Bayesian Network Structure Learning

João Miguel Coelho de Oliveira Ferreira da Trindade

Thesis to obtain the Master of Science degree in

Electrical and Computer Engineering

Supervisor: Prof. Luís Manuel Marques Custódio

Supervision Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Luís Manuel Marques Custódio
Members of the Committee: Prof. Manuel Fernando Cabido Peres Lopes

October 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Acknowledgements

I would like to express my deepest gratitude for my family. It was only thanks to their love and support that I ever managed to reach this far.

I would also like to thank my supervisor, Prof. Luís Manuel Marques Custódio, for his help and guidance throughout this work.

Abstract

Causal reasoning is a fundamental part of human intelligence and can be found across a broad range of fields, from ancient philosophy and theology to medicine and economics. Bayesian networks are probabilistic graphical models that can represent cause-effect relationships and have been used to build system models ranging from medical diagnosis to weather prediction. However, traditional approaches for building Bayesian networks oftentimes prove to be too costly, unethical or impossible. Therefore, there is a need for algorithms that can learn Bayesian networks from observational data alone, which is not trivial due to the combinatorial nature of the search space. Recently, a novel algorithm, *NOTEARS*, has established a new approach, reformulating this problem as continuous optimization, which allows the use of well-studied techniques of machine learning.

This work presents a new meta-algorithm that combines *NOTEARS* with a state-of-the-art traditional algorithm, *FGES*, providing a greater degree of certainty when identifying and interpreting specific relationships encoded on the learned Bayesian networks, even when we do not possess expert knowledge on the field of the observed data. We test the new meta-algorithm on well-known Bayesian networks, showing that it identifies specific relationships with greater precision than the individual algorithms. We also apply it to publicly available data sets and provide a method to evaluate the obtained results when there is no ground truth. In the conducted experiments, the meta-algorithm shows competitive results with the aforementioned algorithms, consistently outperforming *NOTEARS* and, in certain instances, *FGES*.

Keywords: Bayesian networks, Bayesian network parameter learning, Bayesian network structure learning, continuous optimization, causal reasoning

Resumo

O raciocínio causal é uma parte fundamental da inteligência humana, sendo aplicado em áreas desde a filosofia antiga e teologia à medicina e economia. As redes Bayesianas são modelos gráficos probabilísticos capazes de representar relações de causa-efeito, com aplicações em modelos de sistemas de diagnóstico médico e de previsão metereológica, por exemplo. Todavia, as abordagens tradicionais para construir redes Bayesianas vêm frequentemente acompanhadas de custos muito elevados, falta de ética ou inviabilidade. Daí decorre a necessidade de algoritmos capazes de aprender redes Bayesianas exclusivamente a partir de dados observacionais, o que não é trivial dada a natureza combinatória do espaço de procura. Recentemente, um novo algoritmo, *NOTEARS*, estabeleceu uma nova abordagem, reformulando este problema como optimização contínua, o que permite a utilização de técnicas existentes de aprendizagem automática.

O trabalho desta tese consiste na apresentação de um novo meta-algoritmo que combina o *NOTEARS* com um algoritmo tradicional, "state-of-the-art", *FGES*, oferecendo um maior grau de segurança na identificação e interpretação de relações específicas codificadas nas redes Bayesianas aprendidas, mesmo quando não possuímos conhecimento especializado no campo dos dados observados. Testamos este novo meta-algoritmo em redes Bayesianas de referência, demonstrando que ele é capaz de identificar relações específicas com maior precisão do que os algoritmos individuais. Procedemos também à aplicação do novo meta-algoritmo a conjuntos de dados publicamente disponíveis e apresentamos um método para avaliar os resultados obtidos. Nas experiências conduzidas, o novo meta-algoritmo apresentou resultados competitivos com os algoritmos mencionados, superando consistentemente o *NOTEARS* e até o *FGES*.

Palavras-chave: Redes Bayesianas, aprendizagem de parâmetros de redes Bayesianas, aprendizagem da estrutura de redes Bayesianas, optimização contínua, raciocínio causal

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 1
- 1.3 Contributions . . . 2
- 1.4 Outline . . . 2

2 Background **3**
- 2.1 Base Notions of Graph Theory . . . 3
- 2.2 Probabilistic Graphical Models . . . 5
- 2.3 Bayesian Networks . . . 6
  - 2.3.1 D-separation . . . 7
  - 2.3.2 Forward Sampling . . . 8
  - 2.3.3 Bayesian Network Parameter Learning . . . 10
- 2.4 Causal Inference . . . 15

3 Related Work **17**
- 3.1 Base notions . . . 17
- 3.2 Score-based Approaches . . . 17
  - 3.2.1 Scoring functions . . . 17
  - 3.2.2 Search Algorithms . . . 19
- 3.3 Constraint-based Approaches . . . 20
  - 3.3.1 PC Algorithm . . . 20
- 3.4 Hybrid Approaches . . . 21
  - 3.4.1 Max-Min Hill-Climbing . . . 22
- 3.5 Continuous Optimization Approach . . . 22

4 Proposed Solution **25**
- 4.1 Regularized Search . . . 25
  - 4.1.1 Complexity Analysis . . . 27
- 4.2 Metrics for comparing the DAGs . . . 27
- 4.3 Extracting a DAG from a CPDAG . . . 28
- 4.4 Comparing DAGs without Ground Truth . . . 30

5 Experimental Results **33**
- 5.1 Generated Data . . . 33
  - 5.1.1 Data . . . 33
  - 5.1.2 Results . . . 34
- 5.2 Real Data . . . 41
  - 5.2.1 Cardiovascular Disease Data Set . . . 42
  - 5.2.2 Acute Inflammation Data Set . . . 45
  - 5.2.3 Marital Depression Data Set . . . 47
  - 5.2.4 Titanic Data Set . . . 50
  - 5.2.5 Overall Results . . . 52

6 Conclusions **55**
- 6.1 Closing Thoughts . . . 55

6.2 Future Work 56

Bibliography **59**

Appendix A - Overview of the Method used for Comparing DAGs without Ground Truth **63**

Appendix B - Strategies for Non-convex Optimization **64**

List of Figures

2.1 Examples of the distinct classes of graphs. . . . 3
2.2 Examples of paths. . . . 4
2.3 Cycle V_1, V_2, V_3, V_4, V_1. . . . 4
2.4 Trail V_1, V_2, V_3. . . . 4
2.5 Topological ordering V_1, V_2, V_3, V_4. . . . 4
2.6 Naive Bayes network. . . . 5
2.7 Example of a Bayesian network composed of boolean variables. . . . 7
2.8 Important sub-graphs. . . . 8
2.9 Reducing sampling from a multinomial distribution to sampling a uniform distribution in $[0, 1]$. 9
2.10 Sampling the variables in the first topological level. The sampled value is identified by the dashed line, which is green when it is in the interval of values that will correspond to the boolean variable being *True*, or red when it is in the interval of *False* values. . . . 9
2.11 Sampling variable C from the second topological level. The sampled value is identified by the dashed red line, which falls into the interval of values that will correspond to $C =$ *False*. 9
2.12 Sampling the variables in the third topological level. The sampled value is identified by the dashed line, which is green when it is in the interval of values that will correspond to the boolean variable being *True*, or red when it is in the interval of *False* values. . . . 10
2.13 Example of DAGs learned by: 2.13a - a graph structure learning algorithm; 2.13b - a causal discovery algorithm. . . . 16

3.1 Meek's rules. . . . 21

4.1 Overview of the proposed meta-algorithm. . . . 26
4.2 Meta-algorithm example. . . . 27
4.3 Example of all DAGs that can be represented by a specific CPDAG. . . . 29
4.4 Extracting a DAG from a CPDAG. . . . 29
4.5 Bayesian network learned from the example data set. . . . 31

5.1 Bayesian network properties. . . . 34
5.2 Comparison of the CPDAG obtained by *FGES* using the score function *disc-bic-score* with the original DAG. . . . 35
5.3 Comparison of the DAG obtained by *FGES* using the score function *disc-bic-score* with the original DAG. . . . 36
5.4 Precision of the adjacencies of the DAGs obtained by all algorithms, using all score functions. . . . 38
5.5 SHD for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks. . . . 39
5.6 PPV for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks. . . . 39
5.7 TPR for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks. . . . 40
5.8 F_1-Score for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks. . . . 40
5.9 Example CPDAG and DAG. . . . 42
5.10 Obtained CPDAG by the *FGES* algorithm using the *degen-gauss-bic* score function and the extracted DAG for the *Cardio* data set. . . . 44

5.11 Obtained CPDAG by the *Reg-FGES* algorithm using the *bdeu-score* as the scoring function and the extracted DAG for the *Diagnosis* data set. 46
5.12 Obtained DAG by the *NOTEARS* algorithm for the *Marriage* data set. 49
5.13 Obtained CPDAG by the *FGES* algorithm using the *degen-gauss-bic* score function and the extracted DAG for the *Titanic* data set. 51

6.1 Process used to find the relative entropy between the real distribution underlying the data and the joint probability distribution encoded by the learned Bayesian network. 63

List of Tables

2.1 Number of parameters required by the CPTs/CPDs for each node of the Bayesian network of figure 2.7. . . . 7

3.1 Progression for the first few values of the number of DAGs for a network of n nodes. . . . 19

4.1 Example data set comprised of instantiations of boolean random variables. . . . 30
4.2 Approximation of the real probability distribution P^*. . . . 31
4.3 Obtained probability distribution Q. . . . 32

5.1 Properties of the used well-known Bayesian networks. . . . 34
5.2 Precision of the directed edges that were detected by both *NOTEARS* and *FGES* for the various score functions. . . . 34
5.3 Metrics of the obtained DAGs obtained for the *Alarm* network. . . . 37
5.4 Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the various score functions. . . . 37
5.5 Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the score function *bdeu-score*. . . . 37
5.6 Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the score function *disc-bic-score*. . . . 37
5.7 Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the score function *degen-gauss-bic*. . . . 38
5.8 SHD values for the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using as score function the *bdeu-score*. . . . 41
5.9 SHD values for the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using as score function the *disc-bic-score*. . . . 41
5.10 SHD values for the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using as score function the *degen-gauss-bic*. . . . 41
5.11 Age discretization. . . . 43
5.12 Body Mass Index discretization. . . . 43
5.13 Blood pressure discretization. . . . 44
5.14 Marginalized CPD of node *BMI* for analyzing $P(\textit{BMI} \mid \textit{Cardio})$. . . . 44
5.15 Marginalized CPD of node *BMI* for analyzing $P(\textit{BMI} \mid \textit{Cholesterol})$. . . . 45
5.16 Marginalized CPD of node *BP* for analyzing $P(\textit{BP} \mid \textit{BMI})$. . . . 45
5.17 Marginalized CPD of node *BP* for analyzing $P(\textit{BP} \mid \textit{Cardio})$. . . . 45
5.18 Marginalized CPD of node *Cardio* for analyzing $P(\textit{Cardio} \mid \textit{Active})$. . . . 45
5.19 Marginalized CPD of node *Cardio* for analyzing $P(\textit{Cardio} \mid \textit{Cholesterol})$. . . . 45
5.20 Temperature discretization. . . . 46
5.21 Marginalized CPD of node *Temperature* for analyzing $P(\textit{Temperature} \mid \textit{Nephritis})$. . . . 47
5.22 Marginalized CPD of node *Temperature* for analyzing $P(\textit{Temperature} \mid \textit{Inflammation})$. . . . 47
5.23 Marginalized CPD of node *Inflammation* for analyzing $P(\textit{Inflammation} \mid \textit{Micturition})$. . . . 47
5.24 Marginalized CPD of node *Inflammation* for analyzing $P(\textit{Inflammation} \mid \textit{Micturition})$. . . . 47
5.25 BDI score discretization. . . . 48
5.26 Marginalized CPD of node *Working* for analyzing $P(\textit{Working} \mid \textit{Gender})$. . . . 49
5.27 Marginalized CPD of node *BDI* for analyzing $P(\textit{BDI} \mid \textit{Child})$. . . . 49
5.28 Marginalized CPD of node *BDI* for analyzing $P(\textit{BDI} \mid \textit{Marriage})$. . . . 50
5.29 Age discretization. . . . 50
5.30 Siblings/Spouses and Parents/Children discretization. . . . 51

5.31 Fare discretization. 51
5.32 Marginalized CPD of node *Age* for analyzing $P(\textit{Age} \mid \textit{Survived})$. 52
5.33 Marginalized CPD of node *Gender* for analyzing $P(\textit{Gender} \mid \textit{Survived})$. 52
5.34 Marginalized CPD of node *PClass* for analyzing $P(\textit{PClass} \mid \textit{Survived})$. 52
5.35 Relative entropy (KL-divergence) values of the distributions encoded the learned Bayesian networks for all data sets. 52

List of Acronyms

i.i.d.: independently and identically distributed

DAG: Directed Acyclic Graph

CPDAG: Completed Partially Directed Acyclic Graph

MEC: Markov Equivalence Class

GES: Greedy Equivalence Search

FGES: Fast Greedy Equivalence Search

Reg-FGES/R-FGES: Regularized Fast Greedy Equivalence Search

LL: Log Likelihood

BIC: Bayesian Information Criterion

MDL: Minimum Description Length

BD: Bayesian Dirichlet score

BDe: likelihood-equivalence Bayesian Dirichlet score

BDeu: likelihood-equivalence uninformative Bayesian Dirichlet score

MMPC: Max-Min Parents and Children

MLE: Maximum Likelihood Estimator

SHD: Structural Hamming Distance

ACC: Accuracy

PPV: Precision

TPR: Recall

bdeu: bdeu-score (*TETRAD software suite* score function) **db**: discrete-bic-score (*TETRAD software suite* score function)

dgb: degen-gauss-bic (*TETRAD software suite* score function)

Chapter 1

Introduction

1.1 Motivation

Causal reasoning is an integral part of human intelligence, evident from its application throughout time and across a broad spectrum of areas of knowledge. Using it we are able to understand the past, *causes*, and predict the future, *effects*.

Bayesian networks are probabilistic graphical models particularly well-suited to describe cause-effect relationships. On the one hand, their use of probability theory makes them especially adept for modeling stochastic systems, having been applied to climate prediction [1], medicine [2], biological sciences [3], [4], assessing economic trends [5], social modeling [6], and decision making [7]. While on the other hand, their graphical representation provides a simple way to visualize the structure of a model. This can provide valuable insights into the properties of the system it is modeling.

A traditional approach for building a Bayesian network is to conduct randomized experiments, where we intervene in the system and analyze the effects of our intervention on the measurement data. Then, in collaboration with an expert on the field of the system that we are contemplating, we would analyze the results of our interventions and build the network accordingly.

However, this approach generally tends to be either too expensive, due to the running of significant experiments and using an expert's valuable time, or simply impossible, when the experiments are unfeasible or if there is no one with expertise on the subject matter. Therefore, the need for an automated strategy based purely on observational data led to the development of alternative approaches. In the field of probabilistic graphical modeling, this is known as network structure learning [8].

These new methods essentially search for the network structure that best fits the observed data, which is not trivial due to the combinatorial nature of the search space. They make use of clever heuristics and strong assumptions in order to reduce the amount of possible structures to be considered, and still most become intractable for moderately large networks [9].

A recently proposed algorithm, *NOTEARS*, established a new approach, reformulating the search problem as continuous optimization [10]. This allows us to use well-established machine learning methods, while requiring fewer assumptions on the data.

However, neither *NOTEARS* nor the search methods guarantee that the network structure they obtain is the one that best fits the data. Both types of approach are subjected to the pitfalls of non-convex optimization, converging on local optima solutions. Therefore, we should always be skeptical about the quality of the learned network structures [9], [11].

1.2 Objectives

The main objective of this work is that of *knowledge discovery*, for which we will seek to increase the level of certainty on the accuracy of the relationships encoded in the learned Bayesian network.

In order to achieve this goal, we must first review some of the most renowned algorithms for the established Bayesian network structure learning approaches, evaluating their strengths and weaknesses.

Then, we will propose changes to these algorithms to achieve our main objective, which will need to be tested on data generated from well-known Bayesian networks to assess their merit based on commonly used metrics.

Finally, if the changes that we made have proved themselves to be worthwhile in the known Bayesian networks, then we will apply the revised method in a real-world scenario, where we will need to establish a way to evaluate the quality of the learned Bayesian networks.

