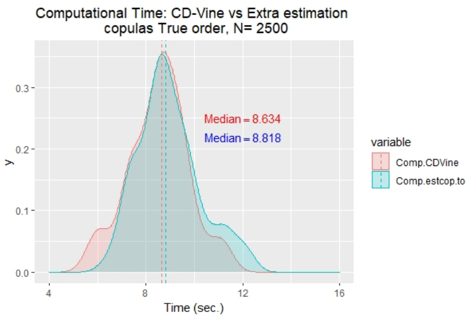


(a) Density plots: Relative Error CD-Vine approach vs Estimation Extra Copula, True Order.

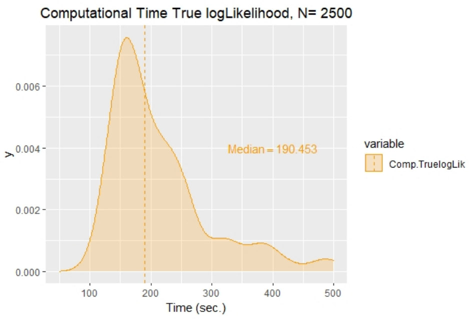


(b) Density plots: Relative Error CD-Vine approach vs Estimation Extra Copula, Parental Order Criteria.

Figure 7.8: Comparison of the LogLikelihood Relative Errors using the CD-Vine approximation vs Estimation of Extra Copulas, simulation Study Diamond-Shape PCBN.



(a) Density plots: Computational Times CD-Vine approach vs Estimation Extra Copula, True Order.



(b) Density plot: Computational Times True LogLikelihood.

Figure 7.9: Comparison of the Computational Times using the studied Approximations vs Computing the True LogLikelihood, Simulation Study Diamond-Shape PCBN.

7.4.4. Simulation Analysis

The best among the approximations of the true logLikelihood is to estimate additional copulas with the true order. Indeed, in more than 75% of the scenarios, the relative error is lower than 1%. This is really good considering that the approximation is more than 20 times faster than the integral computations. Moreover, it can be observed in Figure 7.8a that the proposed true order approach outperforms by far the CD-Vine method. In fact, it is better in 75 out of 80 scenarios [10].

On the other hand, the situation changes a lot if we choose the wrong order. In that case, about 25% of the scenarios yield a relative error larger than 10%. Making this a rather poor approximation. In fact the CD-Vine method beats the proposed approximation with the wrong order in 45 out of 80 scenarios. These scenarios include all cases in which the copula $C_{3,4}$ is highly correlated, $\tau_{3,4} = 0.75$ [11]. This emphasizes how important it is to choose the correct order.

In this case computing all possible orders is not a computationally demanding task [12]. In many other cases, a heuristic criterion for the choice of the parental ordering will be needed. The criterion proposed by Bauer and Czado (2016), selects the true order in only 32 out of 80 scenarios. Despite this, we can observe in Figure 7.8b that this method is superior to the CD-Vine approach. This procedure yields a more accurate approximation in 61 out of 80 scenarios. Moreover, it results in a relative error of less than 1% in approximately half of the scenarios. On the other hand, it can get errors of more than 10%, as a consequence of choosing the wrong order.

7.4.5. Simulation Discussion

This simulation study confirms that implementing search algorithms based on the exact logLikelihood expressions is computationally unfeasible [13]. Instead, approximations are required to speed up computations. In this task, the above results suggest that our proposed method outperforms the CD-Vine approach.

If the number of parents is not too large, all parental orderings can be checked. This would guarantee to find the correct order. Consequently, very accurate and fast approximations are obtained. Heuristic techniques are required in several other situations to handle the parental ordering. The simulation study findings demonstrate that even using heuristics, the proposed approximations are superior to the CD-Vine.

Everything seems to indicate that implementing search algorithms with a score based on this logLikelihood approximation can give good results. However, a more in-depth simulation study is needed to obtain more meaningful results or to support the findings in this simulation study.

[10]The remaining 5 scenarios correspond to the case where all copulas are t.
[11]i.e, parameter configurations 2, 3, 6 and 8 in Table 7.5
[12]Indeed, there are only two.
[13]See Figure 7.9b.

7.5. PROPOSAL

Although the above approximations are relatively fast, starting with an empty graph in score-based algorithms for PCBNs is a computationally challenging task. Instead, it seems like a wise idea to:

1. Apply Gaussian Structure Learning algorithm to our PCBN data to obtain an initial graph, as we studied during Chapter 6.
2. Employ the Hill Climbing algorithm based on the logLikelihood approximation proposed in Section 7.2 and the parental ordering presented in Subsection 7.2.2.

This is precisely the method proposed in this dissertation to learn the structure of PCBNs. It is outlined in Algorithm 4.

Algorithm 4 Proposed algorithm for Structure Learning of PCBNs.

```
    Input Data set X.
    Output Directed Acyclic Graph $\mathscr{G}_{\text{final}}$ with parental ordering $\mathscr{O}$, representing the PCBN.
 1: Transform margins to Gaussian;
 2: Apply Gaussian Learning algorithms to obtain an initial graph $\mathscr{G}_{\text{start}}$;
 3: Order the nodes in $\mathscr{G}_{\text{start}}$ utilizing the parental ordering criteria and set $\mathscr{O} = \text{order}(\mathscr{G}_{\text{start}})$;
 4: Use the logLikelihood approximation to compute the score of $\mathscr{G}_{\text{start}}$, $\text{Score}_{\mathscr{G}_{\text{start}}} = \text{Score}(\mathscr{G}_{\text{start}})$
    and set max.score = $\text{Score}_{\mathscr{G}}$.
 5: repeat
 6:     for for addition, deletion or reversal of randomly selected arcs resulting in a DAG: do
 7:         Order the parents;
 8:         Use the logLikelihood approximation to compute the score of the modified network,
 9:         $\text{Score}_{\mathscr{G}^*} = \text{Score}(\mathscr{G}^*)$
10:         if $\text{Score}_{\mathscr{G}^*} > \text{Score}_{\mathscr{G}}$, set $\mathscr{G} = \mathscr{G}^*$, $\mathscr{O} = \text{order}(\mathscr{G}^*)$ and $\text{Score}_{\mathscr{G}} = \text{Score}_{\mathscr{G}^*}$. then
11:             update max.score = $\text{Score}_{\mathscr{G}}$.
12:         end if
13:     end for
14: until max.score does not increase.
15: $\mathscr{G}_{\text{final}} = \mathscr{G}$ and $\mathscr{O} = \text{order}(\mathscr{G})$.
```

The application of this algorithm is illustrated below in a concrete example.

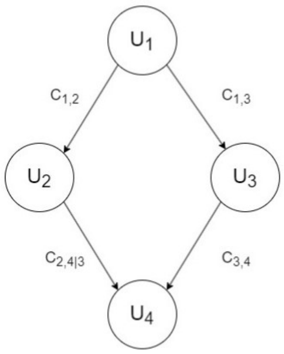
7.5.1. APPLICATION EXAMPLE

Let's continue with the data simulated from the Diamond-shape PCBN in Example 4.2. First, the HC algorithm using the Gaussian procedure [14] is applied. Then, the previously explained criteria is used to choose the parental ordering. The resulting DAG can be seen in Figure 7.10b.

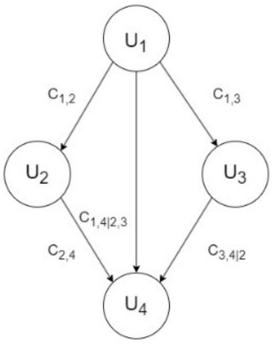
The next step is to compute the chosen score. To account for the penalty, the BIC given by (5.7) is used as in this example. The obtained score is then:

$$\boxed{\text{BIC}_{\text{Initial DAG}} = 6338.122}$$

[14] Please see Section 6.4 for more details.



(a) Diamond shape PCBN.

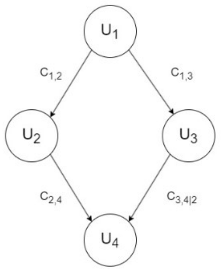


(b) PCBN result from: Gaussian HC Algorithm + Parental Order Criterion.

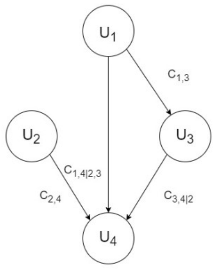
Figure 7.10: True PCBN and Estimated PCBN using Gaussian HC Algorithm and Parental Order Criterion.

The HC algorithm then randomly adds, removes or reverses different edges. It also uses the criteria to order the parents and computes the resulting score. Below, some possible steps of the algorithm are shown.

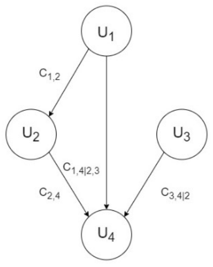
REMOVING ONE EDGE



(a) Removing $U_1 \to U_4$, BIC = 6342.03.



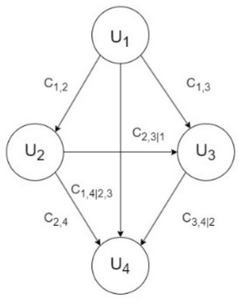
(b) Removing $U_1 \to U_2$, BIC = 5288.03.



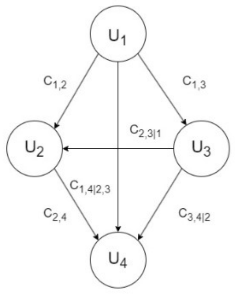
(c) Removing $U_1 \to U_3$, BIC = 3862.97.

Figure 7.11: Removing 1 edge, HC Algorithm applied to Diamond-shape PCBN.

ADDING ONE EDGE



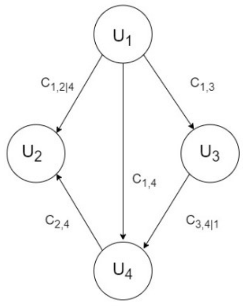
(a) Adding $U_2 \to U_3$,
BIC = 6334.21.



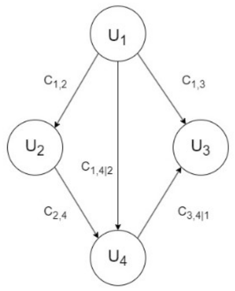
(b) Adding $U_3 \to U_2$,
BIC = 6334.21.

Figure 7.12: Adding 1 edge, HC Algorithm applied to Diamond-shape PCBN.

REVERSING ONE EDGE



(a) Reversing $U_4 \to U_3$,
BIC = 6288.29.



(b) Reversing $U_4 \to U_2$,
BIC = 6060.20.

Figure 7.13: Reversing 1 edge, HC Algorithm applied to Diamond-shape PCBN.

Of all the possible movements we have checked, the DAG that results in the highest BIC is the one given by Figure 7.11a. Therefore, we would recover the true structure. Although, the order of the parents is the wrong one. This example illustrate how promising is the suggested method to learn the structure of PCBNs.

Unfortunately, we arrived at these ideas at the end of the research period. So there was no time left to implement and test this procedure. Another interesting point to investigate in the future is the search for suitable penalties for our score. Therefore, we leave these lines of research open for future work, to support our beliefs.

8 Discussion and Future Work

The aim of this study was to investigate structure learning algorithms for both Gaussian Bayesian Networks and Pair Copula Bayesian Network. First, we presented an overview of the two types of BNs, illustrating its properties and differences. Second, we provided an outline of the different existing structure learning algorithms, showing their efficiency for the Gaussian case and limitations for the copula based. The performance of Gaussian structure learning algorithms for PCBNs was evaluated. Finally, different logLikelihood approaches to construct PCBN score-based algorithms were studied. All these points will be discussed in the following paragraphs.