1.3 Contributions

The main contributions of this work are:

- The formulation of the hypothesis that there are certain relationships so strongly encoded in the data, that they are common to the networks learned by different Bayesian network structure learning algorithms.
- Demonstrating on the data generated from the well-known Bayesian networks that these common relationships are more likely to be in the original Bayesian network than the edges obtained by the individual algorithms.
- Based on the demonstrated validity of the hypothesis, the development of a new meta-algorithm that combines the continuous optimization approach algorithm, *NOTEARS*, with the state-of-the-art search-based approach algorithm, *FGES*.
- Providing a method with which it is possible to evaluate and compare the performance of the various Bayesian network structure learning algorithms when there is no ground truth Bayesian network.

1.4 Outline

This work is split into the following chapters:

- Chapter 2 will define the used notation and introduce the concepts, the fields of study and mathematical tools used on later chapters.
- Chapter 3 will give an overview of the common approaches and some of their most popular algorithms, capping off with the novel continuous optimization approach.
- Chapter 4, where the proposed meta-algorithm and the methods to fairly compare models and transform the results of the algorithms are explained.
- Chapter 5, where the conducted experiments are described, first detailing their setup, proceeded by the experiments on data generated from well-known networks, so as to assess the quality of the proposed solution. This is then followed by tests on real-world data, where the results of the proposed solution serve as a basis for conjecturing over the domains of the used data.
- Chapter 6, where we will recap the important moments of the work, while also establishing possible avenues for future work.

Chapter 2

Background

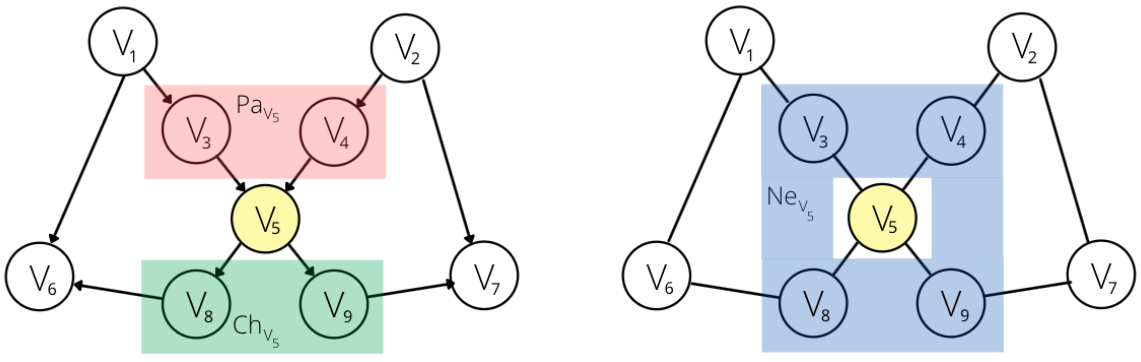
This chapter starts off by covering basic notions of graph theory, which will be necessary for the rest of the work. Then it gives a brief overview of the mathematical tools used (*Probabilistic Graphical Models*), specifying the chosen model (*Bayesian network*), while providing the reasoning for its choice, analyzing some of its most relevant properties and giving a brief overview of Bayesian network parameter learning. Finally, it concludes with the motivation for using Bayesian networks (*Causal inference*) and how it ties into the main focus of this thesis (*Bayesian network structure learning*).

2.1 Base Notions of Graph Theory

A *graph* consists of a set of nodes $\boldsymbol{V} = \{V_1, \ldots, V_n\}$, visually marked as circles, and a set of edges $\boldsymbol{E}$, which represent relationships between the nodes. For instance, the edge E_{V_i,V_j} represents a relationship between nodes V_i and V_j. Graph edges can be either *undirected* (*e.g.,* $V_i - V_j$) or *directed* (*e.g.,* $V_i \rightarrow V_j$), where node V_i is referred to as the *source* node and V_j as the *target* node.

There are two distinct classes of graphs:

- *Directed graphs*, which are comprised solely of directed edges. On these types of graphs, for any given node V_i we can define its set of *parent nodes* as the source nodes of all incoming edges of V_i, denoted as Pa_{V_i}; and its set of *children nodes* as the source nodes of all outgoing edges of V_i, denoted as Ch_{V_i}. In order to illustrate these concepts, see figure 2.1a where we highlight node V_5 in yellow, its parent set, Pa_{V_5}, in red and its children set, Ch_{V_5}, in green.
- *Undirected graphs*, which are comprised solely of undirected edges. On these types of graphs, for any given node V_i we can define its set of *neighbor* (*i.e., adjacent*) *nodes* as the set of all nodes with which V_i is connected to, denoted Ne_{V_i}. In order to illustrate these concepts, see figure 2.1b where we highlight node V_5 in yellow and its neighbor set, Ne_{V_5}, in blue.



(a) Directed graph.

(b) Undirected graph.

Figure 2.1: Examples of the distinct classes of graphs.

In a graph, we define the *degree of a node* as the total number of edges a node is involved with, while the *degree of a graph* is the maximum node degree in the said graph.

Other important graph notions are:

- *Path*: Nodes $V_1, \ldots, V_k$ form a path if there is an edge for every pair of sucessive nodes, be it $(X_i \rightarrow X_{i+1})$ or $(X_i - X_{i+1})$, see figure 2.2a. In the case of directed graphs, a path is said to be directed, see figure 2.2b.

(a) Path V_1, V_2, V_3.

(b) Directed path V_1, V_2, V_3.

Figure 2.2: Examples of paths.

- *Cycle*: Defined as a directed path $V_i, \ldots, V_j$, where $V_i = V_j$, *i.e.,* its end node is its start node, see figure 2.3. If a graph contains no cycles, then it is said to be acyclic.

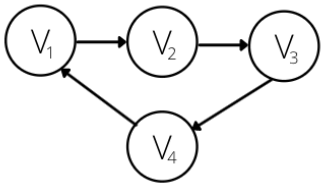


Figure 2.3: Cycle V_1, V_2, V_3, V_4, V_1.

- *Trail*: Defined as a less strict path, where for directed paths edges between successive nodes can have the opposite direction of the succession, *i.e.,* allowing both $(X_i \rightarrow X_{i+1})$ and $(X_i \leftarrow X_{i+1})$. See figure 2.4, noting that it is a trail but not a path.

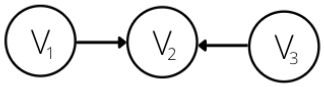


Figure 2.4: Trail V_1, V_2, V_3.

- *Topological ordering*: If for any edge $X_i \rightarrow X_j$, with $i < j$, then $X_1, \ldots, X_j$ is said to be a topological ordering of a graph, see figure 2.5

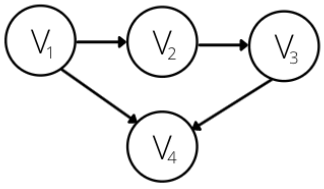


Figure 2.5: Topological ordering V_1, V_2, V_3, V_4.

2.2 Probabilistic Graphical Models

Probabilistic graphical models are powerful knowledge representation tools that make use of probability and graph theory in order to describe the world and make useful predictions about it. They allow us to encode uncertainty into a given mathematical model of the world and they also have an important connection to causal inference.

The probabilistic aspect of modeling is important for two main reasons:

- The real-world is stochastic in nature and not fully observable, therefore it is uncertain.
- Simply making a prediction is not enough, there is also the need to assess the confidence on the prediction.

In order to understand the usefulness of probabilistic graphical modeling, consider the following example. Given a set of n known variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$, we want to predict the outcome variable Y. Therefore the model we build can be expressed as $P(X_1, \ldots, X_n, Y \colon \boldsymbol{\theta})$, where $P(\boldsymbol{X}, Y)$ is the joint probability distribution and $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_m\}$ is the set of the models' adjustable parameters. Our model defines a probability value $[0, 1]$ for each combination of $X_1, \ldots, X_n, Y$. Considering the case when all variables, both known and outcome, are binary, then our model would have 2^{n+1} different combinations of the values of variables.

For real-world problems, the sheer size of the joint probability table is problematic from two points of view:

- In the computational sense, since such a table would be too large to store and to efficiently work with.
- In the statistical sense, since we usually only have limited data and need to efficiently estimate the model's parameters.

Here is where probabilistic graphical modeling makes use of the assumption of *conditional independence* among the variables. A particular case of the choice of these independencies is the *Naive Bayes* assumption, which assumes the independence of the known variables given the outcome variable. This allows us to write the joint probability as a product of factors

$$P(X_1, \ldots, X_n, Y) = P(Y) \prod_{i=1}^{n} P(X_i|Y)\,, \tag{2.1}$$

where each factor $P(X_i|Y)$ can be completely described by a small number of parameters. Thus, the joint distribution is characterized by $\mathcal{O}(n)$ parameters, therefore estimating from data and making predictions is now tractable.

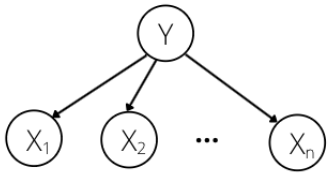


Figure 2.6: Naive Bayes network.

This independence assumption can be conveniently represented in the form of a graph, as seen in figure 2.6. This type of representation allows us to express the notion of causality. The two main types of graphs explored in probabilistic graphical models are:

- *Bayesian Networks (BNs)*, comprised of directed edges in which a given variable X_i is considered to be influenced only by its parents, which are denoted as Pa_{X_i}.
- *Markov Random Fields (MRFs)*, comprised of undirected edges in which a given variable X_i is considered to be influenced only by its neighbors, which are denoted as Ne_{X_i}.

The field of probabilistic graphical modeling can be seen as a conjunction of three distinct aspects:

1. *Representation*, which deals with the issue of choosing a representation for the model, should we use for instance a Bayesian network or a Markov Random Field. This makes use of graph theory in order to construct a tractable model.
2. *Inference*, which deals with the issue of asking questions to the model. This encompasses both *marginal inference*, which queries the marginal probabilities of specific events, and *maximum a posteriori (MAP) inference*, which asks for the most likely assignment of variables.
3. *Learning*, which deals with the issue of fitting the model to real-world data. This is split into learning the parameters of the model, *i.e.,* the factors which simplify the joint probability, and the model's structure itself, *i.e.,* the presence (or absence) of edges between the variables.

The main focus of this work is on the structure learning of discrete Bayesian Networks, which is related to the issue of causal inference. Nonetheless, one should keep in mind that all these aspects of probabilistic graphical modeling are working in tandem.

2.3 Bayesian Networks

A Bayesian network encodes a joint probability distribution P over a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. Formally, a Bayesian network B is a pair $\{G, \boldsymbol{\Omega}\}$, where G is a directed acyclic graph (DAG) in which each node corresponds to one of the random variables, and $\boldsymbol{\Omega}$ specifies the set of conditional probability distributions $P(X_i|\text{Pa}_{X_i})$ for each X_i. The edges or lack of them encode the conditional independence relationships among the variables, with each node X_i being independent of its non-descendant variables given its parents, Pa_{X_i}. Thus, the joint probability distribution of all of the variables is given as

$$P(\boldsymbol{X}) = \prod_{i=1}^{n} P(X_i|\text{Pa}_{Xi})\,, \tag{2.2}$$

where the individual factors $P(X_i \mid \text{Pa}_{X_i})$ are called the *conditional probability distributions (CPDs)*. This equation is known as the chain rule for Bayesian networks [8].

The resulting factorized representation can be substantially more compact, particularly for sparse structures, as opposed to simply using the chain rule of probability,

$$P(\boldsymbol{X}) = P(X_1) \prod_{i=2}^{n} P(X_i \mid X_1, \ldots, X_{i-1})\,, \tag{2.3}$$

where the individual factors $P(X_i \mid X_1, \ldots, X_{i-1})$ are represented by *conditional probability tables (CPTs)*.

For instance, consider the Bayesian network in figure 2.7. Using the chain rule of equation 2.3 and an arbitrarily chosen ordering, (A, B, C, D, E), we would factorize the probability distribution as

$$P(A, B, C, D, E) = P(A)P(B \mid A)P(C \mid A, B)P(D \mid A, B, C)P(E \mid A, B, C, D)\,,$$

where each factor on the right-hand side would be represented by progressively longer CPTs.

Whereas, when using the chain rule for Bayesian networks of equation 2.2, we would factorize the probability distribution as

$$P(A, B, C, D, E) = P(A)P(B)P(C \mid A, B)P(D \mid C)P(E \mid C)\,,$$

where each factor on the right-hand side would be represented by CPDs, generally requiring less parameters than the CPTs.

Since we are dealing with boolean variables, we can describe the parameter $\theta_{X_i=\text{False}} = 1 - \theta_{X_i=\text{True}}$, therefore we only need to store a single column, $X_i = \text{True}$, for all CPTs and CPDs. This means that

the number of parameters that the CPTs and CPDs need to store is the same as their number of rows. Furthermore, the number of rows either requires is 2^q, where q is the number of evidence variables, so $P(X_1)$ requires 1 row, $P(X_2 \mid X_1)$ requires 2 rows, $P(X_3 \mid X_2, X_1)$ requires 4 rows, and so on.

The total number of parameters required can be found on table 2.1, where we can see that even for a small five node boolean variable system, the Bayesian network model allows us to describe the joint probability distribution over $\boldsymbol{X} = \{A, B, C, D, E\}$ with less than a third of the parameters required by just using the chain rule of probability.

Table 2.1: Number of parameters required by the CPTs/CPDs for each node of the Bayesian network of figure 2.7.

Parameters	A	B	C	D	E	**Total**
CPT	1	2	4	8	16	31
CPD	1	1	4	2	2	10

Bayesian networks are of particular interest since:

- They are graphical models, therefore capable of displaying relationships clearly and intuitively, see an example in figure 2.7.
- They are comprised of directed edges, which means that they can represent cause-effect relationships.
- They can handle uncertainty, which is pervasive in most AI application domains, through the use of probability theory.
- They can be used to represent indirect in addition to direct causality.

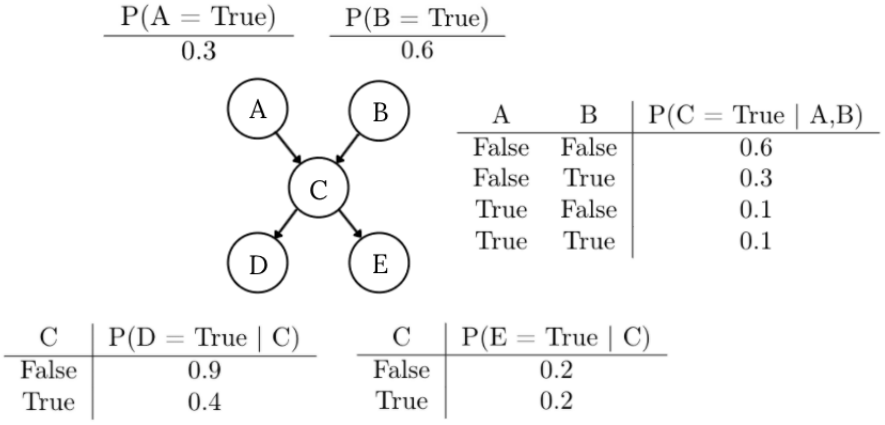


Figure 2.7: Example of a Bayesian network composed of boolean variables.

There are two particularly relevant concepts regarding Bayesian networks:

- *D-separation*, which allows us to visually identify the conditional independencies implied by the Bayesian network's graph.
- *Forward sampling*, which allows us to generate from the Bayesian network a set of samples $\boldsymbol{D}$, *i.e.,* instantiations of all of the network's random variables $\boldsymbol{D} = \{D_1, \ldots, D_M\}$, where $D_m = \{X_1 = x_1^{(m)}, \ldots, X_n = x_n^{(m)}\}$.

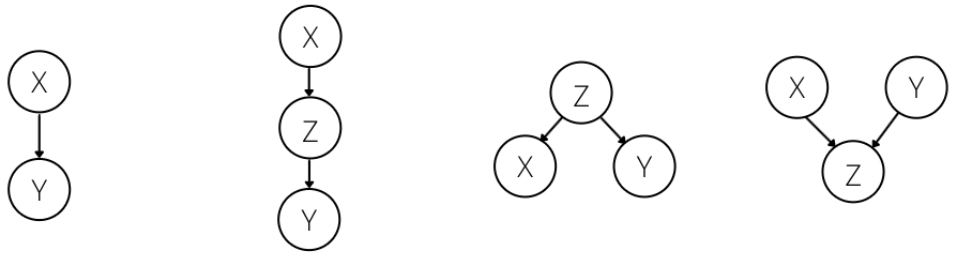
2.3.1 D-separation

In order to comprehend *d-separation*, it is necessary to understand its base concepts, specifically:

- *Independence*: Distribution P satisfies $(X \perp\!\!\!\perp Y)$ if and only if $P(X, Y) = P(X)P(Y)$, which means that knowing the outcome of X does not influence our belief in the outcome of Y.
- *Conditional independence*: Distribution P satisfies $(X \perp\!\!\!\perp Y \mid Z)$ if and only if $P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z)$, which means that given the value of Z, knowing the value of X does not influence our belief in the outcome of Y.

A direct acyclic graph encodes a specific set of conditional independence relationships between its variables [9]. In order to discover this set of independencies, the DAG can be seen as a combination of sub-graphs of the following types:

- *Direct connection*: G is of the form $X \to Y$, in which case $X \not\perp\!\!\!\perp Y \mid \boldsymbol{Z}$ regardless of $\boldsymbol{Z}$.
- *Cascade*: G is of the form $X \to Z \to Y$, in which case $X \perp\!\!\!\perp Y \mid Z$.
- *Common cause*: G is of the form $X \leftarrow Z \to Y$, in which case $X \perp\!\!\!\perp Y \mid Z$.
- *V-structure*: G is of the form $X \to Z \leftarrow Y$, in which case $X \perp\!\!\!\perp Y$ only holds if Z and Ch_Z are unknown.



(a) Direct connection. (b) Cascade. (c) Common cause. (d) V-structure.

Figure 2.8: Important sub-graphs.

Formally, sets $\boldsymbol{X}$ and $\boldsymbol{Y}$ are said to be *d-separated* given $\boldsymbol{Z}$ if there is no active trail between any node $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$ given $\boldsymbol{Z}$, which is denoted as d-sep$_G(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$.

A trail $X_1, \ldots, X_n$ is considered active given a subset $\boldsymbol{Z}$ of observed variables if:

- For all *v-structures* $X_{i-1} \to X_i \leftarrow X_{i+1}$, then X_i or any of its descendants are in $\boldsymbol{Z}$.
- No other node in $\{X_1, \ldots, X_n\}$ is in $\boldsymbol{Z}$.

Consider the distribution P over $\boldsymbol{X}$, then $\boldsymbol{I}(P)$ is the set of independencies of the form $(\boldsymbol{X_A} \perp\!\!\!\perp \boldsymbol{X_B} \mid \boldsymbol{X_C})$ that hold in P, with $\boldsymbol{X_A}, \boldsymbol{X_B}, \boldsymbol{X_C} \subset \boldsymbol{X}$. Now take $\boldsymbol{I}(G)$ to be the set of independencies encoded by a graph G. If $\boldsymbol{I}(G) \subseteq \boldsymbol{I}(P)$, then $\boldsymbol{I}(G)$ is said to be an *independence-map* of $\boldsymbol{I}(P)$.

2.3.2 Forward Sampling

First consider that we want to sample a distribution over a multinomial random variable with k possible outcomes and associated probabilities $\theta_1, \ldots, \theta_k$, *i.e.,* we want to obtain a value (or instantiation) of a random variable that allows multiple possible values whose likelihood is a specific probability value, θ_i for $i \in [0, k]$.

In general, this is not a trivial problem, since computers can only generate samples from very simple distributions, such as the uniform distribution over $[0, 1]$. Therefore, in order to reduce sampling from a multinomial variable to sampling a single uniform variable, we can subdivide a unit interval $[0, 1]$ into k regions with region i having size θ_i, see figure 2.9. Then, we can sample uniformly from the unit interval and return the value of the region in which our sample falls.

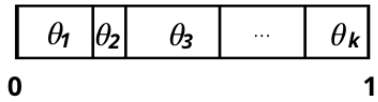


Figure 2.9: Reducing sampling from a multinomial distribution to sampling a uniform distribution in $[0, 1]$.

This technique is extended to Bayesian networks with multinomial variables. Given the joint probability function $P(X_1, \ldots, X_n)$ specified by a Bayesian network, we can sample variables in topological order [8]. This method is called *forward* (or *ancestral*) *sampling* and works as follows:

1. Start by sampling the variables with no parents.
2. Sample the variables on the next topological level by conditioning these variables' CPDs to the values sampled in the previous step.
3. Proceed to the next topological level until all n variables have been sampled.

In order to illustrate this process, consider once more the Bayesian network from figure 2.7. According to the network's topology, we can define three topological levels:

1. *Level 1*, which contains nodes A and B.
2. *Level 2*, which contains node C.
3. *Level 3*, which contains nodes D and E.

In order to apply *forward sampling* to this Bayesian network, we start by sampling the nodes in level 1, A and B, see figure 2.10.

(a) Sampling variable A (b) Sampling variable B

Figure 2.10: Sampling the variables in the first topological level. The sampled value is identified by the dashed line, which is green when it is in the interval of values that will correspond to the boolean variable being *True*, or red when it is in the interval of *False* values.

Moving on to the next topological level, which only contains the node C, this variable will be sampled while conditioned to $A =$ *False* and $B =$ *True*, see figure 2.11.

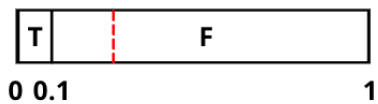


Figure 2.11: Sampling variable C from the second topological level. The sampled value is identified by the dashed red line, which falls into the interval of values that will correspond to $C =$ *False*.

Finally, reaching the third and last topological level, which contains nodes D and E, these will be sampled while conditioned to $C =$ *False*, see figure 2.12.

(a) Sampling variable D (b) Sampling variable E

Figure 2.12: Sampling the variables in the third topological level. The sampled value is identified by the dashed line, which is green when it is in the interval of values that will correspond to the boolean variable being *True*, or red when it is in the interval of *False* values.

When the process of *forward sampling* is concluded, all variables will have been sampled, $\boldsymbol{X} = \{A = \textit{False}, B = \textit{True}, C = \textit{False}, D = \textit{True}, E = \textit{False}\}$. Thus, we will have obtained a sample of the joint probability function defined by the Bayesian network, $D = \{\textit{False}, \textit{True}, \textit{False}, \textit{True}, \textit{False}\}$. By repeating this process M times, we will obtain a set of such data samples, $\boldsymbol{D} = \{D_1, \ldots, D_M\}$.

2.3.3 Bayesian Network Parameter Learning

As previously mentioned, learning Bayesian networks can refer to:

- *Network structure learning*, where we use different algorithms to obtain a DAG from the data.
- *Parameter learning*, where we use different estimators to obtain the CPDs from the data and from a given/learned DAG.

This work's focus is centered on the network structure learning sub-problem, with most of the discussion being on the various Bayesian network structure learning algorithms. However, later on in the work, we will want to learn a full Bayesian network, *i.e.,* learn both the network structure and its parameters. Therefore, it is important to give a basic overview of parameter learning here, which will prove valuable for the later sections.

In order to learn a Bayesian network's parameters, there are two particularly important approaches [8]:

- *Maximum Likelihood Estimator (MLE)*.
- *Bayesian parameter estimation*.

Both of these approaches evaluate of a possible parameter, θ, by seeing how well it predicts the data, *i.e.,* if the data is likely given the parameter, in which case the parameter is a good predictor. Therefore, it is useful to define a:

1. *Parameter space*, Θ, which is the set of possible values of θ that we are considering.
2. *Objective function*, which allows us to evaluate how well the different possible parameters fit to our data set, $\boldsymbol{D}$.

Maximum Likelihood Estimator

Consider a data set $\boldsymbol{D} = \{D_1, \ldots, D_M\}$ composed of M independent and identically distributed (*i.i.d.*) samples of a set of random variables $\boldsymbol{X}$ from an unknown distribution $P^*(\boldsymbol{X})$, *i.e.,* each sample is sampled independently from P^*. Also, assume that we are given a *parametric model* for which we want to estimate its parameters.

Formally, a *parametric model* is defined by a function $P(D\colon \boldsymbol{\theta})$, specified in the terms of a set of parameters. Given a particular set of parameter values $\boldsymbol{\theta}$ and an instance D of $\boldsymbol{X}$, the model assigns a probability to D. In general, for each model, not all parameter values are legal, thus we need to specify the *parameter space*, Θ, which is the set of allowable parameters.

Suppose that X is a multinomial variable that can take values $x_1, \ldots, x_K$. The simplest representation of a multinomial variable is as a vector $\boldsymbol{\theta} \in \mathbb{R}^K$, such that

$$P_{\text{multinomial}}(x\colon \boldsymbol{\theta}) = \theta_k \quad \text{if } x = x_k\,. \tag{2.4}$$

The parameter space of this model is

$$\Theta_{\text{multinomial}} = \left\{ \boldsymbol{\theta} \in [0,1]^K \colon \sum_{k=1}^{K} \theta_k = 1 \right\}. \tag{2.5}$$

We then define the likelihood function, which is the probability the model assigns the data set given a choice of parameters $\boldsymbol{\theta}$,

$$L(\boldsymbol{\theta} \colon \boldsymbol{D}) = \prod_{m=1}^{M} P(D_m \colon \boldsymbol{\theta}) \,. \tag{2.6}$$

We can rewrite the likelihood function in a more compact form by using *sufficient statistics*, which correspond to functions of the data that summarize the relevant information for computing the likelihood.

More formally, a function $\tau(D)$ from instances of $\boldsymbol{X}$ to $\mathbb{R}^l$ (for some l) is a sufficient statistic if, for any two data sets $\boldsymbol{D}$ and $\boldsymbol{D'}$ and any $\boldsymbol{\theta} \in \Theta$, we have that

$$\sum_{D_m \in \boldsymbol{D}} \tau(D_m) = \sum_{D'_m \in \boldsymbol{D'}} \tau(D'_m) \quad \Rightarrow \quad L(\boldsymbol{\theta} \colon \boldsymbol{D}) = L(\boldsymbol{\theta} \colon \boldsymbol{D'}) \,, \tag{2.7}$$

with $\sum_{D_m \in \boldsymbol{D}} \tau(D_m)$ being referred to as the sufficient statistics of the data set $\boldsymbol{D}$.

Continuing with the multinomial model of equation 2.4, a trivial sufficient statistic for the data set is the tuple of counts $\langle M_1, \ldots, M_K \rangle$, where M_k is the number of times that x_k appears in the data set. Given the vector of counts, we can then write the likelihood function as

$$L(\boldsymbol{\theta} \colon \boldsymbol{D}) = \prod_{k=1}^{K} \theta_k^{M_k} \,. \tag{2.8}$$

Once the likelihood function has been defined, we can then use *maximum likelihood estimation* to choose the parameter values. Formally, maximum likelihood estimation states that given a data set $\boldsymbol{D}$, we should choose the parameters $\hat{\boldsymbol{\theta}}$ that satisfy

$$L(\hat{\boldsymbol{\theta}} \colon \boldsymbol{D}) = \max_{\boldsymbol{\theta} \in \Theta} L(\boldsymbol{\theta} \colon \boldsymbol{D}) \,. \tag{2.9}$$

Finally, when estimating the parameters of the multinomial distribution of equation 2.8 [8], the maximum likelihood is achieved when

$$\hat{\theta}_k = \frac{M_k}{M} \,. \tag{2.10}$$

Now, suppose that given a data set $\boldsymbol{D}$ consisting of samples $D_1, \ldots, D_M$, we want to learn the parameters for a Bayesian network with structure $\mathcal{G}$ and parameters $\boldsymbol{\theta}$.

We can write the likelihood function as

$$\begin{aligned} L(\boldsymbol{\theta} \colon \boldsymbol{D}) &= \prod_{m=1}^{M} P_{\mathcal{G}}(D_m \colon \boldsymbol{\theta}) \\ &= \prod_{m=1}^{M} \prod_{i=1}^{n} P(X_i^{(m)} \mid \mathsf{Pa}_{X_i}^{(m)} \colon \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} \left[\prod_{m=1}^{M} P(X_i^{(m)} \mid \mathsf{Pa}_{X_i}^{(m)} \colon \boldsymbol{\theta}) \right], \end{aligned} \tag{2.11}$$

where each of the terms in the square brackets refers to the conditional likelihood of a particular variable given its parents in the network. Furthermore, using $\boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}}$ to denote the subset of parameters that determines $P(X_i \mid \mathsf{Pa}_{X_i})$ in our model, we can write the likelihood as

$$L(\boldsymbol{\theta}\colon \boldsymbol{D}) = \prod_{i=1}^{n} L_i(\boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}}\colon \boldsymbol{D})\,, \tag{2.12}$$

where the local likelihood function for X_i is

$$L_i(\boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}}\colon \boldsymbol{D}) = \prod_{m=1}^{M} P(X_i^{(m)} \mid \mathsf{Pa}_{X_i}^{(m)}\colon \boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}})\,. \tag{2.13}$$

The fact that the likelihood decomposes as a product of independent terms, one for each of the network's CPDs, is known as the *global decomposition* of the likelihood function [8].

Let $\boldsymbol{D}$ be a complete data set for $X_1, \ldots, X_n$, let G be a network structure over these variables, and suppose that the parameters $\boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}}$ are *disjoint* (*i.e.,* each CPD is parameterized by a separate set of parameters that do not overlap) from $\boldsymbol{\theta}_{X_j|\mathsf{Pa}_{X_j}}$ for all $j \neq i$. Let $\hat{\boldsymbol{\theta}}_{X_i|\mathsf{Pa}_{X_i}}$ be the parameters that maximize $L_i(\boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}}\colon \boldsymbol{D})$, then, $\hat{\boldsymbol{\theta}} = \left\langle \hat{\boldsymbol{\theta}}_{X_1|\mathsf{Pa}_{X_1}}, \ldots, \hat{\boldsymbol{\theta}}_{X_n|\mathsf{Pa}_{X_n}} \right\rangle$ maximizes $L(\boldsymbol{\theta}\colon \boldsymbol{D})$ [8].

This means that we can maximize each local likelihood function independently of the rest of the network, and then combine the solutions to get the MLE solution. Furthermore, this decomposition stems from the network structure and does not depend on any particular choice of parameterization of the CPDs.

Now, suppose that we have a variable X with parents $\boldsymbol{U}$. By representing the CPD $P(X \mid \boldsymbol{U})$ as a table, then we will have a parameter $\theta_{x|\boldsymbol{u}}$ for each combination of $x \in \mathsf{Val}(\mathsf{X})$ and $\boldsymbol{u} \in \mathsf{Val}(\boldsymbol{U})$, therefore

$$\begin{aligned} L_X(\boldsymbol{\theta}_{X|\boldsymbol{U}}\colon \boldsymbol{D}) &= \prod_{m=1}^{M} \theta_{x^{(m)}|\boldsymbol{u}^{(m)}} \\ &= \prod_{\boldsymbol{u}\in\mathsf{Val}(\boldsymbol{U})} \left[\prod_{x\in\mathsf{Val}(X)} \theta_{x|\boldsymbol{u}}^{M_{x,\boldsymbol{u}}} \right], \end{aligned} \tag{2.14}$$

where $M_{x,\boldsymbol{u}}$ is the number of times $D_m = x$ and $\boldsymbol{u}^{(m)} = \boldsymbol{u}$ in $\boldsymbol{D}$.

Our goal is to maximize this term while ensuring that $\sum \theta_{x|\boldsymbol{u}} = 1$ for all $\boldsymbol{u}$. Since for different values $\boldsymbol{u}$ of $\boldsymbol{U}$ the choice of parameters are independent from one another, we can independently maximize each of the terms inside the square brackets in equation 2.14.

Therefore, we can further decompose equation 2.14 as a product of multinomial likelihood functions. By using the counts in the data for the different outcomes x, $\{M_{x,\boldsymbol{u}}\colon x \in \mathsf{Val}(X)\}$, we can then compute the MLE parameters,

$$\hat{\theta}_{x|\boldsymbol{u}} = \frac{M_{x,\boldsymbol{u}}}{M_{\boldsymbol{u}}}\,, \tag{2.15}$$

where $M_{\boldsymbol{u}} = \sum_x M_{x,\boldsymbol{u}}$.

Note that we need $M_{\boldsymbol{u}}$ data points to estimate the parameter $\hat{\theta}_{x|\boldsymbol{u}}$. As the number of parents $\boldsymbol{U}$ increases, the number of different $\boldsymbol{u}$ increases exponentially, therefore the number of data points that we expect to have for a single $\boldsymbol{u}$ decreases exponentially. This is known a *data fragmentation* and leads to *overfitting* and the presence of a large amount of zeros in the distribution due to unseen/rarely seen $\boldsymbol{u}$ in the data set. Which means that if the data set is not representative of the real distribution, MLE can lead to parameters that prove themselves inadequate when dealing with unseen data.

Bayesian Parameter Estimation

Similarly to MLE, let us start with a data set $\boldsymbol{D}$ comprised of M *i.i.d.* samples of a set of random variables $\boldsymbol{X}$ from an unknown distribution $P^*(\boldsymbol{X})$. Also, assume that we possess a parametric model $P(D\colon \boldsymbol{\theta})$, where we can choose parameters from a parameter space Θ.