We first looked at the Gaussian case. It was shown that GBNs were equivalent to multivariate Gaussian distributions. Thus, acquiring all the good properties of these distributions. Indeed, these distributions were fully characterized by their mean and covariance matrix, making them computationally tractable using simple algebra. Furthermore, the independencies and conditional independencies of the distribution could be directly derived from the covariance matrix. On the other hand, these networks were very restrictive and made strong assumptions that were frequently violated in practice.

To address the previous problem, we examined PCBNs, in which distributions were expressed as the product of bivariate copulas. These models could account for a wide range of distributional features, such as tail dependencies, non-linearities, and asymmetric dependencies. Thus, strict Gaussian assumptions were relaxed. However, these distributions were not tractable and usually involved computationally expensive integrals. In addition, the independencies and conditionally independencies could no longer be read from the covariance matrix. Finally, due to the simplifying assumptions, different orders produced different decompositions, which made learning the structure of PCBNs a NP-hard problem.

The following step was to give an overview of the different structure learning algorithms. We studied three different algorithms: Graphical Lasso, Constraint-based and Score-based. The good properties of these algorithms for the Gaussian case were illustrated:

- The Graphical Lasso was proven to be asymptotically consistent [1] in estimating the set of non-zero elements of the precision matrix, and therefore recovering the true moralised graph.
- For the Constraint-based we showed that 0-correlation implies independence. As a consequence, accurate independence tests could be performed by simply using the covariance matrix.
- Lastly, the tractability properties of the Gaussian distribution made the Score-based algorithms fast and powerful.

All these statements were validated by means of a simulation study, where the high efficiency of these methods was proven.

Conversely, the non-Gaussian case was shown to present numerous problems:

- Indeed, the Graphical Lasso worked under Gaussian assumptions, causing misspecification for PCBN structures.
- Outside the Gaussian world, 0-correlation did not imply independence. Consequently, other conditional independence test were required in Constraint-based algorithms for PCBNs. The issue was that other types of tests were computationally demanding, making these algorithms computationally infeasible.
- Finally, we showed the shortcomings that arose in score-based algorithms for PCBNs. LogLikelihood scores were not equivalent; even for the same DAG, the scores resulting from taking different parental orderings were quite different. Furthermore, the logLikelihood computation often involved computational expensive integrals, making these algorithms not really practical.

Following these findings, two different procedures were investigated to enhance the structure learning of PCBNs.

First, Gaussian learning algorithms were applied to PCBNs. Indeed, after seeing the high efficiency of the Gaussian case, we wondered whether these procedures could help us to obtain some information about the structure of PCBNs. To this end, we conducted a simulation study to test the suitability of these procedures. The results suggested that these methods were not completely efficient in learning the structure, obtaining misspecifications in general. However, these procedures proved to be beneficial. In fact, all three algorithms were able to recover all true edges. This suggested that we could use these results to obtain an initial graph. This graph could then be used to initialize a score-based algorithm with a score suitable for PCBNs.

Second, Score-based algorithms for PCBNs were examined. As we mentioned earlier, the logLikelihood scores in these networks often involved integrals and were not equivalent.

[1] see Friedman et al., 2008

To tackle the former problem a logLikelihood approximation based on estimating extra copulas was proposed. To address the latter issue, a heuristic technique was introduced. A simulation study was carried out to test the appropriateness of these procedures. The results indicated that our proposed method outperform previously proposed approximations. Moreover, they showed that the computation of exact logLikelihood expressions was computationally unfeasible, making our proposal an attractive candidate.

These two lines of research have shown promise. However, we believe that it is the combination of the two that could really yield powerful results for PCBN structure learning. This dissertation therefore proposes to apply Gaussian learning algorithms to find a graph, which would serve as a starting point for a Hill Climbing algorithm based on our proposed logLikelihood approximation. This method must be tested and validated by means of a simulation study.

Along with the implementation of the proposed method, there are several paths open for further research:

- First, a more in-depth simulation study on the performance of Gaussian methods for PCBNs could be performed. Indeed, more networks could be included, for example a high-dimensional and dense network. On the other hand, we saw that in the larger network, we always overestimated the number of edges. So it could be a good idea:
  - Use other criteria [2] that would result in higher values for the regularization parameter λ. Thus, the moralized network becomes sparse.
  - Reduce the value of the significance level α in hypothesis testing of conditional independencies. For example, following the Bonferroni correction used in the case of multiple testing. By lowering the value of α, the number of test rejecting the null hypothesis would decrease, and thus more edges would be removed from the network.
  - Include heavier penalties than the BIC for score-based algorithms. In this way the number of edges would be reduced.

  It would be interesting to see if by following these procedures it would still be possible to recover the true edges and remove the ones not present in the original network.

- Secondly, a more in-depth simulation study could be conducted on the approximations made. Indeed, testing different networks in greater dimensions might be interesting. It would also be useful to check how the approximations work for networks involving double or even triple integrals for the logLikelihood computation. Two other paths to follow could be:
  - Study appropriate penalties for the score of PCBNs.
  - Try to find approaches to evaluate integrals that are less computationally demanding than the Monte Carlo methods.

More research is needed in this area.

[2] Different from the one implemented in the `GGMselect` package.

Bibliography

Aas, K., Czado, C., Frigessi, A., & Bakken, H. (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and economics*, *44*(2), 182–198. https://doi.org/https://doi.org/10.1016/j.insmatheco.2007.02.001

Bauer, A., & Czado, C. (2016). Pair-Copula Bayesian Networks. *Journal of Computational and Graphical Statistics*, *25*(4), 1248–1271. https://doi.org/10.1080/10618600.2015.1086355

Bauer, A., Czado, C., & Klein, T. (2012). Pair-copula constructions for non-Gaussian DAG models. *Canadian Journal of Statistics*, *40*(1), 86–109. https://doi.org/10.1002/cjs.10131

Bedford, T., & Cooke, R. M. (2002). Vines–a new graphical model for dependent random variables. *The Annals of Statistics*, *30*(4). https://doi.org/10.1214/aos/1031689016

Bouvier, A., Giraud, C., Hue, S., & Verzelen, N. (2022). *Ggmselect: Gaussian graphs models selection* [R package Version 0.1-12.5 — Depends R (>= 3.5.0)]. https://CRAN.R-project.org/package=GGMselect

Brechmann, E. C., & Schepsmeier, U. (2013). Modeling Dependence with C- and D-Vine Copulas: The *R* Package **CDVine**. *Journal of Statistical Software*, *52*(3). https://doi.org/10.18637/jss.v052.i03

Dißmann, J., Brechmann, E., Czado, C., & Kurowicka, D. (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, *59*, 52–69. https://doi.org/10.1016/j.csda.2012.08.010

Elidan, G. (2010). Copula bayesian networks. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2010/file/2a79ea27c279e471f4d180b08d62b00a-Paper.pdf

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, *9*(3), 432–441. https://doi.org/10.1093/biostatistics/kxm045

Giraud, C. (2008). Estimation of Gaussian graphs by model selection. *Electronic Journal of Statistics*, *2*(none). https://doi.org/10.1214/08-EJS228

Giraud, C., Huet, S., & Verzelen, N. (2012). Graph Selection with GGMselect. *Statistical Applications in Genetics and Molecular Biology*, *11*(3). https://doi.org/10.1515/1544-6115.1625

Haff, I. H. (2013). Parameter estimation for pair-copula constructions. *Bernoulli*, *19*(2), 462–491. https://doi.org/DOI:10.3150/12-BEJ413

Hammersley, J. (1964). *Monte carlo methods*. Springer Science & Business Media. https://link.springer.com/book/10.1007/978-94-009-5819-7

Hanea, A. M. (2011). Non-parametric bayesian belief nets versus vines. In D. Kurowicka & H. Joe (Eds.), *Dependence modeling: Vine copula handbook* (pages 281–305). World Scientific. https://doi.org/10.1142/9789814299886_0014

Hanea, A. M., Kurowicka, D., & Cooke, R. M. (2006). Hybrid method for quantifying and analyzing bayesian belief nets. *Quality and Reliability Engineering International*, *22*(6), 709–729. https://doi.org/10.1002/qre.808

Hanea, A., Napoles, O. M., & Ababei, D. (2015). Non-parametric bayesian networks: Improving theory and reviewing applications. *Reliability Engineering & System Safety*, *144*, 265–284. https://doi.org/https://doi.org/10.1016/j.ress.2015.07.027

Hobæk Haff, I., Aas, K., & Frigessi, A. (2010). On the simplified pair-copula construction — Simply useful or too simplistic? *Journal of Multivariate Analysis*, *101*(5), 1296–1310. https://doi.org/10.1016/j.jmva.2009.12.001

Hobæk Haff, I., Aas, K., Frigessi, A., & Lacal, V. (2016). Structure learning in Bayesian Networks using regular vines. *Computational Statistics & Data Analysis*, *101*, 186–208. https://doi.org/10.1016/j.csda.2016.03.003

Joe, H. (1997). *Multivariate Models and Multivariate Dependence Concepts* (Vol. 19970691). Chapman; Hall/CRC. https://doi.org/10.1201/b13150

Joe, H. (2014). *Dependence modeling with copulas*. CRC press.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

Korkmaz, S., Göksülük, D., & Zararsiz, G. (2014). Mvn: An r package for assessing multivariate normality. *R JOURNAL*, *6*(2). https://journal.r-project.org/archive/2014-2/korkmaz-goksuluk-zararsiz.pdf

Kurowicka, D. (2011). Optimal truncation of vines. In D. Kurowicka & H. Joe (Eds.), *Dependence modeling: Vine copula handbook* (pages 233–248). World Scientific. https://www.worldscientific.com/doi/abs/10.1142/9789814299886_0011

Kurowicka, D., & Cooke, R. (2006a). Completion problem with partial correlation vines. *Linear algebra and its applications*, *418*(1), 188–200. https://doi.org/https://doi.org/10.1016/j.laa.2006.01.031

Kurowicka, D., & Cooke, R. (2003). A parameterization of positive definite matrices in terms of partial correlation vines. *Linear Algebra and its Applications*, *372*, 225–251. https://doi.org/10.1016/S0024-3795(03)00507-X

Kurowicka, D., & Cooke, R. (2004). Distribution-free continuous bayesian belief nets. https://doi.org/10.1142/9789812703378_0022

Kurowicka, D., & Cooke, R. (2006b). *Uncertainty Analysis with High Dimensional Dependence Modelling: Kurowicka/Uncertainty Analysis with High Dimensional Dependence Modelling*. John Wiley & Sons, Ltd. https://doi.org/10.1002/0470863072

Lauritzen, S. L. (1996). *Graphical models* (Vol. 17). Clarendon Press.

Le Gall, J.-F. et al. (2016). *Brownian motion, martingales, and stochastic calculus* (Vol. 274). Springer.

Li, L., Wang, J., Leung, H., & Jiang, C. (2010). Assessment of catastrophic risk using bayesian network constructed from domain knowledge and spatial data. *Risk Analysis: An International Journal*, *30*(7), 1157–1175. https://doi.org/https://doi.org/10.1111/j.1539-6924.2010.01429.x

Marco Scutari, R. N., Tomi Silander. (2022). *Bnlearn: Bayesian network structure learning, parameter learning and inference* [R package Version 4.7.1 — Depends R (>= 3.3.0), methods]. https://cran.r-project.org/web/packages/bnlearn/bnlearn.pdf

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika*, *57*(3), 519–530. https://doi.org/10.1093/biomet/57.3.519

McNeil, A. J., Frey, R., & Embrechts, P. (2015). *Quantitative risk management: Concepts, techniques and tools-revised edition*. Princeton university press.