While the MLE approach seeks to find the parameters $\hat{\boldsymbol{\theta}} \in \Theta$ that best fit the data, Bayesian parameter estimation requires that we use probabilities to describe our initial uncertainty about the parameters $\boldsymbol{\theta}$, then using probabilistic reasoning to account for our observations. This is achieved by describing a joint probability distribution over the data and the parameters

$$P(\boldsymbol{D}, \boldsymbol{\theta}) = P(\boldsymbol{D} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})\,, \tag{2.16}$$

where $P(\boldsymbol{D} \mid \boldsymbol{\theta})$ is the likelihood function, and $P(\boldsymbol{\theta})$ is the *prior distribution*, which encodes our initial uncertainty about the parameters.

We can then use Bayes' rule to obtain the *posterior distribution* over the parameters

$$P(\boldsymbol{\theta} \mid \boldsymbol{D}) = \frac{P(\boldsymbol{D} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})}{P(\boldsymbol{D})}\,, \tag{2.17}$$

where $P(\boldsymbol{D})$ is the *marginal likelihood* of the data, *i.e.,* the *a priori* probability of observing the data set $\boldsymbol{D}$ given our prior beliefs, and it is obtained by

$$P(\boldsymbol{D}) = \int_{\Theta} P(\boldsymbol{D} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{2.18}$$

Since the posterior is a product of the *prior* and the likelihood, as seen in equation 2.17, we expect the *prior* to be similar in form to the likelihood. One prior where this is the case is the *Dirichlet distribution*, which is specified by a set of *hyperparameters* $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_K\}$ such that

$$\boldsymbol{\theta} \sim \text{Dirichlet}(\boldsymbol{\alpha}) \quad \text{if} \quad P(\boldsymbol{\theta}) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}\,. \tag{2.19}$$

If $P(\boldsymbol{\theta})$ is Dirichlet($\boldsymbol{\alpha}$), then $P(\boldsymbol{\theta} \mid \boldsymbol{D})$ is Dirichlet($\boldsymbol{\alpha} + \boldsymbol{M}$), where $\boldsymbol{\alpha} + \boldsymbol{M} = \{\alpha_1 + M_1, \ldots, \alpha_K + M_K\}$ and M_k is the number of occurrences of x_k in the data set. This is due to the fact that Dirichlet priors are *conjugate* to the multinomial model.

Formally, a family of priors, $P(\boldsymbol{\theta}\colon \boldsymbol{\alpha})$, is *conjugate* to a particular model, $P(D \mid \boldsymbol{\theta})$, if for any possible data set $\boldsymbol{D}$ of *i.i.d.* samples from $P(D \mid \boldsymbol{\theta})$ and any choice of legal hyperparameters, $\boldsymbol{\alpha}$, for the prior over $\boldsymbol{\theta}$, there are hyperparameters, $\boldsymbol{\alpha'}$, such that $P(\boldsymbol{\theta}\colon \boldsymbol{\alpha'}) \propto P(\boldsymbol{D} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta}\colon \boldsymbol{\alpha})$.

The posterior also allows us to predict the probability of future data instances. Let us assume that we are going to sample a new data instance D_{M+1}, since we already have observed $\boldsymbol{D}$, the Bayesian estimator is the posterior distribution over the new instance

$$\begin{aligned} P(D_{M+1} \mid \boldsymbol{D}) &= \int P(D_{M+1} \mid \boldsymbol{D}, \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid \boldsymbol{D}) d\boldsymbol{\theta} \\ &= \int P(D_{M+1} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid \boldsymbol{D}) d\boldsymbol{\theta} \\ &= \mathbb{E}_{P(\boldsymbol{\theta} \mid \boldsymbol{D})} \{ P(D_{M+1} \mid \boldsymbol{\theta}) \}\,, \end{aligned} \tag{2.20}$$

which means that this prediction is the average over all parameters according to the posterior.

Let $P(\boldsymbol{\theta})$ be a Dirichlet distribution with hyperparameters $\boldsymbol{\alpha}$, and $\alpha = \sum_{k=1}^{K} \alpha_k$, then $\mathbb{E}\{\theta_k\} = \frac{\alpha_k}{\alpha}$. Since the posterior is Dirichlet$(\boldsymbol{\alpha} + \boldsymbol{M})$, where $\boldsymbol{M} = \{M_1, \ldots, M_K\}$ are sufficient statistics from the data, then the prediction with a Dirichlet prior is

$$\begin{aligned} P(x^{(M+1)} = x_k \mid \boldsymbol{D}) &= \mathbb{E}_{P(\boldsymbol{\theta}|\boldsymbol{D})}\{\theta_k\} \\ &= \frac{M_k + \alpha_k}{M + \alpha} \,. \end{aligned} \tag{2.21}$$

For this reason, the hyperparameters are considered to be *pseudo-counts*, *i.e.,* the number of times we expect to have seen the different outcomes in our prior to the experiment [8]. Furthermore, α is known as the *equivalent sample size* and is the total of these pseudo-counts, reflecting our confidence in the prior.

Though the MLE estimate of equation 2.10 assigns probability 0 to values that were not observed in the data set, this should be avoided since they will classify unseen data instances as extremely unlikely. While in Bayesian estimation, even if we do not have prior knowledge, we can use a *uniform* prior that will prevent our estimates from taking values close to 0 by considering all possible values to be equally likely.

Now, consider a network structure G with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_{X_1|\mathsf{Pa}_{X_1}}, \ldots, \boldsymbol{\theta}_{X_n|\mathsf{Pa}_{X_n}})$. Following the Bayesian parameter estimation approach, we need to specify a prior, $P(\boldsymbol{\theta})$, over all possible parameterizations of the network, so that the posterior distribution is

$$P(\boldsymbol{\theta} \mid \boldsymbol{D}) = \frac{P(\boldsymbol{D} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})}{P(\boldsymbol{D})} \,, \tag{2.22}$$

where $P(\boldsymbol{\theta})$ is the prior distribution, $P(\boldsymbol{D} \mid \boldsymbol{\theta})$ is the likelihood function, and $P(\boldsymbol{D})$ is a normalizing constant known as the *marginal likelihood*.

The prior, $P(\boldsymbol{\theta})$, is said to satisfy *global parameter independence* if its form is

$$P(\boldsymbol{\theta}) = \prod_{i=1}^{n} P(\boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}}) \,, \tag{2.23}$$

which in turn allows us to decompose the likelihood, $P(\boldsymbol{D} \mid \boldsymbol{\theta})$, into local likelihoods

$$P(\boldsymbol{D} \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} P(\boldsymbol{\theta}_{X_i|\mathsf{Pa}_{X_i}} \mid \boldsymbol{D}) \,. \tag{2.24}$$

Furthermore, for a variable X with parents $\boldsymbol{U}$, the prior, $P(\boldsymbol{\theta}_{X|\boldsymbol{U}})$, is said to satisfy *local parameter independence* if

$$P(\boldsymbol{\theta}_{X|\boldsymbol{U}}) = \prod_{\boldsymbol{u}} P(\boldsymbol{\theta}_{X|\boldsymbol{u}}) \,. \tag{2.25}$$

When the prior, $P(\boldsymbol{\theta})$, satisfies both global and local parameter independence, then

$$P(\boldsymbol{\theta} \mid \boldsymbol{D}) = \prod_{i=1}^{n} \prod_{\boldsymbol{u}_{X_i}} P(\boldsymbol{\theta}_{X_i|\boldsymbol{u}_{X_i}} \mid \boldsymbol{D}) \,. \tag{2.26}$$

In addition, if $P(\boldsymbol{\theta}_{X|\boldsymbol{u}})$ is a Dirichlet prior with hyperparameters $\alpha_{x_1|\boldsymbol{u}}, \ldots, \alpha_{x_K|\boldsymbol{u}}$, then the posterior, $P(\boldsymbol{\theta}_{X|\boldsymbol{u}} \mid \boldsymbol{D})$, is a Dirichlet distribution with hyperparameters $\alpha_{x_1|\boldsymbol{u}} + M_{x_1,\boldsymbol{u}}, \ldots, \alpha_{x_K|\boldsymbol{u}} + M_{x_K,\boldsymbol{u}}$. Therefore, when predicting a new data instance, we have

$$P(X_i^{(M+1)} = x_i \mid \boldsymbol{U}^{(M+1)} = \boldsymbol{u}, \boldsymbol{D}) = \frac{\alpha_{x_i|\boldsymbol{u}} + M_{x_i,\boldsymbol{u}}}{\sum_{i=1}^{n} \alpha_{x_i|\boldsymbol{u}} + M_{x_i,\boldsymbol{u}}} \,. \tag{2.27}$$

With regards to choosing a *parameter prior* for a Bayesian network, where each node X_i has a set of multinomial distributions $\boldsymbol{\theta}_{X_i|\mathsf{pa}_{X_i}}$ (one for each of the parents of X_i's instantiations, pa_{X_i}), these parameters will each have a separate Dirichlet prior subject to the hyperparameters

$$\boldsymbol{\alpha}_{X_i|\mathsf{pa}_{X_i}} = (\alpha_{x_i^1|\mathsf{pa}_{X_i}}, \ldots, \alpha_{x_i^{K_i}|\mathsf{pa}_{X_i}}) , \tag{2.28}$$

where K_i is the number of values of X_i.

One approach uses a fixed prior, $\alpha_{x_i^j|\mathsf{pa}_{X_i}} = 1$, for all the hyperparameters of the network, which is known as the *K2 prior*.

Another approach considers an imaginary data set $\boldsymbol{D'}$ of "prior" instances, from which we then use counts as hyperparameters

$$\alpha_{x_i|\mathsf{pa}_{X_i}} = \alpha_{x_i,\mathsf{pa}_{X_i}} , \tag{2.29}$$

where $\alpha_{x_i,\mathsf{pa}_{X_i}}$ is the number of times that $X_i = x_i$ and $\mathsf{Pa}_{X_i} = \mathsf{pa}_{X_i}$ in $\boldsymbol{D'}$. Furthermore, we can avoid storing $\boldsymbol{D'}$ by using the size of the data set, α, and a representation $P'(X_1, \ldots, X_n)$ of the frequencies of the events in $\boldsymbol{D'}$, thus

$$\alpha_{x_i|\mathsf{pa}_{X_i}} = \alpha \cdot P'(x_i, \mathsf{pa}_{X_i}) . \tag{2.30}$$

This approach is known as the *BDe prior* and it chooses to represent P' by a Bayesian network, then using inference algorithms (*e.g., Variable-Elimination)* to efficiently compute $P'(x_i, \mathsf{pa}_{X_i})$.

Furthermore, if we assume that the prior distribution over the parameters is uniform, *i.e.,* all values are equally likely, then we can compute the hyperparameters using just the equivalent sample size, α,

$$\alpha_{x_i|\mathsf{pa}_{x_i}} = \frac{\alpha}{r_i \cdot q_i} , \tag{2.31}$$

where $r_i = |X_i|$ is the number of possible values of X_i, and $q_i = \sum_{X_j \in \mathsf{Pa}_{X_i}} r_j$ is the number of possible configurations of the parent set , Pa_{X_i}, of X_i. This is known as the *likelihood-equivalent uninformative Bayesian Dirichlet (BDeu)* prior. For a more detailed analysis on the choice of a Bayesian network parameter prior, see [8].

2.4 Causal Inference

Consider a system composed by a set of random variables, which interact and influence each other. Having access to a set of measurement data of these variables, how can we model the system? This problem is tackled in the probabilistic graphical modeling literature [12]–[15], where the strategy laid out is to use graphical models to model the joint probability distribution over the set of variables that describes the system.

We can describe this system as a graph G, composed by a set of nodes V that correspond to the random variables of the system and by a set of edges E that describe relationships between a pair of nodes. In the case that these edges are directed, *i.e.*, from a source node to a target node, and there are no loops in the graph, then the graph directly encodes a notion of causality between connected nodes.

So given a set of random variables and a set of data that corresponds to measurements of these variables, the problem is how to pick the directed acyclic graph that best fits our data. This is known as a problem of causal inference [9], [16], [17], while in the field of probabilistic graphical modeling it is known as graph structure learning [12], [18], [19].

From Bayes' theorem, we have

$$P(G \mid \boldsymbol{D}) = \frac{P(G)P(\boldsymbol{D} \mid G)}{P(\boldsymbol{D})} , \tag{2.32}$$

where G is a specific Bayesian network structure chosen out of all possible network structures, $\boldsymbol{D}$ is the set of data samples of the system's random variables, while $P(\boldsymbol{D})$ is a normalization constant that does not depend upon structure. Thus, to determine the posterior distribution for network structures, $P(G \mid \boldsymbol{D})$, we need to compute the marginal likelihood of the data, $(p(\boldsymbol{D} \mid G))$ for each possible structure.

Unfortunately, the Bayesian approach of equation 2.32 is often impractical. One important computation bottleneck is produced by the average over models due to the large number of possible structure hypotheses, which will be further elaborated upon later. Consequently, when it is impossible to exclude almost all of these hypotheses, this approach is intractable.

Researchers use two approaches to address this problem:

- *Model selection*: select a "good" model (*i.e.*, structure hypothesis) from among all possible models, and use it as if it were the correct model.
- *Selective model averaging*: select a manageable number of good models from among all possible models and pretend that these models are exhaustive.

Most literature on learning with Bayesian networks is concerned with model selection [20]. In some of these approaches, a *criterion* is used to measure the degree to which a network structure fits the prior knowledge and data. Meanwhile, selective model averaging is more complex, because in order to identify significantly different yet complementary network structures, a single criterion is unlikely to work [21].

It is important to note that though the fields of *graph structure learning* and *causal inference* are closely related, they are not one and the same. The former is simply interested in learning a graph structure from observed data, while the latter's goal is that the links between nodes express causal relationships. Learning *causal graphs* from observational data alone is generally intractable [22]. This is due to the fact that common graph structure learning algorithms assume that all variables are observed, *i.e.,* there are no *latent* (*i.e.,* hidden) variables.

Usually, causal discovery algorithms make use of interventional data and *do-calculus* to establish causal relationships [23]. This is necessary in order to account for possible *confounders*, which are latent variables that are a common parent of observed variables and which may be the cause for an edge between those variables. For instance, see figure 2.13, where both DAGs encode $X \not\perp\!\!\!\perp Y$, yet when viewing edges under a causal lens the graphs imply very different causal relationships. On the one hand, using a graph structure learning algorithm we would learn the DAG that implied X *causes* Y (or the opposite), as seen in figure 2.13a. While on the other hand, using a causal discovery algorithm we would find that there is no causal relationship between X and Y, as seen in figure 2.13b.

(a) DAG with no latent variables.

(b) DAG where Z is a confounder.

Figure 2.13: Example of DAGs learned by: 2.13a - a graph structure learning algorithm; 2.13b - a causal discovery algorithm.

In the next chapter, we will review the approaches that were developed for tackling the problem of Bayesian network structure learning from a model selection perspective, highlighting their most-renowned algorithms.

Chapter 3

Related Work

This chapter covers the algorithms used to learn the structure of Bayesian networks. It starts with the main classes of the traditionally used approaches, providing insight on some of their most popular algorithms. Then it shifts into the newer approaches, focusing on the algorithm that reformulated the original search problem as continuous optimization, *NOTEARS* [10].

3.1 Base notions

Given a data set $\boldsymbol{D} = \{D_1, \ldots, D_M\}$, where D_m is an instantiation of all the variables in $\boldsymbol{V}$, Bayesian network structure learning is the problem of learning a graph structure from $\boldsymbol{D}$. Assuming $\boldsymbol{D}$ is complete, *i.e.,* all variables of $\boldsymbol{X}$ have an observed instantiation, its set of parameters is maximized using frequency counts from the data, as seen in section 2.3.3. Consequently, finding the optimal Bayesian network is reduced to finding the optimal structure that fits the data.

Since the focus is on Bayesian networks, the problem of structure learning amounts to learning the DAG from data. Traditional approaches are split into two major classes:

- Score-based approaches.
- Constraint-based approaches.

3.2 Score-based Approaches

Score-based approaches for Bayesian network structure learning resort to a:

1. *Scoring function*, which is used to measure how well a given structure fits the data.
2. *Search algorithm*, which is used to find the best scoring structure out of all possible DAGs.

3.2.1 Scoring functions

Useful scoring functions are *decomposible*, which means that the score for a given network B can be computed as the sum of scores for its individual variables

$$\text{Score}(B \mid \boldsymbol{D}) = \sum_{i=1}^{n} \text{Score}(X_i \mid \text{Pa}_{X_i}, \boldsymbol{D})\,. \tag{3.1}$$

Commonly used scoring functions fall into one of two camps:

- *Information-theoretic scoring functions*.
- *Bayesian scoring functions*.

The former are based in the log-likelihood function, which is the log probability of D given B. Assuming the data samples are independent and identically distributed, the log likelihood (LL) can be computed as

$$\mathsf{LL}(\boldsymbol{D} \mid B) = \sum_{m=1}^{M} \log P(D_m \mid B) = \sum_{i=1}^{n} \sum_{m=1}^{M} \log P(X_i^{(m)} \mid \mathsf{pa}_{X_i}^{(m)}) \,, \tag{3.2}$$

where $X_i^{(m)}$ is the instantiation of X_i in the data sample D_m, and $\mathsf{pa}_{X_i}^{(m)}$ is the instantiation of X_i's parents in the data sample D_m.

Unfortunately, the LL score is not a good scoring function, since adding a new edge never decreases the likelihood of the network. This favors densely connected networks, where inference has a higher computational cost, and leads to over-fitting to the training data.

In order to address these issues, commonly used scoring functions include a penalizing factor to offset the log-likelihood term and favor less complex networks. Furthermore, most Bayesian networks used for real-world problems tend to be sparsely connected [24].

Bayesian Information Criterion (BIC) score

The BIC score is equivalent to the *minimum description length (MDL)* [25], whose goal is to minimally encode $\boldsymbol{D}$ as:

1. The network structure can be encoded by storing the conditional probability tables of all variables, which requires $\frac{\log M}{2} \times p$, where $\frac{\log M}{2}$ is the memory space expected to be required by a single probability value, and p is the number of individual values for all variables.
2. The unexplained data can be encoded with $\mathsf{LL}(\boldsymbol{D}|B)$ bits.

This way we obtain the MDL penalty term

$$\mathsf{LL\text{-}PN}_{\mathsf{MDL}}(X_i, B, \boldsymbol{D}) = -\frac{\log M \times p_i}{2} \,, \tag{3.3}$$

where p_i is the number of independent parameters for X_i.

The intuition behind $\mathsf{LL\text{-}PN}_{\mathsf{MDL}}$ is that the more complex the model structure, then the longer the encoding. The BIC score is equivalent to the LL score of equation 3.2 with the added the MDL penalty term of equation 3.3. It requires a sufficiently large set of training data, since it is based on the asymptotic behavior of models.

The BIC score is one of the most popular scoring functions, and has been shown to be very competitive with scores that require more assumptions on the nature of the training data [26].

Bayesian Dirichlet scores

For a Bayesian network B with network structure G and Dirichlet priors, where $P(\boldsymbol{\theta}_{X_i|\mathsf{pa}_{X_i}} \mid G)$ has hyperparameters $\{(\alpha^G_{x_i|\boldsymbol{u}_i} : j = 1, \ldots, |X_i|\}$, the *Bayesian Dirichlet (BD) score* [27] is

$$\mathsf{BD}(G, \boldsymbol{D}) = P(B) \prod_{i=1}^{n} \prod_{\boldsymbol{u}_i \in \mathsf{Val}(\mathsf{Pa}^G_{X_i})} \frac{\Gamma(\alpha^G_{X_i|\boldsymbol{u}_i})}{\Gamma(\alpha^G_{X_i|\boldsymbol{u}_i} + M_{\boldsymbol{u}_i})} \prod_{x_i^j \in \mathsf{Val}(X_i)} \frac{\Gamma(\alpha^G_{x_i|\boldsymbol{u}_i} + M_{x_i^j, \boldsymbol{u}_i})}{\Gamma(\alpha^G_{x_i^j|\boldsymbol{u}_i})} \,, \tag{3.4}$$

where $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the *Gamma function*, $M_{\boldsymbol{u}_i}$ is the number of instances in the data set $\boldsymbol{D}$ that $\mathsf{Pa}_{X_i} = \boldsymbol{u}_i$, and $\alpha^G_{X_i|\boldsymbol{u}_i} = \sum_{m=1}^{M} \alpha^G_{x_i^j|\boldsymbol{u}_i}$.

However, it is unusable in practice since it requires specifying a parameter for all possible variable-parent combinations, $\alpha^G_{x_i|\boldsymbol{u}_i}$, and was not score equivalent, *i.e.,* it did not assign the same score to structures that encoded the same set of d-separation facts. Nonetheless, certain cases of the BD score are in fact useful.

In order to address the fact that the BD score is not score equivalent, the *likelihood-equivalence Bayesian Dirichlet (BDe)* score [28] was proposed, introducing the parameter of *equivalent sample size*, α. From this parameter and a prior distribution over network structures, P', all network parameters can be computed as

$$\alpha_{x_i|\mathsf{pa}_{X_i}} = \alpha \cdot P'(x_i, \mathsf{pa}_{X_i}) \,, \tag{3.5}$$

which is the same as the prior used for Bayesian parameter estimation from equation 2.30 seen in section 2.3.3. The only difference is that this prior distribution is defined over network structures, $\boldsymbol{G}$, instead of parameters $\boldsymbol{\theta}$. However, the BDe score is still inadequate since it requires computing $P'(x_i, \mathsf{pa}_{X_i})$, which might not be trivial.

The BDe score was then further improved, originating the *likelihood-equivalence uninformative Bayesian Dirichlet (BDeu) score* [29], which assumed that the prior distribution over network structures was uniform, *i.e.,* they were all equally likely. With this assumption, in order to compute the hyperparameters, the only required parameter is the equivalent sample size, α,

$$\alpha_{x_i|\mathsf{pa}_{X_i}} = \frac{\alpha}{r_i \cdot q_i} \,, \tag{3.6}$$

where $r_i = |X_i|$ is the number of possible values of X_i, and $q_i = \sum_{X_j \in \mathsf{Pa}_{X_i}} r_j$ is the number of possible configurations of the parent set, Pa_{X_i}, of X_i.

The density of the network structure is directly correlated to the value of α, and it has been shown that it is very sensitive to it [30]. Therefore, when the density of the desired network structure is completely unknown, α's selection is not trivial.

3.2.2 Search Algorithms

Having now a score function that allows us to evaluate the fitness of possible network structures to the data, we want to evaluate possible DAGs.

The "simplest" method is known as exhaustive search and evaluates all possible directed acyclic graphs, choosing the one with the best score. Like all brute force methods, this quickly becomes intractable since the number of possible DAGs is super-exponential to the number of nodes [31]. If R_n is the number of DAGs with n vertices, then

$$R_n = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} R_{n-k} \,, \tag{3.7}$$

for $n \geq 1$, and with $R_0 = 1$.

Table 3.1: Progression for the first few values of the number of DAGs for a network of n nodes.

n	1	2	3	4	5	...	10
R_n	1	3	25	543	29281	...	4175098976430598143

Therefore a search strategy for traversing the possible DAGs search space is required. Here is where the decomposability of the scoring function aids us. Instead of scoring all possible DAGs, this property allows us to score mere edge operations, such as adding, deleting or inverting an edge. This allows the creation of greedy algorithms that iteratively perform the edge operation that maximizes the scoring function, starting either with an empty or a complete graph.

However, one should note that though this clever heuristic greatly simplifies the search over possible DAGs, it comes at a cost. Since the graph space is highly non-convex, there is the risk of getting stuck in local maxima of the scoring function. Strategies to escape these local optima include randomly disturbing the network or the use of simulated annealing [32], check Appendix B.

Greedy Equivalence Search

GES conducts its search in the space of Markov Equivalence Classes, which are represented as completed partially directed acyclic graphs (CPDAGs), also known as *patterns* [33] and follows these steps:

1. Start with an empty graph, *i.e.,* all possible marginal and conditional independence constraints.
2. Repeatedly add or reverse a specific edge, with the chosen operation being the one with the highest score according to the chosen score function, until a maximum is reached.
3. Repeatedly remove edges, as long as it increases the scoring function.
4. When a maximum is reached, the result is the CPDAG of the desired structure.

Two DAGs are said to be in the same Markov Equivalence Class if they share the same d-separation facts. In this case, they can both be represented by a CPDAG which contains directed and undirected edges. In this graph, an undirected edge means that neither possible direction would alter the d-separation facts encoded by the graph [9].

The GES algorithm makes use of the *Meek Conjecture* [33]. This conjecture states that for two DAGs H and G, such that H is an independence map of G, *i.e.,* any independence implied by the structure of H is also implied by the structure of G, then there is a finite sequence of edge operations that obey the following properties:

1. After each edge change, G is a DAG and H remains an independence map of G.
2. After all edge changes, $G = H$.

3.3 Constraint-based Approaches

Constraint-based approaches make use of independence tests between the variables, in order to identify a set of edge constraints that the best DAG must satisfy [18]. Since we know that if two variables are independent, then there is no edge connecting them. This type of approach requires extensive testing, so for large networks (above 200 nodes) it becomes intractable.

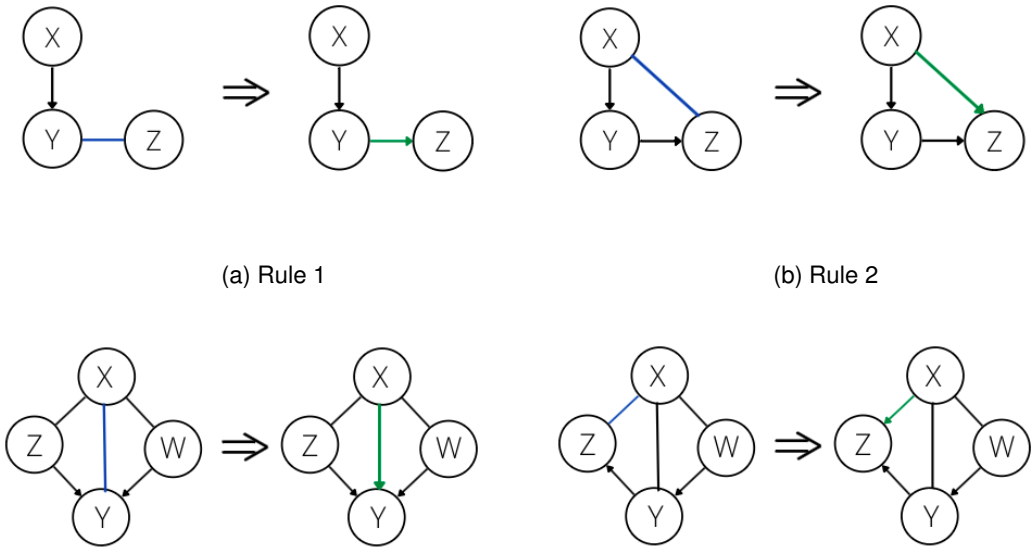
3.3.1 PC Algorithm

Similarly to *Greedy Equivalence Search*, this algorithm reduces the DAG search space to the CPDAG search space. However, it reaches the skeleton of the desired graph differently, doing so as follows:

1. Start with a fully connected undirected graph.
2. For each pair of adjacent nodes X and Y, find the set of nodes $\boldsymbol{Z}$ that are adjacent to X yet are not Y.
3. Check if $X \perp\!\!\!\perp Y \mid Z$ holds, *i.e.,* if X and Y are conditionally independent given $Z \in \boldsymbol{Z}$.
4. If so, remove the edge connecting X and Y and add Z to the separation sets of X and Y, denoted S_{XY} and S_{YX} accordingly.

At the end of this process, we will have the skeleton of the desired graph. In order to transform it into the CPDAG, we will have to identify all possible v-structures in the following manner:

1. For each pair of non-adjacent nodes X and Y with common neighbor Z, check if $Z \notin S_{XY}$.
2. If so, then replace $X - Z - Y$ with $X \rightarrow Z \leftarrow Y$.
3. This results in a partially directed acyclic graph (PDAG), of which we can still assign a direction to certain undirected edges according to *Meek's rules* [34], expressed in figure 3.1:



(a) Rule 1

(b) Rule 2

(c) Rule 3

(d) Rule 4

Figure 3.1: Meek's rules.

Finally, we will have obtained the desired CPDAG. From here, a specific DAG can then be extracted, as we will see later.

It is required to identify all possible v-structures in step 2) in order to obtain the CPDAG, because otherwise the graph would encode different d-separation facts. Thus, the possible v-structures are enforced in an earlier step prior to obtaining the CPDAG, so that it represents a single Markov Equivalence Class.

For instance, consider the DAGs in figures 2.8b and 2.8c, which both state $X \perp\!\!\!\perp Y \mid Z$, therefore are in the same Markov Equivalence Class and can be represented as a CPDAG of three nodes and two undirected edges. Now consider the DAG in figure 2.8d, which states $X \not\perp\!\!\!\perp Y \mid Z$, therefore belongs to different Markov Equivalence Class.

The PC algorithm's evaluation of the independence between pairs of nodes that represent discrete random variables resorts to two well-known tests imported from the field of statistics: the *chi-square test* [35] and the *g-square test* [9],

$$\chi^2 = \sum \frac{(\text{Observed} - \text{Expected})}{\text{Expected}}, \tag{3.8}$$

$$G^2 = 2 \times \sum (\text{Observed}) \ln\left(\frac{\text{Observed}}{\text{Expected}}\right). \tag{3.9}$$

3.4 Hybrid Approaches

This type of algorithms combines both approaches discussed previously. They follow these steps:

1. In the same vein as *constraint-based algorithms*, they use conditional information tests to infer the skeleton of the desired graph.
2. Then, they employ the methods of *score-based algorithms*, greedily performing local search, choosing the edge operation that maximizes a specified score.

These hybrid algorithms share both the positive and negative aspects of the previously mentioned approaches. An extensive conditional independence testing phase is intractable for large networks, though

it has been shown to be theoretically sound. While the skeleton orientation phase incurs the risks of non-convex optimization, it does not provide any theoretical guarantees on the network structure itself [26].

3.4.1 Max-Min Hill-Climbing

On its first part, it applies a local discovery algorithm called *Max-Min Parents and Children (MMPC)* for finding the skeleton of the desired graph, that works similarly to the *PC algorithm*. For each node, it is also looking for sets of possible parents and children nodes.

In order to orient the skeleton, it performs *Greedy Hill-Climbing*:

1. Similarly to *Greedy Equivalence Search*, it begins with an empty graph.
2. However, it constrains the search space by considering only adding edges that were identified in the first phase, when it applied *MMPC*.
3. The other operations are not constrained, and it performs the edge operation that maximizes the specified score.

This algorithm has been shown to achieve competitive results when compared with the more traditional algorithms [36].

3.5 Continuous Optimization Approach

With the recent boom in machine learning, spearheaded by neural networks and their seemingly endless possible applications, one would expect for it to have somehow translated into advances on graph structure learning. This in fact happened for Markov Random Fields, which was recognized as a convex problem [37], therefore solvable using black-box convex optimizers such as *CVX* [38].

Unfortunately, this did not translate for discrete Bayesian Networks. Some promising work has been made in order to distinguish DAGs from the same CPDAG [39]. Here the authors extend the notion of additive noise to discrete models, so that if the joint distribution $P(X, Y)$ admits such a model $Y = f(X) + N$, with $N \perp\!\!\!\perp X$, but not the reversed model, then $X - Y$ is identified as $X \to Y$. However this approach is only tractable for very small networks and the authors themselves only tested it for bivariate systems.

While *constraint-based methods* become intractable for large networks, due to the conditional independence testing performed to identify forbidden edges, the problem with *score-based methods* is one of maintaining acyclicity throughout the edge operations during the search. Both approaches however make strong assumptions on the structure of the data, namely that there are no latent, *i.e., unobservable* variables (also known as *confounders* in the causal inference literature) or need to specify a maximum allowed node degree.

In order to avoid specifying a maximum node degree, which is used to limit the search for the possible parent set for each node, a new approach has been developed [40]. It starts by estimating the score of a large number of parent sets. Then, for only the most promising parent sets, it exactly computes their score.

After obtaining all these possible parent sets, one should use the state-of-the-art exact structure learning optimization solver, *GOBNILP* [41]. This is an integer program that will select for each node the parent set that yields the best scoring DAG, while not introducing cycles.

Recently, a new approach has been developed that avoids the need for extensive knowledge of graph theory and transforms the search over the possible DAG space problem into a continuous optimization problem subject to a novel condition of acyclicity [10]. The author's approach amounts to an equality constraint optimization problem, which ensures the acyclicity of the resulting DAG,

$$h(W) = \mathrm{tr}\left(e^{W \circ W}\right) - d = 0\,, \tag{3.10}$$

where W is the weighted adjacency matrix, $\circ$ is the *Hadamard product*, tr(.) is the trace operator, and d is the number of variables.