Meek, C. (2013). Causal inference and causal explanation with background knowledge. *CoRR*, *abs/1302.4972*. http://arxiv.org/abs/1302.4972

Meinshausen, N., & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The annals of statistics*, *34*(3), 1436–1462. https://doi.org/DOI:10.1214/009053606000000281

Morales Napoles, O., Cooke, R. M., & Kurowicka, D. (2010). About the number of vines and regular vines on n nodes. http://resolver.tudelft.nl/uuid:912abf55-8112-48d2-9cca-323f7f6aecc7

Müller, D., & Czado, C. (2018). Representing Sparse Gaussian DAGs as Sparse R-Vines Allowing for Non-Gaussian Dependence. *Journal of Computational and Graphical Statistics*, *27*(2), 334–344. https://doi.org/10.1080/10618600.2017.1366911

Müller, D., & Czado, C. (2019). Dependence modelling in ultra high dimensions with vine copulas and the Graphical Lasso. *Computational Statistics & Data Analysis*, *137*, 211–232. https://doi.org/10.1016/j.csda.2019.02.007

Nagler, T. (2021). *Vinecopula: Statistical inference of vine copulas* [R package version 2.4.3 — Depends R (>= 3.1.0)]. https://cran.r-project.org/web/packages/VineCopula/VineCopula.pdf

Narasimhan, B., Koller, M., Johnson, S. G., Hahn, T., Bouvier, A., Kiêu, K., Gaure, S., & Narasimhan, M. B. (2022). Package 'cubature'. http://cran.fhcrc.org/web/packages/cubature/cubature.pdf

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan kaufmann.

Pircalabelu, E., Claeskens, G., & Gijbels, I. (2017). Copula directed acyclic graphs. *Statistics and Computing*, *27*(1), 55–78. https://doi.org/10.1007/s11222-015-9599-9

Pourret, O., Na, P., Marcot, B., et al. (2008). *Bayesian networks: A practical guide to applications*. John Wiley & Sons.

Robinson, R. W. (1977). Counting unlabeled acyclic digraphs. In C. H. C. Little (Ed.), *Combinatorial mathematics v* (pp. 28–43). Springer Berlin Heidelberg.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., & Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, *308*(5721), 523–529. https://www.science.org/doi/epdf/10.1126/science.1105809

Schepsmeier, U. (2019). A goodness-of-fit test for regular vine copula models. *Econometric Reviews*, *38*(1), 25–46. https://doi.org/https://doi.org/10.1080/07474938.2016.1222231

Scutari, M., & Denis, J.-B. (2021). *Bayesian networks: With examples in r*. Chapman; Hall/CRC. https://doi.org/https://doi.org/10.1201/9780429347436

Scutari, M., Graafland, C. E., & Gutiérrez, J. M. (2019). Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, *115*, 235–253. https://doi.org/10.1016/j.ijar.2019.10.003

Sklar, A. (1959). Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, *8*, 229–231.

Spirtes, P., Glymour, C., & Scheines, R. (1993). *Causation, prediction, and search*. Springer.

Spirtes, P., Glymour, C. N., Scheines, R., & Heckerman, D. (2000). *Causation, prediction, and search*. MIT press.

Theodoridis, S. (2020). Chapter 15, 16 - probabilistic graphical models: Part i and ii. In S. Theodoridis (Ed.), *Machine learning (second edition)* (Second Edition, pp. 771–820). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-818803-3.00027-1

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Vepa, A., Saleem, A., Rakhshan, K., Daneshkhah, A., Sedighi, T., Shohaimi, S., Omar, A., Salari, N., Chatrabgoun, O., Dharmaraj, D., Sami, J., Parekh, S., Ibrahim, M., Raza, M., Kapila, P., & Chakrabarti, P. (2021). Using Machine Learning Algorithms to Develop a Clinical Decision-Making Tool for COVID-19 Inpatients. *International Journal of Environmental Research and Public Health*, *18*(12), 6228. https://doi.org/10.3390/ijerph18126228

Wang, W., & Wells, M. T. (2000). Model selection and semiparametric inference for bivariate failure-time data. *Journal of the American Statistical Association*, *95*(449), 62–72. https://doi.org/10.1080/01621459.2000.10473899

Zhu, K. (2022). Selections of vine structures and their applications. https://doi.org/https://doi.org/10.4233/uuid:84ba9f5e-2c5c-4280-9789-35fd650fc617

Zhu, K., & Kurowicka, D. (2022). Regular vines with strongly chordal pattern of (conditional) independence. *Computational Statistics & Data Analysis*, *172*, 107461. https://doi.org/10.1016/j.csda.2022.107461

A
PROOFS IN BNS

A.1. PROOF OF THEOREM 3.1

The proof of Theorem 3.1 is as follows:

- $(1 \Rightarrow 2)$ [Contradiction]

  Assume there is directed cycle and complete ordering. Since there is a directed cycle, then there exists a vertex v_{i_1} such that $v_{i_1} \to v_{i_2} \to \cdots \to v_{i_n} \to v_{i_1}$. Since there is complete ordering we get $v_{i_1} < v_{i_2} < \cdots < v_{i_n} < v_{i_1}$, which cannot happen because from complete ordering $v_{i_1} \nless v_{i_1}$.

- (2⇒1)

  If there is a complete ordering the vertices can be numbered such that $v_1 < v_2 < \cdots < v_n$ (not unique). Which means that we cannot follow arcs in the graph such that we start in one vertex and end up at this vertex again. Hence there is no directed cycle.

A.2. DERIVATION OF (3.4)

The derivation of (3.4) is as follows:

1. As we are dealing with a DAG, Theorem 3.1 states that there exists a complete ordering. We denote it as: $v_1 < v_2 < \cdots < v_n$, where parents are earlier in the ordering than children.

2. The joint probability density can be decomposed as the product of PDFs as follows:

$$f_{X_1,\dots,X_n}(x_1,\dots,x_n) = f_{X_n|X_1,\dots,X_{n-1}}(x_n|x_1,\dots,x_{n-1}) \cdot f_{X_1,\dots,X_{n-1}}(x_1,\dots,x_{n-1}).$$

3. Using the local Markov property and the fact that v_n is a leaf node, we have that:

$$f_{X_n|X_1,\dots,X_{n-1}}(x_n|x_1,\dots,x_{n-1}) = f_{X_n|\mathbf{X}_{pa(v_n)}}(x_n|\mathbf{x}_{pa(v_n)}).$$

4. Proceeding in the same manner:

$$f_{X_1,\dots,X_{n-1}}(x_1,\dots,x_{n-1}) = f_{X_{n-1}|X_1,\dots,X_{n-2}}(x_{n-1}|x_1,\dots,x_{n-2}) \cdot f_{X_1,\dots,X_{n-2}}(x_1,\dots,x_{n-2}).$$

5. Restricting to the subgraph $\mathscr{G}' = (\mathbb{V} \setminus v_n, E \setminus (v_i, v_n)_{v_i \in E})$, where v_{n-1} is a leaf node:

$$f_{X_{n-1}|X_1,\dots,X_{n-2}}(x_{n-1}|x_1,\dots,x_{n-2}) = f_{X_{n-1}|\mathbf{X}_{pa(v_{n-1})}}(x_{n-1}|\mathbf{x}_{pa(v_{n-1})}).$$

6. Using the complete order of the vertices and repeating the process recursively, the desired factorisation in (3.4) is obtained.

B

Equivalence class of a BN

B.1. Proof of Theorem 3.3

Let's assume that $X_i \perp\!\!\!\perp X_j | X_k \in \mathscr{I}(\mathscr{G}_1)$ and we are going to show that $X_i \perp\!\!\!\perp X_j | X_k \in \mathscr{I}(\mathscr{G}_2)$. If the two graphs have the same skeleton, then removing directionality, the trails between X_i and X_j are the same in both $\mathscr{G}_1$ and $\mathscr{G}_2$. Let's study some trail between X_i and X_j in $\mathscr{G}_1$. As $X_i \perp\!\!\!\perp X_j | X_k \in \mathscr{I}(\mathscr{G}_1)$, then X_k blocks the trail between X_i and X_j in $\mathscr{G}_1$. Consider two cases:

1. The trail in $\mathscr{G}_1$ is blocked because it contains X_k and its connection is not a v-structure. Then, the connection at X_k in $\mathscr{G}_2$, will also not be a v-structure. Consequently, the trail in $\mathscr{G}_2$ will be blocked by X_k too.

2. The trail in $\mathscr{G}_1$ does not contain X_k. Since this path is blocked by X_k, it contains a v-structure centred at $\rightarrow X_l \leftarrow$, such that X_k belongs neither to the v-structure ($X_k \neq X_l$), nor to the descendants of it ($X_k \notin de(X_l)$). Let us consider all the directed paths between X_l and its descendants. It is clear that X_k does not belong to any of these paths in $\mathscr{G}_1$. In $\mathscr{G}_2$ all these trails must also be directed the same, because otherwise a v-structure that is not in $\mathscr{G}_1$ would be introduced. Therefore, X_k is not presented in any of these trail in $\mathscr{G}_2$. Consequently, the trail in $\mathscr{G}_2$ will be blocked by X_k too.

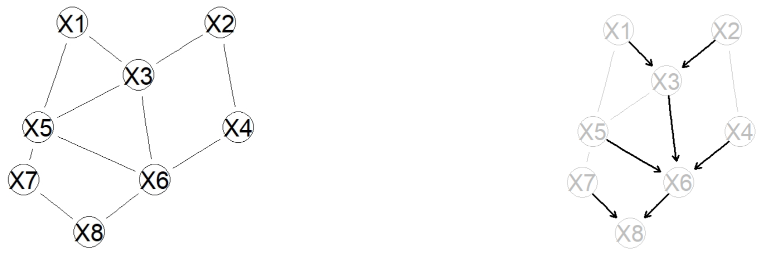
This result can be easily generalised for higher dimensional cases: if $\boldsymbol{X}_I \perp\!\!\!\perp \boldsymbol{X}_J | \boldsymbol{X}_K \in \mathscr{I}(\mathscr{G}_1)$ then $\boldsymbol{X}_I \perp\!\!\!\perp \boldsymbol{X}_J | \boldsymbol{X}_K \in \mathscr{I}(\mathscr{G}_1)$. Using the reverse order, we have that if $X_i \perp\!\!\!\perp X_j | X_k \in \mathscr{I}(\mathscr{G}_2)$ then $X_i \perp\!\!\!\perp X_j | X_k \in \mathscr{I}(\mathscr{G}_1)$. We therefore arrive to the desired result:

$$\mathscr{I}(\mathscr{G}_1) = \mathscr{I}(\mathscr{G}_2).$$

B.2. FINDING EQUIVALENCE CLASSES OF A BN

Example B.1 *The objective of this example is to find the equivalence class of the BN represented in Figure 3.5. The procedure is as follows:*

1. *The command `skel` of the `bnlearn` (Marco Scutari, 2022) package is used to obtain the skeleton. This can be seen in Figure B.1a.*
2. *The command `vstructs` of the `bnlearn` (Marco Scutari, 2022) package is used to get the vstructs. Including them in the skeleton graph, we get the semi-direct graph in Figure B.1b. Note that reversion any of these edges change the v-structures.*
3. *By directing the remaining indirect edges and avoiding the formation of new v-structures or cycles, we obtain the equivalence class of the DAG:*



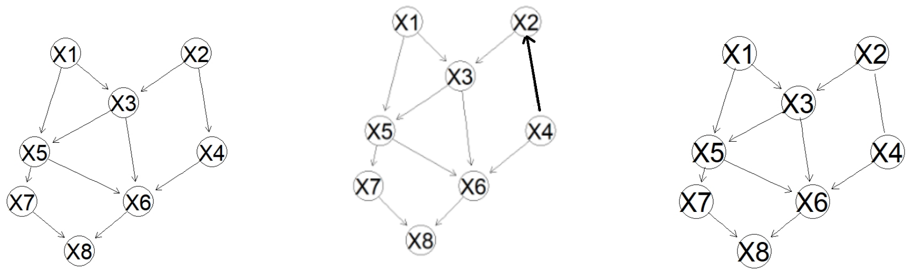
(a) Skeleton.

(b) v-structures.

Figure B.1: Skeleton and v-structures of our Bayesian Network.

- *If we introduce an arc $X_5 \to X_3$, then we get an v-struct $X_5 \to X_3 \leftarrow X_2$.*
  So we set $X_3 \to X_5$.
- *We then have to introduce an arc $X_1 \to X_5$.*
  Otherwise we would form a cycle: $X_1 \to X_3 \to X_5 \to X_1$.
- *If we introduce an arc $X_7 \to X_5$, then we get an v-struct $X_7 \to X_5 \leftarrow X_1$.*
  So we set $X_5 \to X_7$.
- *Hence, the only arc that we can reverse is $X_2 \to X_4$. Indeed, introducing $X_4 \to X_2$, we do not change the v-structures neither the skeleton.*

Therefore, there are only two DAGs in that equivalence class. The command `cpdag` of the `bnlearn` (Marco Scutari, 2022) package can be used to calculate and represent the equivalence class. This representation can be seen in Figure B.2. In many other cases, however, we could obtain equivalence classes with a large number of graphs.



(a) DAG 1 (b) DAG 2 (c) Equivalence Class representation.

Figure B.2: Equivalent DAGs.

C
Gaussian Bayesian Networks

C.1. Equivalence between GBNs and Multivariate Gaussian Distributions.

If $\mathscr{B}$ is a GBN $\Rightarrow$ It defines a joint distribution that is jointly Gaussian.

Let X_{n+1} be a continuous random variable with parents $\boldsymbol{X}^T = (X_1, \ldots, X_n)$, such that:

$$f_{X_{n+1}|X_1,\ldots,X_n}(x_{n+1}|x_1,\ldots,x_n) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T\boldsymbol{x}; \sigma^2\right),$$

where $\boldsymbol{\beta}^T = (\beta_1, \ldots, \beta_n)$ and $\boldsymbol{x}^T = (x_1, \ldots, x_n)$. Assume that $\boldsymbol{X}^T = (X_1, \ldots, X_n)$ are jointly Gaussian with distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We then have:

$$f_{X_1,\ldots,X_n,X_{n+1}}(x_1,\ldots,x_n,x_{n+1}) = \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu} \\ \beta_0 + \boldsymbol{\beta}^T\boldsymbol{\mu}\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma} & \boldsymbol{\Sigma}\boldsymbol{\beta} \\ \boldsymbol{\beta}^T\boldsymbol{\Sigma} & \sigma^2\end{pmatrix}\right). \tag{C.1}$$

Proof:

Let's first study the expectation, variance and covariance of this joint distribution. The properties of conditional expectation are used for this task. These can be seen for instance in Le Gall et al. (2016). We have:

$$\begin{aligned}
\mathbb{E}[X_{n+1}] &= \mathbb{E}[\mathbb{E}[X_{n+1}|X_1,\dots,X_n] = \mathbb{E}[\beta_0 + \boldsymbol{\beta}^T \boldsymbol{X}] = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{\mu} \\
\mathrm{Var}[X_{n+1}] &= \mathbb{E}[\mathrm{Var}[X_{n+1}|X_1,\dots,X_n]] + \mathrm{Var}[\mathbb{E}[X_{n+1}|X_1,\dots,X_n]] \\
&= \mathbb{E}[\sigma^2] + \mathrm{Var}[\beta_0 + \boldsymbol{\beta}^T \boldsymbol{X}] = \sigma^2 + \sum_{i,j=1}^{n} \beta_i \,\mathrm{Cov}(X_i, X_j)\beta_j = \sigma^2 + \boldsymbol{\beta}^T \Sigma \boldsymbol{\beta} \\
\mathrm{Cov}(X_i, X_{n+1}) &= \mathbb{E}[X_i X_{n+1}] - \mathbb{E}[X_i]\mathbb{E}[X_{n+1}] \\
&= \mathbb{E}[\mathbb{E}[X_i X_{n+1}|X_1,\dots,X_n]] - \mathbb{E}[X_i]\mathbb{E}[X_{n+1}] \qquad (C.2)\\
&= \mathbb{E}[X_i(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{X})] - \mathbb{E}[X_i]\mathbb{E}[X_{n+1}] \\
&= \beta_0\mathbb{E}[X_i] + \sum_{j=1}^{n} \beta_j \mathbb{E}[X_i X_j] - \beta_0\mathbb{E}[X_i] + E[X_i]\sum_{j=1}^{n} \beta_j \mathbb{E}[X_j] \\
&= \sum_{j=1}^{n} \beta_j \left(\mathbb{E}[X_i X_j] - \mathbb{E}[X_i]\mathbb{E}[X_j]\right) = \sum_{j=1}^{n} \Sigma_{i,j}\beta_j = \boldsymbol{\Sigma}_i \boldsymbol{\beta},
\end{aligned}$$

where $\boldsymbol{\Sigma}_i$ is the i-th row of the matrix $\boldsymbol{\Sigma}$.

We still need to prove that the joint distribution $(X_1 \dots, X_n, X_{n+1})$ is a Multivariate Gaussian Distribution. For this purpose, we slightly change the notation [1] to the one used by Kurowicka and Cooke (2006b). They show that GBNs can be seen as a set of regression equations:

$$X_i = \mu_i + \sum_{j \in pa(i)} b_{i,j}(X_j - \mu_j) + \sqrt{\boldsymbol{v}} Z_j, \; i = 1,\dots,n+1, \qquad (C.3)$$

where it is assumed without loss of generality that the sequence of indices is ordered, so that the matrix $\boldsymbol{B} = [b_{i,j}]$ is strictly upper triangular. $\boldsymbol{Z} = (Z_1,\dots,Z_n,Z_{n+1})$ are independent standard Gaussian variables and $\boldsymbol{v} = (v_1,\dots,v_{n+1})$ is a vector of conditional variances. This can be rewritten as:

$$\boldsymbol{X} - \boldsymbol{\mu} = \boldsymbol{B}^T(\boldsymbol{X} - \boldsymbol{\mu}) + \boldsymbol{S}^T \boldsymbol{Z},$$

where $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{v}) = \boldsymbol{S}^T \boldsymbol{S}$. This can also be expressed as:

$$\boldsymbol{S}^T \boldsymbol{Z} = (\boldsymbol{I} - \boldsymbol{B}^T)(\boldsymbol{X} - \boldsymbol{\mu}).$$

Since $\boldsymbol{B}$ is strictly upper triangular then $(\boldsymbol{I} - \boldsymbol{B}^T)$ is invertible and

$$\boldsymbol{X} - \boldsymbol{\mu} = (\boldsymbol{I} - \boldsymbol{B}^T)^{-1} \boldsymbol{S}^T \boldsymbol{Z} = \boldsymbol{U}^T \boldsymbol{S}^T \boldsymbol{Z} = \boldsymbol{A}^T \boldsymbol{Z},$$

where $\boldsymbol{U} = (\boldsymbol{I} - \boldsymbol{B})$ and $\boldsymbol{A} = \boldsymbol{S}\boldsymbol{U}$, we finally get

$$\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{A}^T \boldsymbol{Z},$$

which is precisely the definition of a Multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{A}^T \boldsymbol{A}$. This concludes the proof.

[1] Working with the inverse of the covariance matrix in (C.1) is extremely cumbersome.

Conditioning joint Gaussian distributions ⇒ produces a GBN.

Let $(\boldsymbol{X}, X_{n+1})$, with $\boldsymbol{X} = (X_1, \dots, X_n)$, be multivariate Gaussian distribution distributed as:

$$f_{X_1,\dots,X_n,X_{n+1}}(x_1,\dots,x_n,x_{n+1}) = \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}\\ \mu_{n+1}\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}} & \boldsymbol{\Sigma}_{\boldsymbol{X},X_{n+1}}\\ \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}} & \Sigma_{X_{n+1},X_{n+1}}\end{pmatrix}\right). \tag{C.4}$$

Then the conditional density

$$f_{X_{n+1}|X_1,\dots,X_n}(x_{n+1}|x_1,\dots,x_n) = \mathcal{N}\left(\beta_0 + \boldsymbol{\beta}^T\boldsymbol{x}; \sigma^2\right),$$

is such that:

- $\beta_0 = \mu_{n+1} - \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\mu}$,
- $\boldsymbol{\beta} = \boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},X_{n+1}}$,
- $\sigma^2 = \Sigma_{X_{n+1},X_{n+1}} - \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},X_{n+1}}$.

Proof:

All conditional distributions of a multivariate Gaussian distribution are also Gaussian, we therefore only need to prove the previous equations.

Let's define the variable $Z = X_{n+1} + \boldsymbol{A}\boldsymbol{X}$, where $\boldsymbol{A} = -\boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}$. We then have:

$$\mathrm{Cov}(Z,\boldsymbol{X}) = \mathrm{Cov}(X_{n+1},\boldsymbol{X}) + \boldsymbol{A}\,\mathrm{Cov}(\boldsymbol{X},\boldsymbol{X}) = \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}} - \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}} = 0. \tag{C.5}$$

Hence Z and $\boldsymbol{X}$ are uncorrelated and, since they are Gaussian, they are independent. Using this fact, we then have:

$$\begin{aligned}\beta_0 + \boldsymbol{\beta}^T\boldsymbol{X} = \mathbb{E}[X_{n+1}|\boldsymbol{X}] = \mathbb{E}[Z - \boldsymbol{A}\boldsymbol{X}|\boldsymbol{X}] = \mathbb{E}[Z] - \boldsymbol{A}\boldsymbol{X} = \mu_{n+1} + \boldsymbol{A}\left(\boldsymbol{\mu} - \boldsymbol{X}\right)\\ = \mu_{n+1} + \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}(\boldsymbol{X} - \boldsymbol{\mu}),\end{aligned} \tag{C.6}$$

which proves the first 2 points. For the covariance matrix, note that:

$$\begin{aligned}\mathrm{Var}[X_{n+1}|\boldsymbol{X}] &= \mathrm{Var}[Z - \boldsymbol{A}\boldsymbol{X}|\boldsymbol{X}]\\ &= \mathrm{Var}[Z|\boldsymbol{X}] + \mathrm{Var}[\boldsymbol{A}\boldsymbol{X}|\boldsymbol{X}] - \boldsymbol{A}\,\mathrm{Cov}[Z,\boldsymbol{X}|\boldsymbol{X}] - \mathrm{Cov}[Z,\boldsymbol{X}|\boldsymbol{X}]\boldsymbol{A}^T\\ &= \mathrm{Var}[Z]\end{aligned} \tag{C.7}$$

$$\begin{aligned}\mathrm{Var}[Z] = \mathrm{Var}[X_{n+1} + \boldsymbol{A}\boldsymbol{X}] &= \mathrm{Var}[X_{n+1}] + \boldsymbol{A}\,\mathrm{Var}[\boldsymbol{X}]\boldsymbol{A}^T + \boldsymbol{A}\,\mathrm{Cov}[\boldsymbol{X}, X_{n+1}] + \mathrm{Cov}[X_{n+1},\boldsymbol{X}]\boldsymbol{A}^T\\ &= \mathrm{Var}[X_{n+1}] + \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},X_{n+1}}\\ &- \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},X_{n+1}} - \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},X_{n+1}}\\ &= \Sigma_{X_{n+1},X_{n+1}} - \boldsymbol{\Sigma}_{X_{n+1},\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X},\boldsymbol{X}}{}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{X},X_{n+1}},\end{aligned} \tag{C.8}$$

which proves the last point

These previous results demonstrate the equivalence between GBNs and multivariate Gaussian distributions. Conversely, while the two representations are equivalent in their expressive power, there is not a one-to-one correspondence between their parameterizations (Koller and Friedman, 2009).