This in turn makes the previous *score-based approaches*, which were engaged with maximizing a specific score function while searching in the DAG (or CPDAG) space, into a continuous optimization problem of the form

$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times d}} \quad & F(W) = \frac{1}{2n} \|X - XW\|_F^2 + \lambda \|W\|_1 \\ \text{s.t.} \quad & h(W) = 0 \,, \end{aligned} \tag{3.11}$$

where n is the total number of samples, $\|.\|_F$ is the Frobenius norm, and λ is a regularization parameter that controls the sparsity of the identified weights.

This algorithm, known as *NOTEARS*, uses the following strategy to solve this equality constraint problem (ECP):

1. Convert the constrained problem into a sequence of unconstrained sub-problems. This is achieved via the use of the augmented Lagrangian strategy [42].
2. Optimize the unconstrained sub-problems, for which they employ L-BFGS and Proximal Quasi-Newton optimization techniques [43].
3. Threshold the resulting weighted adjacency matrix, W.

This approach enables the use of several well-studied optimization techniques, such as gradient descent. Though it is a non-convex optimization problem, the authors found that the obtained results were close to the ones found by the state-of-the-art exact algorithm, *GOBNILP* [41].

This novel continuous optimization approach has reinvigorated the DAG structure learning field, sparking multiple new models that improve upon *NOTEARS* [44] or that branch out in other directions [22].

Chapter 4

Proposed Solution

This chapter starts by explaining the meta-algorithm devised from the algorithms covered on the previous chapter, then going into its complexity analysis. This is then followed by specifying the metrics that will be used for comparing the DAGs obtained for generated data seen in the next chapter, which is succeeded by a detailed explanation of the method used to extract a DAG from a CPDAG. Finally, it concludes with the method used to evaluate and compare the Bayesian networks learned from real data in chapter 5.

4.1 Regularized Search

Motivated by the results achieved with an algorithm that required less assumptions on the data [10], there remains a question on how well traditional algorithms really stack up against the recent continuous optimization approach.

When we apply a structure learning algorithm to a specific set of data, which was generated from a known Bayesian network through *forward sampling*, then we can compare the original and the obtained graph. The edges of the obtained graph can be classified as being either:

- *True Positive (TP)*, when it matches an edge in the true graph.
- *False Positive (FP)*, when it does not match an edge in the true graph.

However, if the underlying graph is unknown, then knowing for certain which one of these types a specific edge is becomes a much more complex problem. By analyzing how the different algorithms perform in controlled experiments, we can establish a degree of certainty on the *truthfulness* of the edges identified. This then raises the question of how to increase our sureness on the quality of the obtained graph structure.

While different structure learning algorithms return different graphs for the same data set, there is some overlap on the identified edges. This leads to the hypothesis that these common edges have a higher degree of certainty than the rest and represent some of the data's underlying dependencies.

Following this hypothesis, a meta-algorithm was devised, combining the traditional score-based algorithm, *FGES*, with the recent continuous optimization algorithm, *NOTEARS*. This new meta-algorithm applies these distinct structure learning algorithms so as to find the set of directed edges that the resulting graphs have in common. Then, re-applies the *FGES* algorithm using this set of common directed edges as prior knowledge, *i.e.,* enforcing that instead of starting the search from an empty graph, it now starts from a graph that only contains the set of directed edges common to both *FGES* and *NOTEARS*. Thanks to this process, a regularized form of *FGES* is obtained, which from now on will be referred to as the *Reg-FGES* algorithm. For an overview of the proposed meta-algorithm, see figure 4.1.

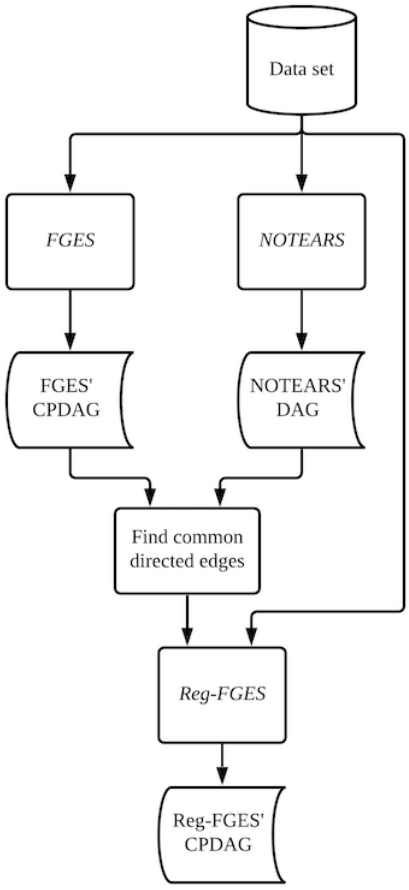


Figure 4.1: Overview of the proposed meta-algorithm.

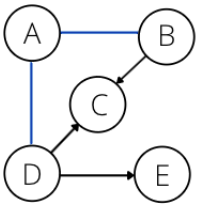
In order to illustrate the proposed meta-algorithm, consider that we want to learn a DAG from a data set comprised of instantiations of five random variables, $\boldsymbol{X} = \{A, B, C, D, E\}$. Following the meta-algorithm, we:

1. Apply the *FGES* algorithm, which starts from an empty graph and iteratively performs the edge operation (adding, deleting or reversing an edge) that maximizes the graph's score according to the chosen score function. This will result in a CPDAG, as seen in figure 4.2a.

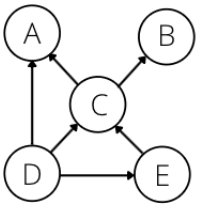
2. Apply the *NOTEARS* algorithm, which iteratively updates the whole weighted adjacency matrix via optimization of the objective function $F(W)$, see equation 3.11. This will result in a DAG, as seen in figure 4.2b.

3. Compare the CPDAG obtained from the *FGES* algorithm with the DAG obtained from the *NOTEARS* algorithm in order to find the directed edges that were detected by both. This results in a set of directed edges, as seen in figure 4.2c.

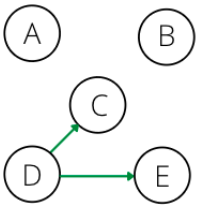
4. Using the set of directed edges that were detected by both algorithms as prior knowledge, apply the *FGES* algorithm again. However, instead of starting from an empty graph, it starts from a graph that only contains this set of common edges. This algorithm, *Reg-FGES*, will result in a CPDAG, as seen in figure 4.2d.



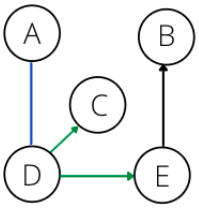
(a) CPDAG obtained by *FGES*



(b) DAG obtained by *NOTEARS*



(c) Overlapping directed edges



(d) CPDAG obtained by *Reg-FGES*

Figure 4.2: Meta-algorithm example.

The blue colored edges in the CPDAGs of figures 4.2a and 4.2d are undirected edges. Meanwhile, the green colored edges in figures 4.2c and 4.2d are the directed edges that were detected by both the *FGES* and *NOTEARS* algorithms, as seen in figures 4.2a and 4.2b.

Note also that, though *Reg-FGES* starts from a graph that contains the edges $D \to C$ and $D \to E$, it outputs a different CPDAG than the one obtained by *FGES*, as seen in figures 4.2a and 4.2d. This is due to the fact that the *FGES* algorithm iteratively performs the local edge operation (adding, removing or reversing an edge) that maximizes the chosen score function. This means that it is subject to getting stuck on a local maximum, therefore starting from different graphs may lead to different results, though it is also possible that they both reach the same CPDAG.

4.1.1 Complexity Analysis

Since the devised method is a meta-algorithm, its overall computational complexity will be the sum of complexities of the individual algorithms. Seeing as how we are regularizing *FGES* with *NOTEARS*, the complexity is as follows:

1. *FGES*: $\mathcal{O}(n^2)$, where n is the number of nodes/variables [45]. Keep in mind that this "low" complexity is only achieved by sufficiently bounding the maximum node degree, otherwise it would be exponential in the number of variables $\mathcal{O}(e^n)$, since it is a combination problem.
2. *NOTEARS*: $\mathcal{O}(n^3)$, due to the fact that the innovative acyclicity constraint requires evaluating the weighted adjacency matrix exponential, $e^{W \circ W}$, see equation 3.10 [10].
3. *Reg-FGES*: $\mathcal{O}(n^2)$, same as *FGES*, yet in practice the newly-attained prior knowledge will restrict the search space of Markov Equivalent Classes, therefore speeding up the search. While *FGES* starts its search from the empty graph, its regularized version, *Reg-FGES*, will start from a graph containing the directed edges that were found by both *NOTEARS* and *FGES*.

Since *NOTEARS* clearly dominates the other algorithms, the overall complexity of the meta-algorithm will be $\mathcal{O}(n^3)$.

4.2 Metrics for comparing the DAGs

In order to compare obtained graphs between themselves and the original, one can make use of a confusion matrix by counting the values of

- *True Positives (TP)*, which correspond to the obtained edges that match the original ones.

- *True Negatives (TN)*, which correspond to the absent edges in the obtained model that also do not exist in the original.
- *False Positives (FP)*, which correspond to the obtained edges that do not exist in the original graph.
- *False Negatives (FN)*, which are edges that exist in the original model, yet are not present in the obtained model.

With these values, one can then compute the commonly used metrics:

$$\text{Accuracy:} \quad \text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}\,; \tag{4.1}$$

$$\text{Precision:} \quad \text{PPV} = \frac{\text{TP}}{\text{TP + FP}}\,; \tag{4.2}$$

$$\text{Recall:} \quad \text{TPR} = \frac{\text{TP}}{\text{TP + FN}}\,; \tag{4.3}$$

$$\text{F1-Score:} \quad \text{F}_1\text{-Score} = 2 \times \frac{\text{PPV} \times \text{TPR}}{\text{PPV + TPR}}\,. \tag{4.4}$$

An alternative metric, popular with this type of problems is the *Structural Hamming Distance (SHD)*. This measures the "distance" from the obtained graph to the original one, by computing the total of edge operations required to turn one into the other

$$\text{SHD} = A + D + R\,, \tag{4.5}$$

where the possible edge operations are:

- Adding an edge (A).
- Deleting an edge (D).
- Reversing the direction of an edge (R).

4.3 Extracting a DAG from a CPDAG

It is important to note that *FGES* returns a CPDAG. However, the goal is to obtain the original graph or a graph as close to it as we can, *i.e.,* the DAG with the lowest possible *SHD* value.

Since the CPDAG represents a Markov Equivalence Class, *i.e.,* a class of DAGs that represent the same set of conditional independences, it is well-nigh impossible to distinguish between DAGs within the same Markov Equivalence Class. This is due to the fact that DAGs in the same Markov Equivalence Class are score-equivalent, *i.e.,* the scoring function assigns the same value to them since they encode the same probability distribution [45]. Recent research with *Additive Noise Models* has shown some promise in this endeavour, yet it is still limited to graphs with a single pair of nodes due to high computational costs [46].

So in order to extract a DAG from a CPDAG, the method described in [9] was applied. For example, consider the DAGs in figures 4.3a, 4.3b and 4.3c. All three of these graphs encode $X \perp\!\!\!\perp Y \mid Z$, and therefore they are in the same Markov Equivalence Class and can be represented as in figure 4.3d, which is the output of *FGES*.

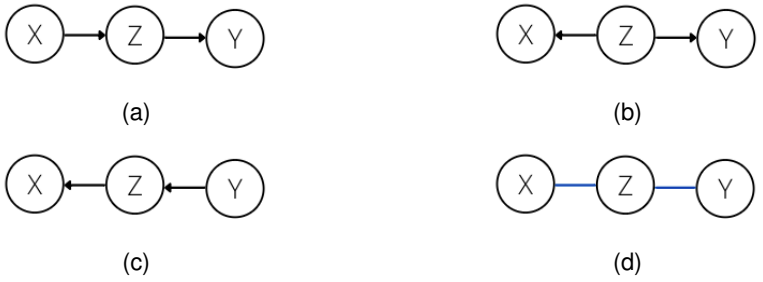


Figure 4.3: Example of all DAGs that can be represented by a specific CPDAG.

The method used works as follows:

1. Pick a random undirected edge from the CPDAG and give it a random direction.
2. Check if there are any remaining undirected edges that share the target node of the originally undirected edge that was given a direction in the previous step.
3. If so, then assign a direction to the undirected edges that were identified in the previous step, in such a way that no new *v-structures* are formed.
4. Repeat the previous two steps, effectively propagating the effect of the first step, as long as there are undirected edges that form a trail starting with the first oriented edge.
5. When all undirected edges that were along this trail are directed, repeat the process starting at the first step, for as long as there are undirected edges.
6. Finally, when all undirected edges have been oriented, we will have a DAG.

In order to illustrate this process, consider that we want to extract a DAG from the CPDAG in figure 4.3d. We start by randomly picking an undirected edge and assigning it a random direction, see figure 4.4a where the chosen edge was $(X - Z)$ and the assigned direction was $(X \rightarrow Z)$. Then the remaining undirected edge in blue, $(Z - Y)$, cannot have the direction of the red arrow, as seen in figure 4.4c, because then the graph would encode $X \not\perp\!\!\!\perp Y \mid Z$, therefore being a different Markov Equivalence Class from the one in figure 4.3. So the only possible direction for the previously undirected edge, $(Z - Y)$, is $(Z \rightarrow Y)$, as seen in figure 4.4.

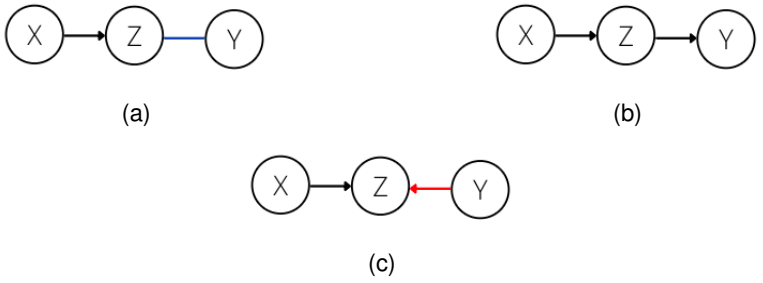


Figure 4.4: Extracting a DAG from a CPDAG.

However, since certain edges were given an arbitrary direction, this may result in an output DAG where these edges may have the opposite orientation of the original one. Therefore, in order to be fair to the *FGES* and *Reg-FGES* algorithms whose output allowed different DAGs, this process of orienting edges needs to be performed multiple times, so as to obtain the resulting graph with the lowest *SHD* value as possible.

Yet, when only the data set is available and the underlying graph is unknown, an obtained graph's *SHD* value is impossible to compute. Alternatively, for simulated experiments one could also compare a CPDAG to the original DAG, in order to orient undirected edges, though it would be even more unfair in its comparison to the result of *NOTEARS*. In addition, when there is no ground truth graph it would be

impossible to orient the undirected edges to obtain a DAG in this manner, much less compare different possible results.

This is precisely the motivating factor for the new meta-algorithm. For in this case it is impossible to know the precision (*PPV*) of a resulting DAG, *i.e.,* know how many of the identified edges are in fact *True Positives*. If in simulated-data experiments, where we can use the previously discussed metrics, we find that the directed edges that were detected by both algorithms have a higher *PPV* than the ones detected by the individual algorithms, then the starting hypothesis will have been validated. While in real-data experiments, *i.e.,* when we only possess the data set, it follows that the common directed edges detected by both algorithms should also more likely be *True Positives*.

4.4 Comparing DAGs without Ground Truth

In real-data experiments, all we possess is the data set of instantiations of the random variables. This means that the original DAG underlying the data is unknown, therefore the metrics described in section 4.2 are impossible to compute, specifically:

- *SHD*, since it measures the distance in terms of edge operations (adding, deleting or reversing an edge) between the obtained DAG and the original DAG. Obviously in the case that the latter is unknown, this is impossible to find out.
- *ACC, PPV, TPR, F_1-Score*, since they require knowing the confusion matrix of the obtained DAG, *i.e.,* identifying which edges are *True Positives, True Negatives, False Positives, False Negatives*. Again, this is only possible when the true DAG is known.

Therefore, in order to evaluate the DAGs obtained by the different algorithms, we will make use of the concept of *relative entropy*, also known as *Kullback-Leibler distance* [47].

The *relative entropy* $D(p\|q)$ is a measure of the inefficiency of assuming that the distribution is q when the true distribution is p, *i.e.,* it is a measure of the distance between the two distributions. More formally, let $p(x)$ and $q(x)$ be two probability mass functions, where p is the true distribution, then the *relative entropy* is defined as

$$D(p\|q) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}\,. \tag{4.6}$$

This distance is always non-negative and is zero if and only if $p = q$. However, it is important to note that it is not a true distance between the distributions since it is not symmetric and does not satisfy the triangle inequality.