C.2. INDEPENDENCIES AND CONDITIONAL INDEPENDENCIES IN GAUSSIAN DISTRIBUTIONS

X_i and X_j, joint gaussian distribution, are independent $\iff \Sigma_{i,j} = 0$.
proof:

- $(\Rightarrow)$

$$\Sigma_{i,j} = \mathbb{E}[X_i X_j] - \mathbb{E}[X_i]\mathbb{E}[X_j].$$

As the variables are independent then

$$\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j].$$

Hence we get $\Sigma_{i,j} = 0$.

- $(\Leftarrow)$

The characteristic function of (X_i, X_j) can be computed as:

$$\varphi_{X_i,X_j}(t_i,t_j) = e^{it_i\mu_i + it_j\mu_j - \frac{1}{2}\left(t_i^2\Sigma_{1,1}^2 + 2t_it_j\Sigma_{i,j} + t_j^2\Sigma_{j,j}^2\right)}$$

If $\Sigma_{i,j} = 0$, then the previous expression can be factorised as:

$$\varphi_{X_i,X_j}(t_i,t_j) = e^{it_i\mu_i - \frac{1}{2}t_i^2\Sigma_{1,1}^2} \cdot e^{it_j\mu_j - \frac{1}{2}t_j^2\Sigma_{j,j}^2} = \varphi_{X_i}(t_i)\cdot\varphi_{X_j}(t_j)$$

characteristic function of a pair of random variables is equal to the product of their characteristic functions, the random variables are independent, which concludes the proof.

Counterexample: 0-correlation does not imply independence outside the multivariate Gaussian distribution.

Take a random variable X with $E[X] = 0$ and $E[X^3] = 0$, e.g. Gaussian random variable with zero mean. Take $Y = X^2$. It is clear that X and Y are related, but:

$$Cov(X,Y) = E[X\cdot Y] - \underbrace{E[X]}_{0}\cdot E[Y] = E[X^3] = 0.$$

X_i and X_j are independent given the remaining variables $\iff \Omega_{i,j} = 0$.

We are going to show the ideas behind this previous result. Let's split the multivariate random vector $\boldsymbol{X} = (X_1,\ldots,X_n)$ into two components: $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2)$, where $\boldsymbol{X}_1 = (X_i, X_j)$ and $\boldsymbol{X}_2 = (\boldsymbol{X}\backslash\{X_i, X_j\})$ are respectively two and $n-2$ dimensional random vectors.

If the joint distribution of $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2)$ is:

$$f_{\boldsymbol{X}_1,\boldsymbol{X}_2}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2}\\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2}\end{pmatrix}\right),$$

then the conditional distribution $\boldsymbol{X}_1|\boldsymbol{X}_2$ is given by:

$$f_{\boldsymbol{X}_1|\boldsymbol{X}_2}(\boldsymbol{x}_1|\boldsymbol{x}_2) = \mathcal{N}\left(\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{1,1} - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\right)$$

This above statement can be demonstrated using the same reasoning we used to show that joint Gaussian distribution conditioning leads to GBNs. Notice that this distribution is bidimensional, so we denote its precision matrix and mean vector as:

$$\left(\boldsymbol{\Sigma}_{1,1} - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\right)^{-1} = \boldsymbol{K} = \begin{pmatrix} k_{i,i} & k_{i,j} \\ k_{j,i} & k_{j,j} \end{pmatrix}$$

$$\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) = \boldsymbol{\mu} = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$$

On the other hand, the inverse of the covariance matrix of $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2)$ is:

$$\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2} \\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2} \end{pmatrix}^{-1} =$$

$$\begin{pmatrix} \left(\boldsymbol{\Sigma}_{1,1} - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\right)^{-1} & -\left(\boldsymbol{\Sigma}_{1,1} - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\right)^{-1}\boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1} \\ -\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\left(\boldsymbol{\Sigma}_{1,1} - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\right)^{-1} & \boldsymbol{\Sigma}_{2,2}^{-1} + \boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\left(\boldsymbol{\Sigma}_{1,1} - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\right)^{-1}\boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1} \end{pmatrix}$$

Therefore, the first block of the precision matrix $\boldsymbol{\Omega}$ of $(\boldsymbol{X}_1, \boldsymbol{X}_2)$, coincides with the precision matrix $\boldsymbol{K}$ of the distribution $\boldsymbol{X}_1|\boldsymbol{X}_2$. Then it follows that $\boldsymbol{\Sigma}_{i,j}^{-1} = k_{i,j}$.

If $\boldsymbol{\Sigma}_{i,j}^{-1} = k_{i,j} = 0$, we can then factorise the conditional probability as:

$$\begin{aligned} f_{X_i,X_j|\boldsymbol{X}_2}(x_i, x_j|\boldsymbol{X}_2) &\propto exp\left(\frac{-1}{2}(x_i - \mu_i, x_j - \mu_j)\begin{pmatrix} k_{i,i} & 0 \\ 0 & k_{j,j} \end{pmatrix}\begin{pmatrix} x_i - \mu_i \\ x_j - \mu_j \end{pmatrix}\right) \\ &\propto \underbrace{\exp\left(\frac{-1}{2}(x_i - \mu_i)k_{i,i}(x_i - \mu_i)\right)}_{\propto f_{X_i|\boldsymbol{X}_2}(x_i|\boldsymbol{x}_2)} \cdot \underbrace{\exp\left(\frac{-1}{2}(x_j - \mu_j)k_{j,j}(x_j - \mu_j)\right)}_{\propto f_{X_j|\boldsymbol{X}_2}(x_j|\boldsymbol{x}_2)}, \end{aligned} \tag{C.9}$$

where μ_i and μ_2 depends on $\boldsymbol{x}_2$. Therefore we have that:

$$f_{X_i,X_j|\boldsymbol{X}_2}(x_i, x_j|\boldsymbol{X}_2) = f_{X_i|\boldsymbol{X}_2}(x_i|\boldsymbol{X}_2) \cdot f_{X_j|\boldsymbol{X}_2}(x_j|\boldsymbol{X}_2), \tag{C.10}$$

which shows the desired result: $X_i \perp\!\!\!\perp X_j | \boldsymbol{X} \backslash \{X_i, X_j\}$.

This previous result can be extended to:

X_i and X_j are conditionally independent given $X_{k_1}, \ldots, X_{k_n} \iff \tilde{\Sigma}_{i,j}^{-1} = 0$, where $\tilde{\Sigma}$ denote the covariance sub-matrix formed by: $X_i, X_j, X_{k_1}, \ldots, X_{k_n}$.

These results follow from the above reasoning plus the fact that the marginal distribution $(X_i, X_j, X_{k_1}, \ldots, X_{k_n})$ is multivariate Gaussian with mean and covariance matrix resulting from eliminating the irrelevant variables (the variables to be marginalized) from the vector of means and the covariance matrix of $\boldsymbol{X}$.

C.3. COMPUTATIONS IN GBNS

Computing the joint distribution (X_2, X_3) of the GBN given by Figure 4.1.

To do so, we compute the required means and covariance.

$$\mathbb{E}[X_2] = \mathbb{E}[\mathbb{E}[X_2|X_1]] = \mathbb{E}[\mu_2 + \beta_1 X_1] = \mu_2 + \beta_1\mu_1.$$

$$\mathbb{V}[X_2] = \mathbb{E}[\mathbb{V}[X_2|X_1]] + \mathbb{V}[\mathbb{E}[X_2|X_1]] = \mathbb{E}[\sigma_2^2] + \mathbb{V}[\mu_2 + \beta_1 X_1] = \sigma_2^2 + \beta_1^2\sigma_1^2.$$

In a similar way, we obtain:

$$\mathbb{E}[X_3] = \mu_3 + \beta_2\mu_1 \ , \ \mathbb{V}[X_3] = \sigma_3^2 + \beta_2^2\sigma_1^2.$$

Lastly, we compute the crossed term $\Sigma_{2,3}$:

$$\begin{aligned} \mathbb{COV}[X_2, X_3] &= \mathbb{E}[X_2 \cdot X_3] - \mathbb{E}[X_2]\cdot\mathbb{E}[X_3] = \mathbb{E}[\mathbb{E}[X_2 \cdot X_3|X_1]] - \mathbb{E}[X_2]\cdot\mathbb{E}[X_3] \\ &= \mathbb{E}[\mathbb{E}[X_2|X_1]\cdot\mathbb{E}[X_3|X_1]] - \mathbb{E}[X_2]\cdot\mathbb{E}[X_3] \\ &= \mathbb{E}[(\mu_2 + \beta_1 X_1)\cdot(\mu_3 + \beta_2 X_1)] - (\mu_2 + \beta_1\mu_1)\cdot(\mu_3 + \beta_2\mu_1) \\ &= \beta_1\beta_2\left(\mathbb{E}[X_1^2] - \mu_1^2\right) = \beta_1\beta_2\left(\mathbb{E}[X_1^2] - \mathbb{E}[X_1]^2\right) = \beta_1\beta_2\sigma_1^2. \end{aligned} \tag{C.11}$$

Therefore the distribution of (X_2, X_3) is given by:

$$(X_2, X_3) \sim N\left(\begin{pmatrix}\mu_2 + \beta_1\mu_1 \\ \mu_3 + \beta_2\mu_1\end{pmatrix}, \begin{pmatrix}\sigma_2^2 + \beta_1^2\sigma_1^2 & \beta_1\beta_2\sigma_1^2 \\ \beta_1\beta_2\sigma_1^2 & \sigma_3^2 + \beta_2^2\sigma_1^2\end{pmatrix}\right). \tag{C.12}$$

Independencies and conditional independencies presented in the GBN given by Figure 4.1 using the theoretical covariance matrix.

Using the `rbmn` package[2], the mean vector, the covariance matrix and the correlation matrix are obtained:

$$\mu = \begin{pmatrix}1\\2\\2\\5\end{pmatrix}, \quad \mathbf{\Sigma} = \begin{pmatrix}1 & 1 & 1 & 2\\ 1 & 2 & 1 & 3\\ 1 & 1 & 2 & 3\\ 2 & 3 & 3 & 7\end{pmatrix}, \quad \tilde{\mathbf{\Sigma}} = \begin{pmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 2\end{pmatrix}$$

Computing the inverse we have that:

$$\mathbf{\Sigma}^{-1} = \begin{pmatrix}3 & -1 & -1 & 0\\ -1 & 2 & 1 & -1\\ -1 & 1 & 2 & -1\\ 0 & -1 & -1 & 1\end{pmatrix}, \quad \tilde{\mathbf{\Sigma}}^{-1} = \begin{pmatrix}3 & -1 & -1\\ -1 & 1 & 0\\ -1 & 0 & 1\end{pmatrix}$$

Therefore:

$$\Sigma_{1,4}^{-1} = 0 \Rightarrow X_1 \perp\!\!\!\perp X_4 | X_2, X_3,$$

$$\tilde{\Sigma}_{2,3}^{-1} = 0 \Rightarrow X_2 \perp\!\!\!\perp X_3 | X_1.$$

[2] https://cran.r-project.org/web/packages/rbmn/rbmn.pdf

D

Results Simulation Study Chapter 6

Below we show all the results obtained in the simulation studied carried out in Chapter 6.

Table D.1: Simulation results for $N = 500$, small network.

		Gaussian			Non-Gaussian		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	0.384	0.000	2.769	0.683	0.031	2.505
Constraint	SHD	2.133	0.033	2.905	2.200	0.067	3.303
	Hamming	1.033	0.033	2.602	1.050	0.065	2.500
Score	SHD	0.984	0.000	2.600	1.183	0.098	2.870
	Hamming	0.583	0.000	2.600	0.716	0.031	2.435

Table D.2: Simulation results for $N = 2500$, small network.

		Gaussian			Non-Gaussian		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	0.067	0.000	1.635	0.666	0.033	1.833
Constraint	SHD	1.433	0.033	4.867	1.534	0.065	4.669
	Hamming	0.967	0.033	2.135	1.016	0.065	2.279
Score	SHD	0.167	0.000	2.203	0.817	0.000	2.447
	Hamming	0.067	0.000	1.441	0.600	0.000	1.935

Table D.3: Simulation results for $N = 7500$, small network.

		Gaussian			Non-Gaussian		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	0.000	0	1.072	0.933	0.000	1.646
Constraint	SHD	1.083	0	4.835	1.000	0.065	4.436
	Hamming	0.933	0	1.502	1.000	0.065	1.870
Score	SHD	0.000	0	1.067	0.867	0.000	2.894
	Hamming	0.000	0	1.067	0.867	0.000	1.743

Table D.4: Simulation results for $N = 500$, big network.

		Gaussian			Non-Gaussian		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	1.667	0.000	2.962	2.200	0.000	5.950
Constraint	SHD	1.134	0.200	1.723	1.816	0.520	4.228
	Hamming	0.733	0.143	1.290	1.200	0.438	4.090
Score	SHD	1.917	1.005	2.923	2.900	1.600	8.642
	Hamming	0.734	0.433	1.228	1.417	0.572	5.708

Table D.5: Simulation results for $N = 2500$, big network.

		Gaussian			Non-Gaussian		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	0.133	0.000	0.428	1.217	0.005	9.493
Constraint	SHD	0.400	0.133	0.733	2.417	0.372	8.358
	Hamming	0.267	0.038	0.585	1.750	0.238	7.400
Score	SHD	0.433	0.167	0.985	3.866	0.477	13.452
	Hamming	0.200	0.100	0.390	2.117	0.205	10.432

Table D.6: Simulation results for $N = 7500$, big network.

		Gaussian			Non-Gaussian		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
G.Lasso	Hamming	0.000	0.000	0.062	1.917	0.000	16.277
Constraint	SHD	0.367	0.100	0.862	4.934	0.467	13.100
	Hamming	0.300	0.100	0.695	2.716	0.353	12.483
Score	SHD	0.300	0.105	0.713	5.900	0.310	20.608
	Hamming	0.133	0.067	0.267	3.966	0.133	16.567

Table D.7: Number of edges, for the small network with 5 edges.

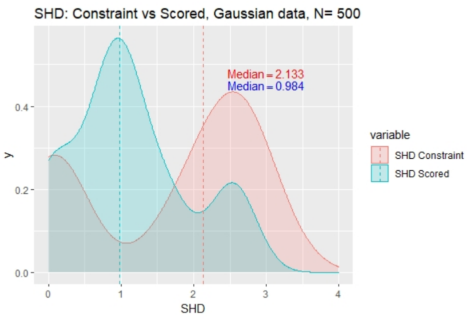
		Gaussian			**Non-Gaussian**		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
N=500	G.Lasso	4.916	2.365	5.033	4.666	2.633	5.570
	Constraint	4.000	2.698	5.035	4.000	2.960	5.240
	Score	4.867	3.000	5.002	4.800	3.098	5.238
N=2500	G.Lasso	5.000	3.800	5.033	5.050	3.995	6.000
	Constraint	4.067	3.233	5.100	4.883	3.233	5.935
	Score	5.000	4.297	5.002	5.033	4.492	5.741
N=7500	G.Lasso	5.000	4.000	5.033	5.300	4.164	6.000
	Constraint	4.167	3.698	5.100	5.167	3.930	6.000
	Score	5.000	5.000	5.000	5.100	5.000	6.000
N=15000	G.Lasso	5.000	4.000	5.033	5.550	4.098	6.000
	Constraint	4.200	4.000	5.100	5.434	4.130	6.000
	Score	5.000	5.000	5.000	5.284	5.000	6.000

Table D.8: Number of edges, for the big network with 9 edges.

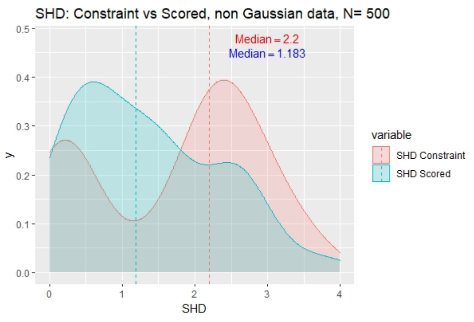
		Gaussian			**Non-Gaussian**		
		Median	$Q_{0.05}$	$Q_{0.95}$	Median	$Q_{0.05}$	$Q_{0.95}$
N=500	G.Lasso	9.166	8.343	10.295	10.000	8.567	12.952
	Constraint	8.850	8.467	9.400	9.467	8.533	11.385
	Score	9.500	9.272	9.723	10.100	9.433	14.335
N=2500	G.Lasso	9.084	9.000	9.390	10.133	9.000	18.183
	Constraint	9.233	9.033	9.557	10.650	9.233	16.323
	Score	9.200	9.100	9.390	11.116	9.205	19.355
N=7500	G.Lasso	9.000	9.000	9.062	10.917	9.000	25.277
	Constraint	9.300	9.100	9.695	11.716	9.353	21.483
	Score	9.133	9.067	9.267	12.966	9.133	25.567
N=15000	G.Lasso	9.000	9.000	9.033	12.733	9.000	29.995
	Constraint	9.267	9.067	9.662	12.933	9.267	24.953
	Score	9.067	9.000	9.167	14.550	9.100	28.500

Table D.9: Simulation results: median Running Times.

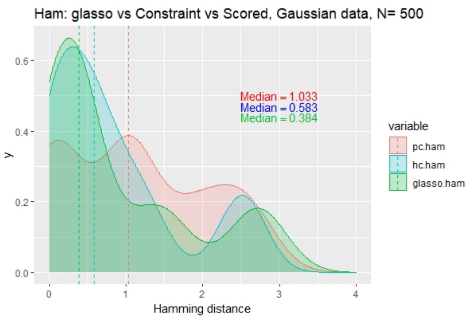
		Small Network		Big Network	
		Gaussian	**Non-Gaussian**	**Gaussian**	**Non-Gaussian**
N=500	G.Lasso	0.432	0.462	0.481	0.433
	Constraint	0.007	0.006	0.017	0.024
	Score	0.015	0.016	0.030	0.035
N=2500	G.Lasso	0.445	0.424	0.530	0.456
	Constraint	0.007	0.006	0.017	0.024
	Score	0.038	0.037	0.103	0.122
N=7500	G.Lasso	0.487	0.420	0.601	0.510
	Constraint	0.011	0.010	0.030	0.068
	Score	0.096	0.084	0.332	0.352
N=15000	G.Lasso	0.510	0.481	0.685	0.586
	Constraint	0.017	0.018	0.082	0.228
	Score	0.178	0.178	0.649	0.726



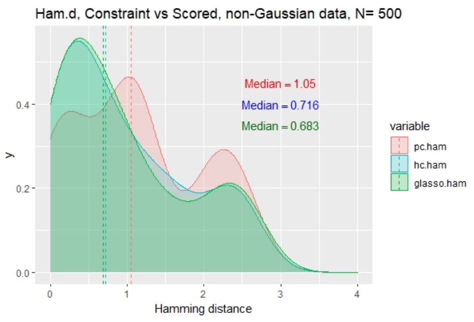
(a) Density plots SHD Gaussian case.



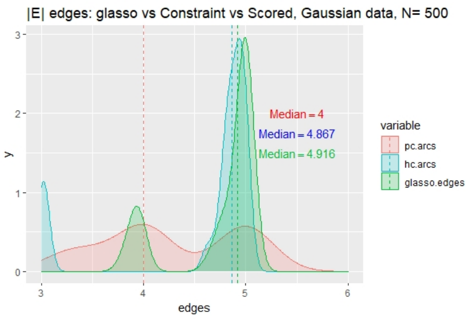
(b) Density plot SHD non-Gaussian case.



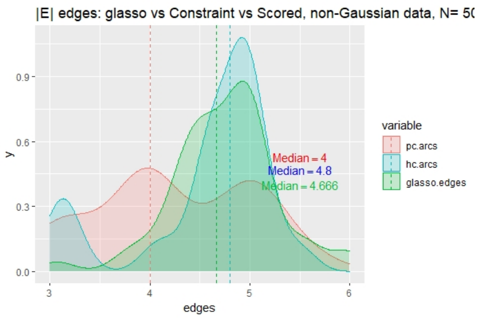
(c) Density plots Hamming distance Gaussian case.



(d) Density plots Hamming distance non-Gaussian case.

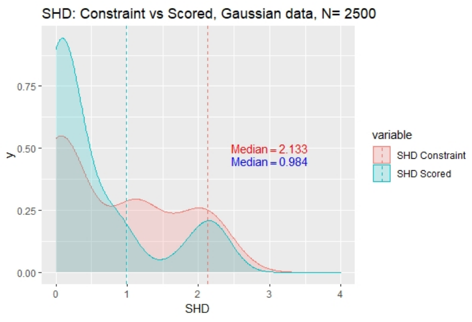


(e) Density plots number of edges Gaussian case.

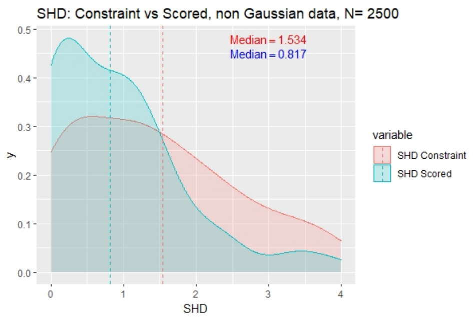


(f) Density plots number of edges non-Gaussian case.

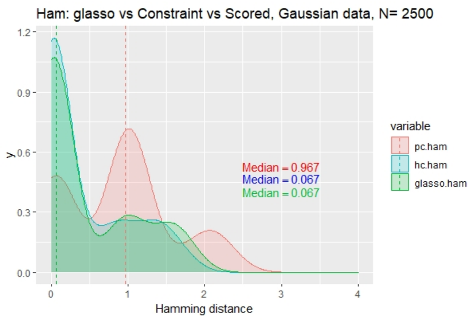
Figure D.1: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 500$ generated from the small network



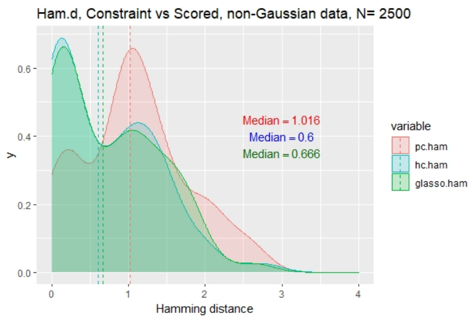
(a) Density plots SHD Gaussian case.