Nonetheless, it is still useful for our purposes, since we will use it to compare the closeness of the distributions encoded by the learned Bayesian networks to the real distribution underlying the data.

In order to illustrate this method, consider the following example where we start from a data set $\boldsymbol{D} = \{D_1, \ldots, D_M\}$ seen in table 4.1, where row i corresponds to the sample D_i. The data set's samples are instantiations of the boolean random variables in set $\boldsymbol{X} = \{X_1, \ldots, X_n\}$, and column j corresponds to the random variable X_j.

Table 4.1: Example data set comprised of instantiations of boolean random variables.

A	B	C	D	E
False	True	False	True	False
False	False	True	False	True
True	True	True	False	False
False	True	False	True	False
True	True	True	False	True
False	True	False	True	False
False	False	True	True	False
True	True	False	False	True

The motivation behind using Bayesian networks is the fact that the joint probability distribution underlying the data is unknown. Therefore, in order to evaluate the obtained Bayesian networks via the *relative entropy*, we will need to use an approximation of the real distribution.

A trivial approximation can be found by:

1. Identifying the set of distinct samples (*i.e.,* rows) of the data set, $\boldsymbol{Z} = \{z_1, \ldots, z_p\}$, where $p \leq m$.
2. Computing the relative frequency of each distinct sample, z_i for $i \in [0, p]$.

Continuing with our example, we approximate the joint probability distribution underlying the data set with the probability distribution defined by the relative frequency of each distinct sample, see table 4.2.

Table 4.2: Approximation of the real probability distribution P^*.

A	B	C	D	E	$\boldsymbol{P^*}$
False	True	False	True	False	0.375
False	False	True	False	True	0.125
True	True	True	False	False	0.125
True	True	True	False	True	0.25
False	False	True	True	False	0.125

Essentially, we defined a new random variable, Z, whose event space is the set of distinct rows of the data set, $\boldsymbol{Z}$. We then use the relative frequency of each value of Z to define the probability distribution over this new variable, thus obtaining an approximation of the real probability distribution underlying the data set, P^*.

Now that we have an approximation of the real distribution, the next step is to learn a Bayesian network from the data set, which is done in two steps:

1. Use a Bayesian network structure learning algorithm, such as the ones we discussed at length in chapter 3, to learn a DAG from the data set.
2. With the learned DAG and the data set, learn the parameters of the Bayesian network (*i.e.,* its CPDs) using MLE, thus obtaining the full Bayesian network, see figure 4.5.

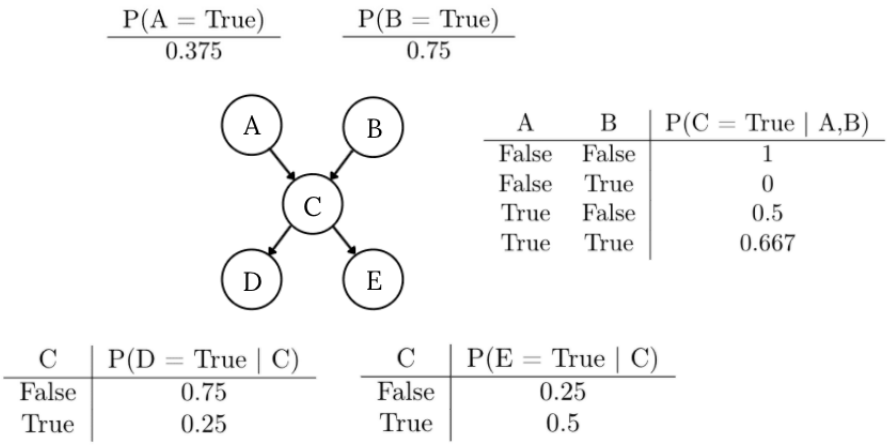


Figure 4.5: Bayesian network learned from the example data set.

Having learned a Bayesian network from the data, we can also use the new discrete random variable Z that we defined when looking for an approximation of the real probability distribution, P^*, to obtain the probability distribution encoded by the Bayesian network, Q.

Now, instead of using the relative frequency, we will make use of the equation 2.2 to compute the parameters of the distribution Q for each distinct row of the data set, which are then normalized (so that they sum to one), see table 4.3.

Table 4.3: Obtained probability distribution Q.

A	B	C	D	E	$\boldsymbol{Q}$
False	True	False	True	False	0.547
False	False	True	False	True	0.121
True	True	True	False	False	0.146
True	True	True	False	True	0.146
False	False	True	True	False	0.04

Finally, we now have an approximation of the real probability distribution, P^*, and the probability distribution encoded by the learned Bayesian network, Q. Therefore, using equation 4.6, we can compute the relative entropy,

$$\begin{aligned} D(P^*\|Q) &= \sum_{z\in \boldsymbol{Z}} P^*(z)\log \frac{P^*(z)}{Q(z)} \\ &\approx 0.119\,. \end{aligned}$$

The obtained entropy value is relatively low, however it is not trivial to evaluate the quality of the learned Bayesian network based solely on this parameter. The closer it is to zero the better, but how close is close enough so that it is a high quality Bayesian network?

While this is a relevant question, for we want to learn Bayesian networks that are well-fitted to the observed data, our current goal is to evaluate the learned Bayesian networks obtained by the different Bayesian network structure learning algorithms. Thus, the relative entropy values computed for the various learned Bayesian networks can be used as a comparison measure, allowing us to identify which algorithm achieved the best results.

In summary, the whole method used to evaluate the obtained Bayesian networks is the following:

1. Find an approximation of the joint probability distribution underlying the data, P^*, by computing the relative frequency of the distinct combinations of the random variable values (*i.e.,* rows) of the data set.
2. Using the DAG obtained by the various Bayesian network structure learning algorithms, obtain the full Bayesian network by learning its parameters, *i.e.,* the CPD tables, using MLE, see equation 2.15.
3. Use equation 2.2 for each of the unique combination of the random variable values identified in step 1), thus obtaining the joint probability distribution of the obtained Bayesian network, Q.
4. Compute the *relative entropy*, $D(P^*\|Q)$, using equation 4.6.

It is important to note that despite the drawbacks of MLE highlighted in section 2.3.3, it is the most appropriate Bayesian network parameter estimator for our purposes.

The approximation of the real probability distribution underlying the data, P^*, was computed using the relative frequencies of each state of the new random variable Z. Therefore, considering that our goal is to compute the relative entropy between P^* and the distribution encoded by the learned Bayesian network, Q, MLE is the appropriate choice, since it also uses the relative frequency of each variable's state for each of its parents' states.

Keep in mind though that this is only because we want to evaluate which of the structure learning algorithms lead to the Bayesian network that best fit the data set. If instead we wanted to use the learned Bayesian network for inference, then we should resort to Bayesian parameter estimation, so as to avoid MLE's problems of *overfitting* and *insufficient data set size*.

For an overview of the method used to obtain the relative entropy between P^* and Q, see figure 6.1 in Appendix A.

Chapter 5

Experimental Results

This chapter analyzes the results obtained by the different Bayesian network structure learning algorithms. Its first half goes over the experiments with data generated from well-known Bayesian networks, specifying the parameters used for the simulations, and seeks to show the validity of the hypothesis that motivated the meta-algorithm from the previous chapter. Then it concludes with the experiments on real data, for which the underlying graph is unknown, using the method described in section 2.3.3 to evaluate and compare the Bayesian networks learned by the different algorithms.

5.1 Generated Data

It has been shown [48] that the *FGES* algorithm, which is an optimized and parallelized version of the original *GES* algorithm [49], outperforms both *PC algorithm* [9] and *MMHC* algorithm [36]. Therefore, the meta-algorithm 4.1 was applied with *FGES* and *NOTEARS* [10], in order to regularize *FGES*, thus obtaining *Reg-FGES*.

The *FGES* algorithm was implemented with the [50] package, while *NOTEARS* was implemented with the code publicly provided by the authors at `https://github.com/xunzheng/notears`.

Since *FGES* requires specifying the maximum node, all trials were run with this parameter set to 10. Bounding this parameter is especially useful for larger networks, in order to lower the computation time.

The scoring functions that were used for *FGES* were those available in the well-renowned *Tetrad software suite* provided by the Center for Causal Discovery, which can be found at `https://github.com/cmu-phil/tetrad`. Since all used Bayesian networks were discrete, the used scoring functions were the *bdeu-score* (see equation 3.6), *discrete-bic-score* (see equation 3.3) and *degen-bic-score* [51]. These are the ones that could handle discrete data, with the latter being able to handle a mix of discrete and continuous data. From now on, they will be referred to as *bdeu* for *bdeu-score*, *db* for *disc-bic-score* and *dgb* for *degen-gauss-bic*.

On the other hand, while *NOTEARS* does not require the same level of output processing as *FGES*, it is still a factor to consider. Since it produces a weighted adjacency matrix at the end of its optimization steps, this requires thresholding the individual weights, otherwise it might contain cycles. Following the recommendation of the authors [10] and also of the authors of one of its follow-up papers [44], this threshold value, ω, was set to 0.3. This hyperparameter serves to remove loops from the weighted adjacency matrix obtained via the *NOTEARS* algorithm. See the original paper for an in-depth analysis of this hyperparameter [10].

5.1.1 Data

In order to test the hypothesis that edges identified by both algorithms were more likely to be true edges, there was the need to use well-known Bayesian networks. All of the used networks were obtained from `https://www.bnlearn.com/bnrepository/` and can be seen on table 5.1 and figure 5.1. Keep in mind that *max in-degree* is the number of parents that the node with most parent nodes in the entire graph has.

Table 5.1: Properties of the used well-known Bayesian networks.

Network	Nodes	Edges	Parameters	Max In-Degree
Asia [52]	8	8	18	2
Child [53]	20	25	230	2
Insurance [54]	27	52	984	3
Alarm [2]	37	46	509	4
Hailfinder [1]	56	66	2656	4
Win95pts	76	112	574	7
Andes [55]	223	338	1157	6

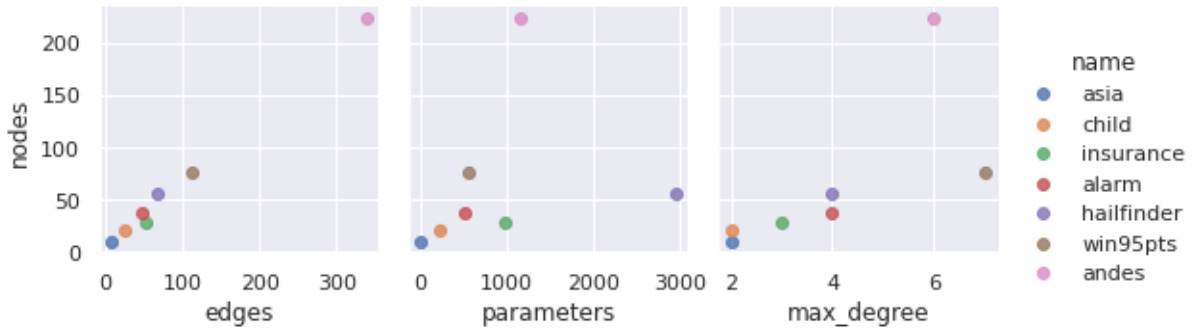


Figure 5.1: Bayesian network properties.

Since the *Bayesian network repository* supplies both the graphs and conditional probability tables for well-known Bayesian networks, in order to generate a data set for a given network, it is a simple matter of repeatedly applying forward sampling, as seen in section 2.3.2.

5.1.2 Results

In order to test the starting hypothesis that the directed edges detected by both algorithms are highly likely to be true edges, since the true graph is known, then it is a simple matter of analyzing the precision values of these edges, see table 5.2.

Table 5.2: Precision of the directed edges that were detected by both *NOTEARS* and *FGES* for the various score functions.

PPV [%]	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
bdeu	**100.0**	**100.0**	62.5	**95.0**	71.43	**92.45**	96.36
db	-	**100.0**	**85.71**	94.44	**100.0**	86.36	**100.0**
dgb	80.0	**100.0**	**85.71**	88.89	71.43	87.5	91.53

Note that for the score function *disc-bic-score* on the *Asia* Bayesian network, no directed edge detected by *NOTEARS* was detected by *FGES*, therefore there were no common directed edges.

Apart from a few relatively low precision values, there are multiple instances where the precision values are perfect, whilst the rest are also relatively high. Motivated by these experimental results, the hypothesis now appears to be validated. That is, there is likely a set of edges crucial to the underlying probability distribution of the data, therefore being captured by both algorithms.

Following the meta-algorithm, these directed edges common to both *NOTEARS* and *FGES* are then considered as prior knowledge and fed into *FGES* algorithm, so as to regularize it. By ensuring these edges are part of the final output graph, one would expect to speed up the search process, since the search space has been decreased, and possibly to achieve better results.

For now, we will focus on the CPDAG obtained by *FGES* using the score function *disc-bic-score* on the *Alarm* Bayesian network, as seen on figure 5.2. The edges are color-coded in the following manner

- Black edges are the edges of the original DAG. An example of a black edge on figure 5.2 is the edge (*Kinkedtube* → *Ventlung*), which is marked with a black 1.
- Green edges are directed edges of the CPDAG that match an edge in the original DAG, *i.e.,* there is also a black edge between the same nodes that has the same direction as the green edge. An example of a green edge on figure 5.2 is the edge (*Disconnect* → *Venttube*), which is marked with a green 2. Note how there is a black edge between the same nodes with the same direction, which is marked with a black 2.
- Yellow edges are directed edges of the CPDAG that have the opposite direction of an edge in the original DAG, *i.e.,* there is a black edge between the same nodes that has the opposite direction of the yellow edge. An example of a yellow edge on figure 5.2 is the edge (*Ventlung* → *Intubation*), which is marked with a yellow 3. Note how there is a black edge between the same nodes with the opposite direction, (*Intubation* → *Ventlung*), which is marked with a black 3.
- Blue edges are undirected edges of the CPDAG for which one of the possible directions matches an edge in the original DAG, *i.e.,* there is a black edge between the same nodes. An example of a blue edge on figure 5.2 is the edge (*Anaphylaxis* – *TPR*), which is marked with a blue 4. Note how there is a black edge between the same nodes, (*Anaphylaxis* → *TPR*), which is marked with a black 4.
- Red edges can either be directed or undirected edges of the CPDAG between nodes that are not connected by an edge in the original graph. An example of a red undirected edge on figure 5.2 is the edge (*Anaphylaxis* – *Insuffanesth*), which is marked with a red 5. While an example of a red directed edge is the edge (*Ventube* → *Minvol*), which is marked with a red 6. Note that in neither of these examples there is a black edge between the same nodes.

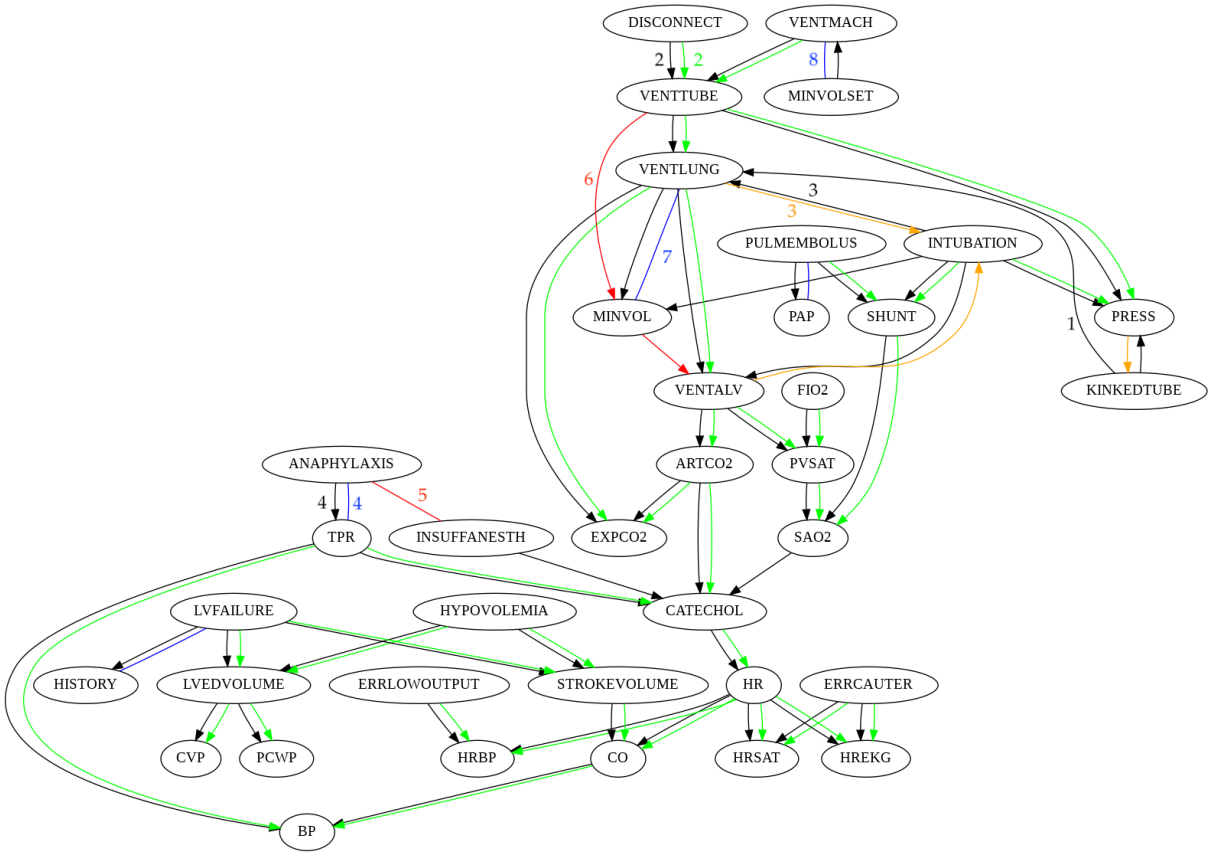


Figure 5.2: Comparison of the CPDAG obtained by *FGES* using the score function *disc-bic-score* with the original DAG.