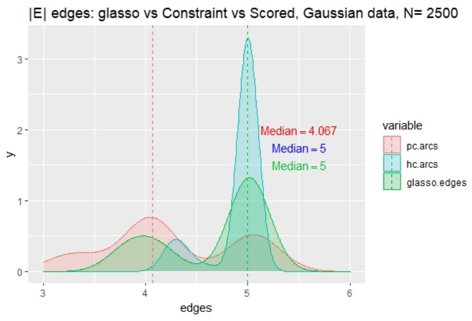
(b) Density plot SHD non-Gaussian case.



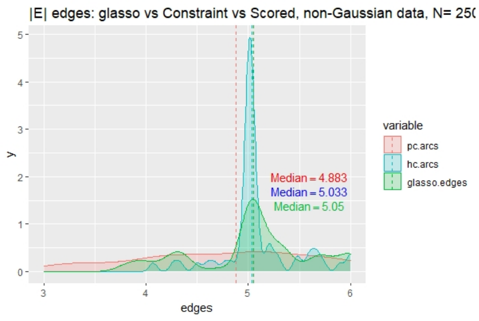
(c) Density plots Hamming distance Gaussian case.



(d) Density plots Hamming distance non-Gaussian case.

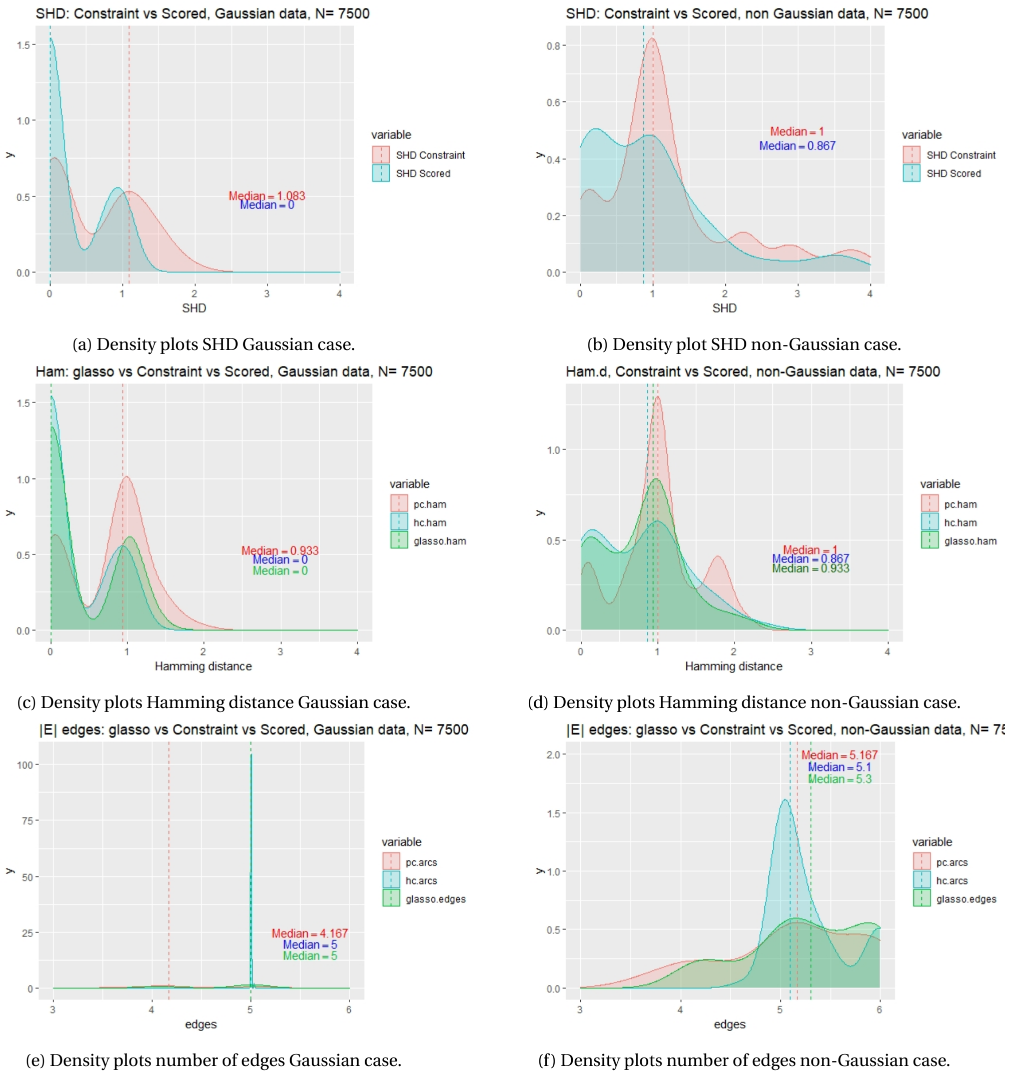


(e) Density plots number of edges Gaussian case.



(f) Density plots number of edges non-Gaussian case.

Figure D.2: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 2500$ generated from the small network



(a) Density plots SHD Gaussian case.

(b) Density plot SHD non-Gaussian case.

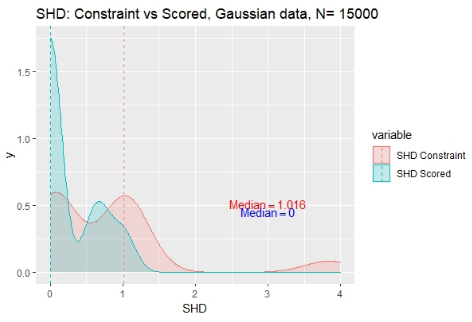
(c) Density plots Hamming distance Gaussian case.

(d) Density plots Hamming distance non-Gaussian case.

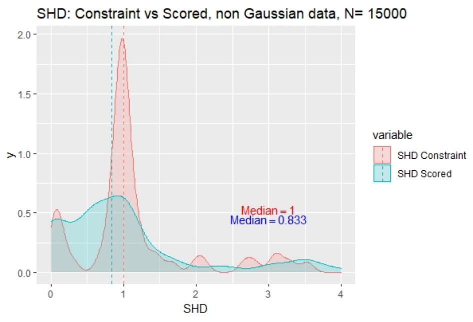
(e) Density plots number of edges Gaussian case.

(f) Density plots number of edges non-Gaussian case.

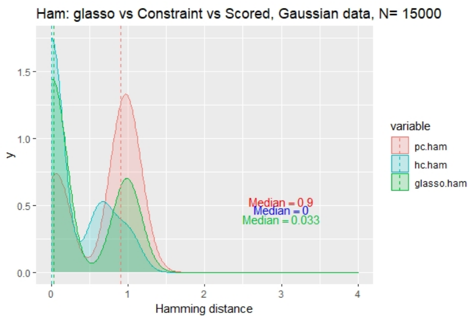
Figure D.3: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 7500$ generated from the small network



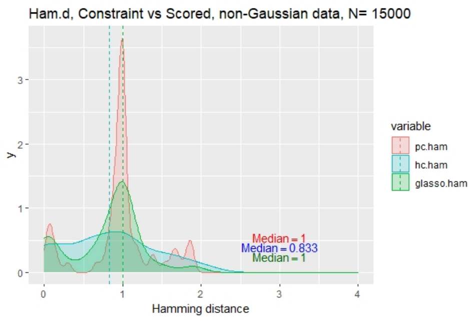
(a) Density plots SHD Gaussian case.



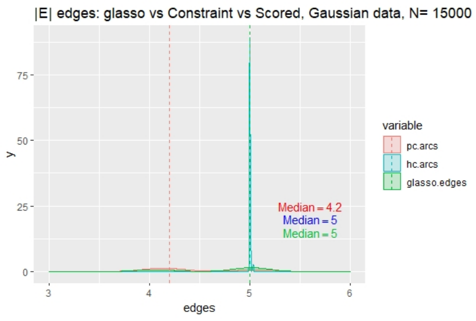
(b) Density plot SHD non-Gaussian case.



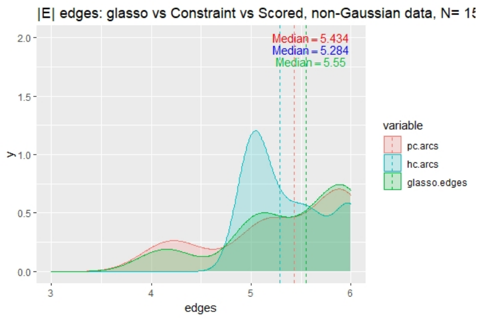
(c) Density plots Hamming distance Gaussian case.



(d) Density plots Hamming distance non-Gaussian case.

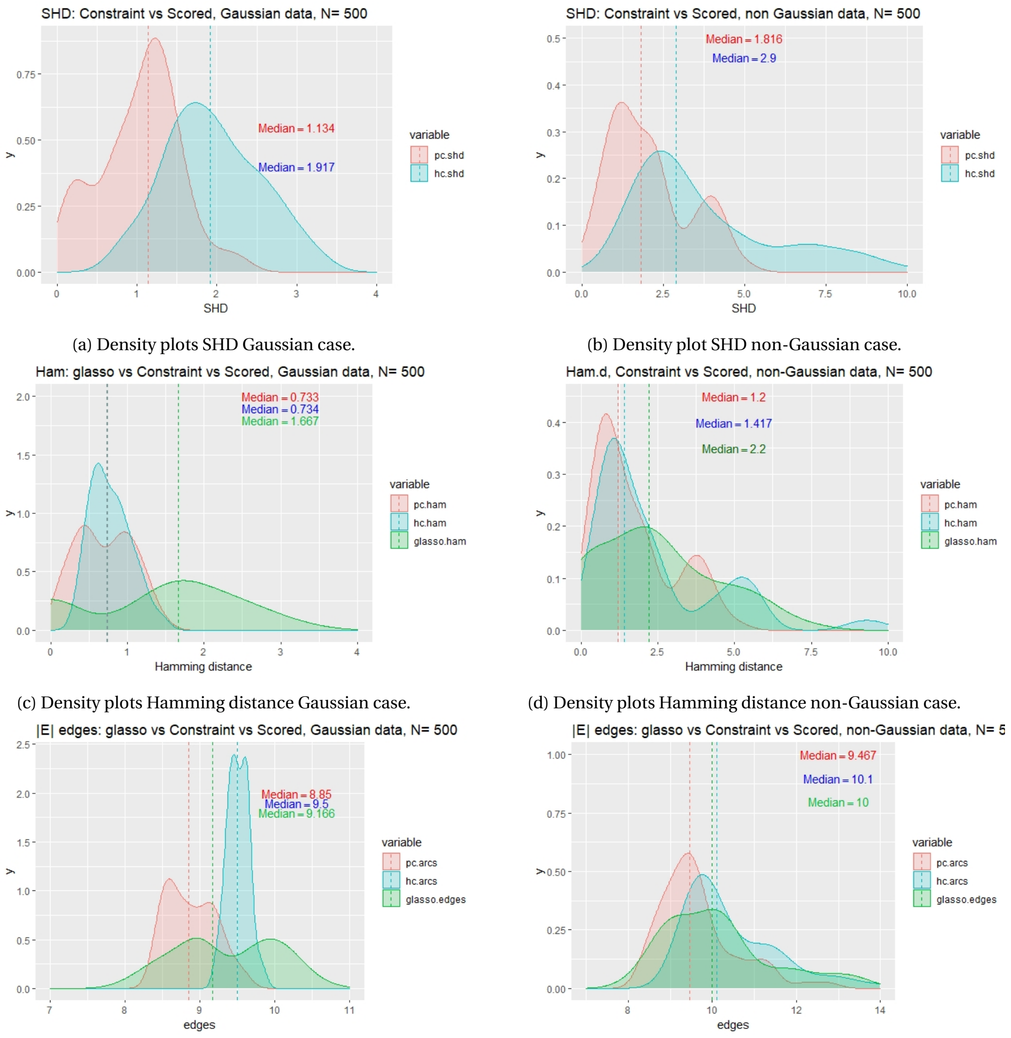


(e) Density plots number of edges Gaussian case.



(f) Density plots number of edges non-Gaussian case.

Figure D.4: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 15000$ generated from the small network



(a) Density plots SHD Gaussian case.

(b) Density plot SHD non-Gaussian case.

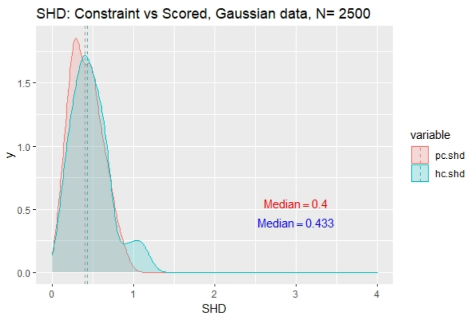
(c) Density plots Hamming distance Gaussian case.

(d) Density plots Hamming distance non-Gaussian case.

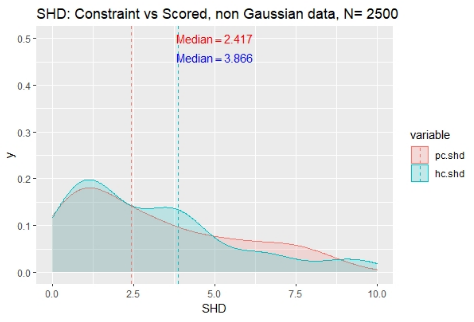
(e) Density plots number of edges Gaussian case.

(f) Density plots number of edges non-Gaussian case.

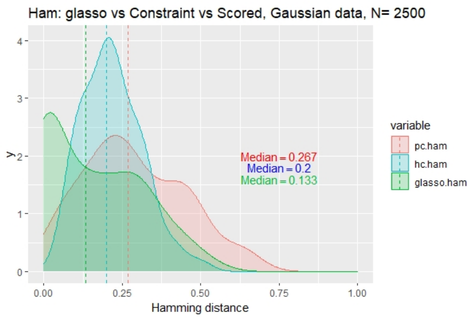
Figure D.5: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 500$ generated from the big network



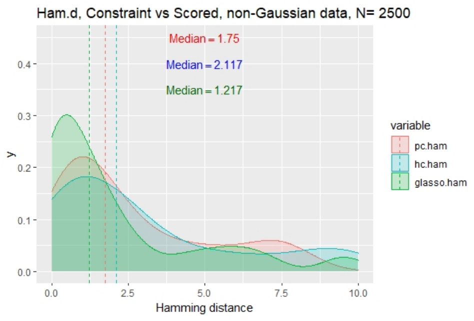
(a) Density plots SHD Gaussian case.



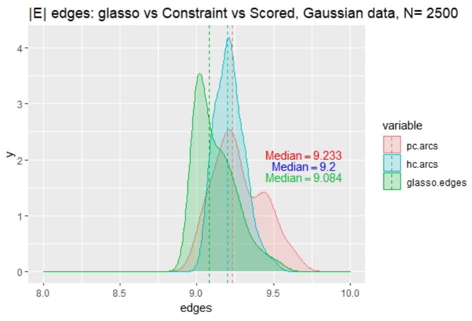
(b) Density plot SHD non-Gaussian case.



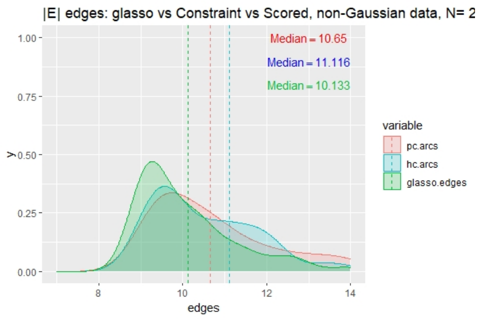
(c) Density plots Hamming distance Gaussian case.



(d) Density plots Hamming distance non-Gaussian case.

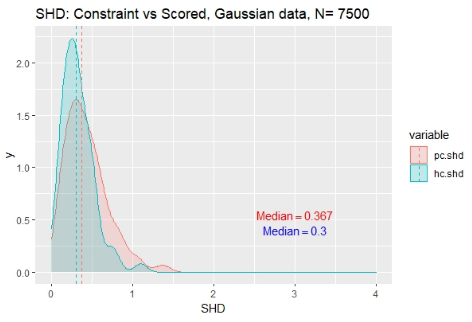


(e) Density plots number of edges Gaussian case.

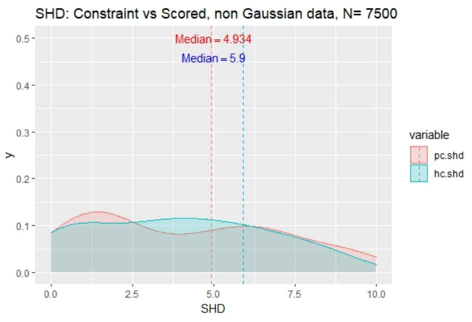


(f) Density plots number of edges non-Gaussian case.

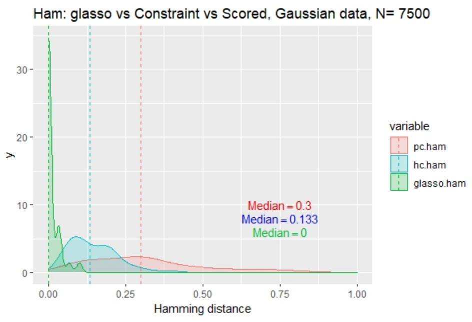
Figure D.6: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 2500$ generated from the big network



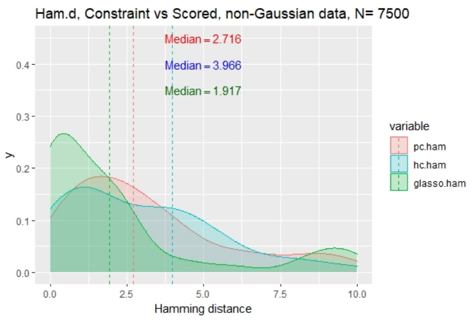
(a) Density plots SHD Gaussian case.



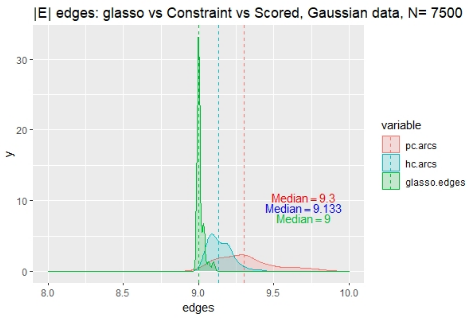
(b) Density plot SHD non-Gaussian case.



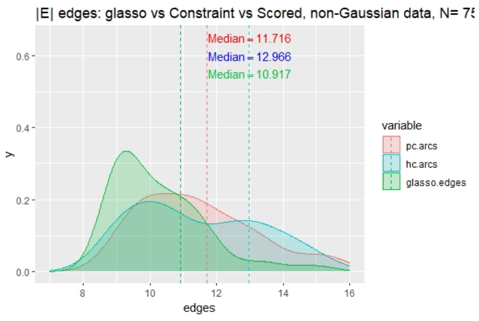
(c) Density plots Hamming distance Gaussian case.



(d) Density plots Hamming distance non-Gaussian case.

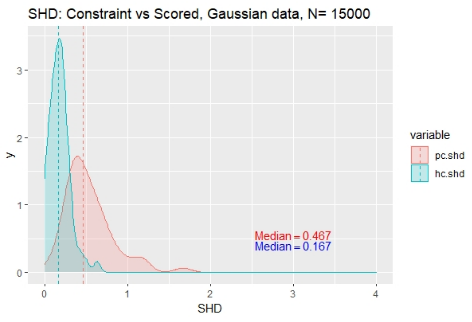


(e) Density plots number of edges Gaussian case.

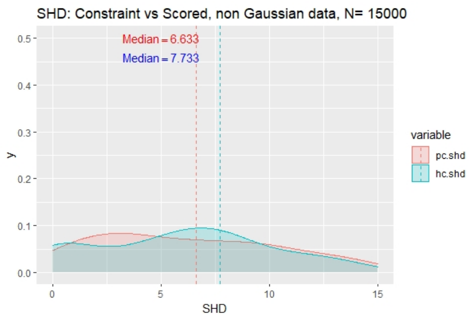


(f) Density plots number of edges non-Gaussian case.

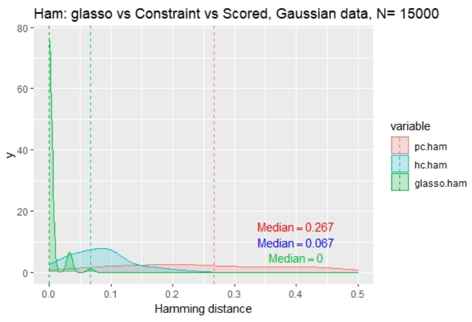
Figure D.7: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 7500$ generated from the big network



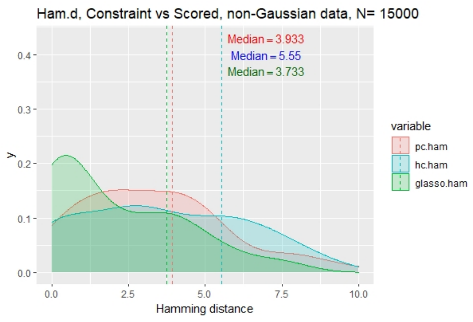
(a) Density plots SHD Gaussian case.



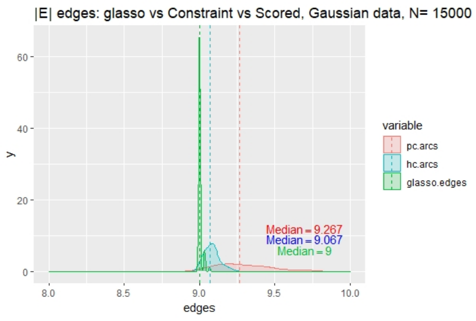
(b) Density plot SHD non-Gaussian case.



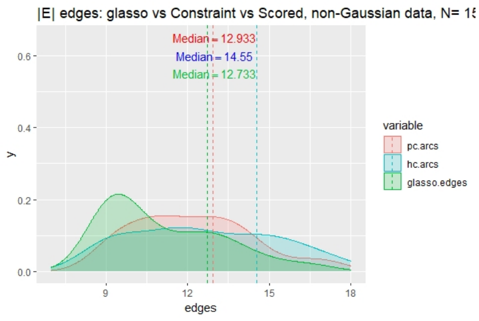
(c) Density plots Hamming distance Gaussian case.



(d) Density plots Hamming distance non-Gaussian case.



(e) Density plots number of edges Gaussian case.



(f) Density plots number of edges non-Gaussian case.

Figure D.8: Comparison of the obtained results applying Graphical Lasso, PC and Hill Climbing to , Gaussian and non-Gaussian data, for $N = 15000$ generated from the big network

E

Code

All the code used in this dissertation to build the examples, obtain figures and tables and perform the simulation studies can be found in the following GitHub repository: https://github.com/Amadeovi/Master-Thesis-Amadeo-Villar. This repository also includes an explanation of the different scripts and its purposes.