The step that follows is extracting a DAG from the CPDAG in figure 5.2, using the method previously explained. The resulting DAG can be seen on figure 5.3.

The previously mentioned color-code is altered slightly due to assigning a direction to the undirected edges of the CPDAG in figure 5.2. The edges that were blue in the CPDAG are now color-coded as follows:

- Purple edges are edges in a DAG extracted from a CPDAG that match an edge in the original DAG, *i.e.,* there is also a black edge between the same nodes that has the same direction as the purple edge. An example of a purple edge on figure 5.3 is the edge $(\textit{Ventlung} \rightarrow \textit{Minvol})$, which is marked with a purple 7. Note how there is a black edge between the same nodes with the same direction, which is marked with a black 7. This purple edge corresponds to the blue edge $(\textit{Ventmach} - \textit{Minvolset})$ in the CPDAG in figure 5.2, where it is marked with a blue 7.

- Pink edges are edges in a DAG extracted from a CPDAG that have the opposite direction of an edge in the original DAG, *i.e.,* there is a black edge between the same nodes that has the opposite direction of the pink edge. An example of a pink edge on figure 5.3 is the edge $(\textit{Ventmach} \rightarrow \textit{Minvolset})$, which is marked with a pink 8. Note how there is a black edge between the same nodes with the opposite direction, $(\textit{Minvolset} \rightarrow \textit{Ventmach})$, which is marked with a black 8. This pink edge corresponds to the blue edge $(\textit{Ventmach} - \textit{Minvolset})$ in the CPDAG in figure 5.2, where it is marked with a blue 8.

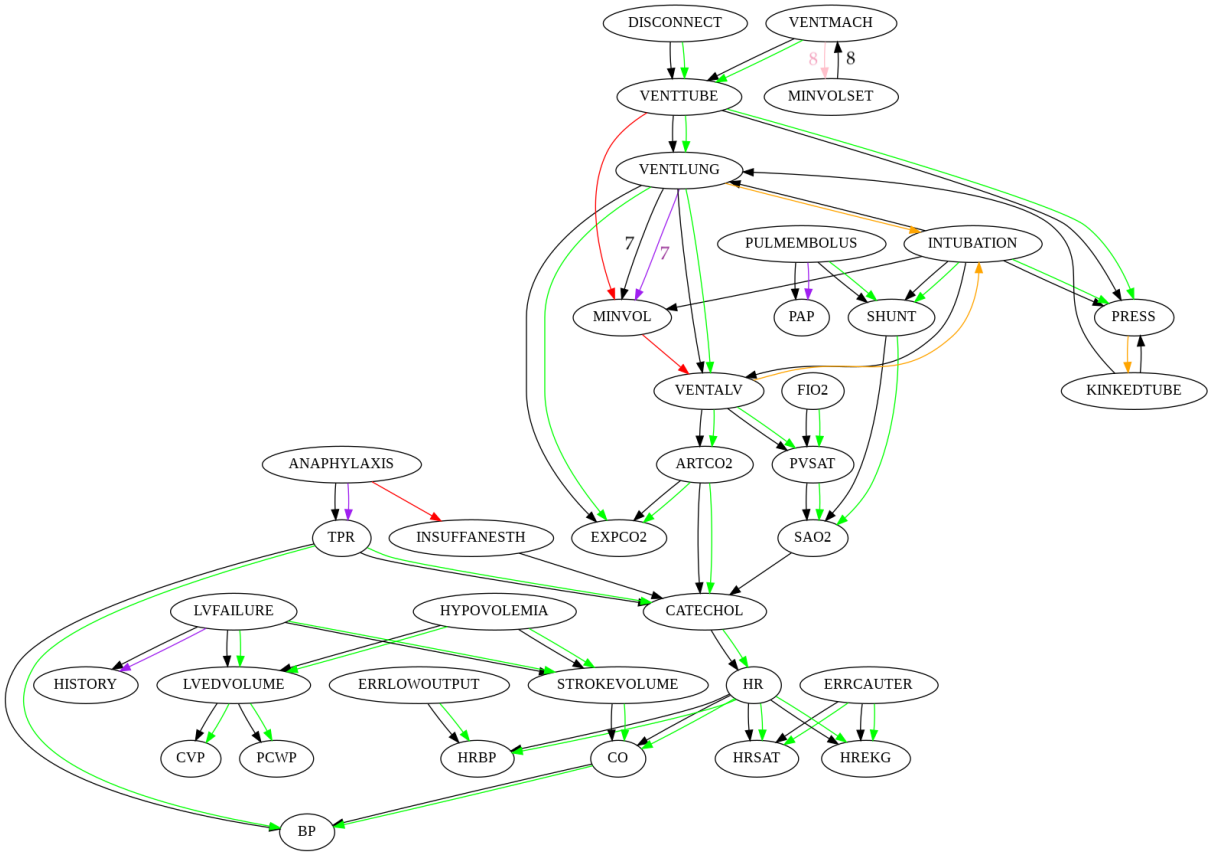


Figure 5.3: Comparison of the DAG obtained by *FGES* using the score function *disc-bic-score* with the original DAG.

Now it is straightforward to compute the relevant metrics for the final obtained DAG. The results for all these metrics of the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the various score functions, for the *Alarm* network can be found on table 5.3. Keep in mind that the values for *ACC, PPV, TPR*, and the F_1*-Score* are percentages.

Table 5.3: Metrics of the obtained DAGs obtained for the *Alarm* network.

Metrics	*Edges*	*TP*	*FN*	*FP*	*ACC*	*PPV*	*TPR*	*F_1-Score*	*SHD*
NOTEARS	59	21	25	38	95.68	35.59	45.65	40.0	63
FGES/bdeu	48	41	5	7	96.58	**85.42**	**89.13**	**87.24**	**12**
FGES/db	45	38	8	7	**96.78**	84.44	82.61	83.51	15
FGES/dgb	57	39	7	18	95.93	68.42	84.78	75.73	25
R-FGES/bdeu	49	37	9	12	96.49	75.51	80.43	77.89	21
R-FGES/db	46	36	10	10	96.7	78.26	78.26	78.26	20
R-FGES/dgb	58	37	9	21	95.85	63.79	80.43	71.15	30

It is also of interest to compute the precision of its detected *adjacencies*. In this context, when two nodes are connected by an edge, they are said to be neighbors or *adjacent*. Therefore, all non-red colored edges in the CPDAG in figure 5.2 are proper adjacencies, since they establish a link between a pair of variables that is connected in the original DAG. Then, the precision of the adjacencies is the ratio of proper adjacencies over all detected edges, regardless of their color. The adjacency precision values obtained by all the algorithms and score functions can be found on table 5.4.

Table 5.4: Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the various score functions.

PPV [%]	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	66.67	50.0	40.0	45.76	23.53	54.47	78.26
FGES/bdeu	50.0	**100.0**	79.25	91.67	66.67	62.34	86.14
FGES/db	**83.33**	**100.0**	**92.86**	**93.33**	75.38	**78.43**	**89.15**
FGES/dgb	**83.33**	**100.0**	72.13	77.19	**79.71**	75.68	82.16
R-FGES/bdeu	50.0	**100.0**	78.85	89.8	64.38	58.28	80.78
R-FGES/db	**83.33**	**100.0**	**92.86**	91.3	75.38	73.91	88.79
R-FGES/dgb	**83.33**	**100.0**	72.13	75.86	71.01	67.97	80.06

The adjacency precision values obtained for the *Alarm* network in table 5.4 and the precision values on table 5.3 show an obvious disparity. This leads to the conclusion that the DAGs obtained by the various algorithms and score functions contain a significant amount of edges that have the opposite direction in the original DAG. While not ideal since they fail to identify the correct edge direction, it is encouraging that they identify the existence of an edge between nodes that are connected in the original DAG.

Finally, comparing the adjacency precision values of the DAGs obtained by the various algorithms and score functions with the adjacencies expressed by the set of directed edges common to both *NOTEARS* and *FGES* only further corroborates the starting hypothesis. See tables 5.5, 5.6, 5.7, for each specific score function. Also see figure 5.4 for an overall comparison.

Table 5.5: Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the score function *bdeu-score*.

PPV [%]	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	66.67	50.0	40.0	45.76	23.53	54.47	78.26
FGES/bdeu	50.0	**100.0**	79.25	91.67	66.67	62.34	86.14
R-FGES/bdeu	50.0	**100.0**	78.85	89.8	64.38	58.28	80.78
Common/*bdeu*	**100.0**	**100.0**	**100.0**	**100.0**	**85.71**	**92.45**	**96.36**

Table 5.6: Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the score function *disc-bic-score*.

PPV [%]	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	66.67	50.0	40.0	45.76	23.53	54.47	78.26
FGES/db	83.33	**100.0**	92.86	93.33	75.38	78.43	89.15
R-FGES/db	83.33	**100.0**	92.86	91.3	75.38	73.91	88.79
Common/*db*	-	**100.0**	**100.0**	**100.0**	**100.0**	**90.91**	**100.0**

Table 5.7: Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the score function *degen-gauss-bic*.

PPV [%]	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	66.67	50.0	40.0	45.76	23.53	54.47	78.26
FGES/dgb	83.33	**100.0**	72.13	77.19	79.71	75.68	82.16
R-FGES/dgb	83.33	**100.0**	72.13	75.86	71.01	67.97	80.06
Common/*dgb*	**100.0**	**100.0**	**85.71**	**94.44**	**85.71**	**89.58**	**96.61**

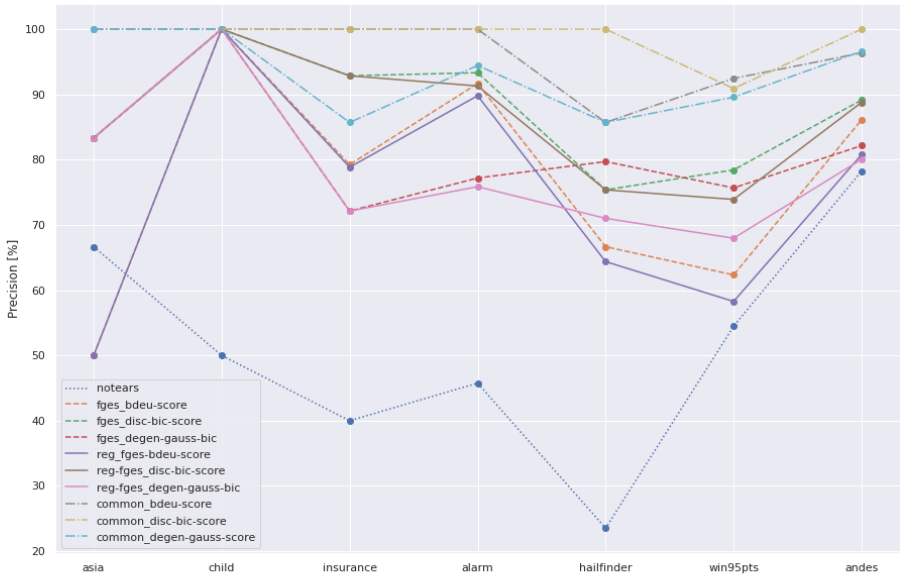


Figure 5.4: Precision of the adjacencies of the DAGs obtained by all algorithms, using all score functions.

Having now seen the strength of the starting hypothesis, and possessing a clear notion of the comparisons made for a specific network, the most relevant metrics for all networks can be seen in figures 5.5, 5.6, 5.7 and 5.8. Note that the legend in figure 5.5 is the one used in all plots, where the dotted line represents *NOTEARS*, the dashed lines represent *FGES* and the full lines represent *Reg-FGES*.

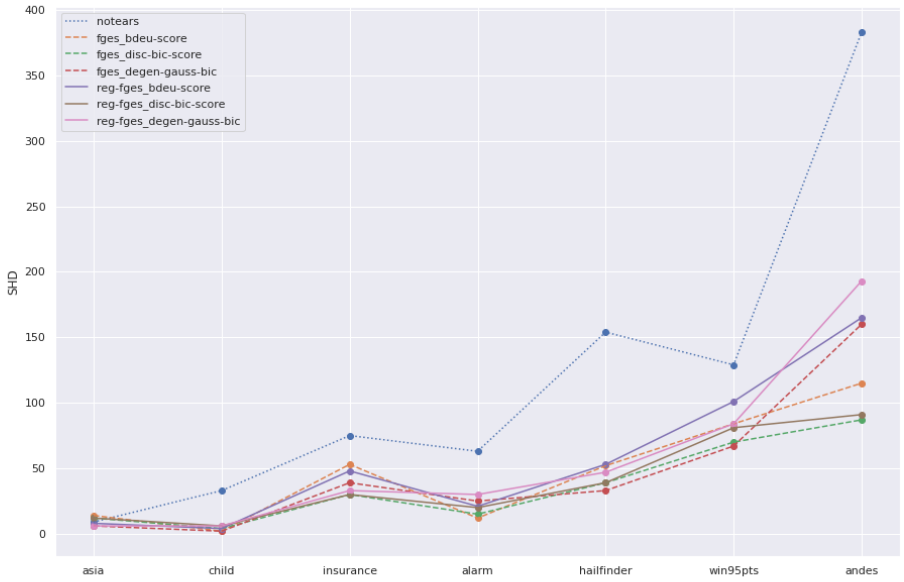


Figure 5.5: SHD for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks.

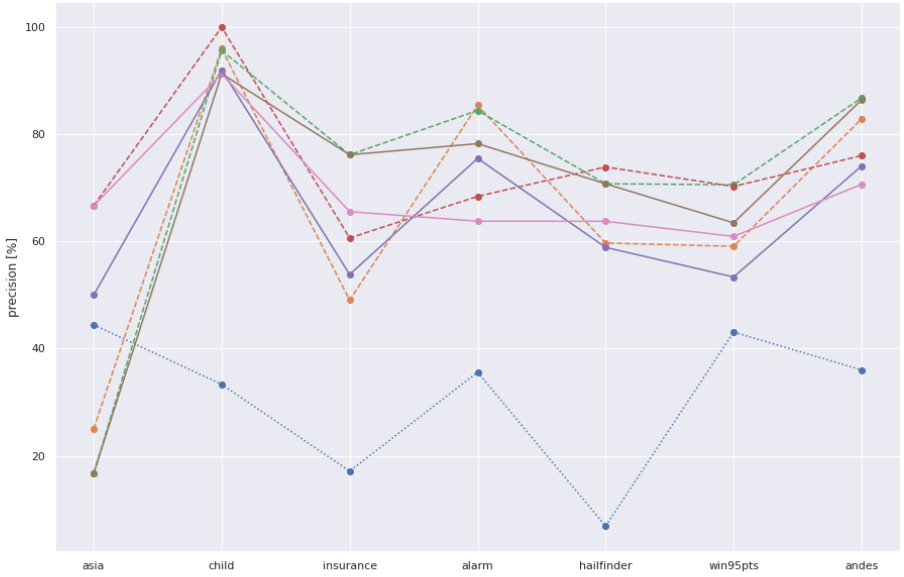


Figure 5.6: PPV for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks.

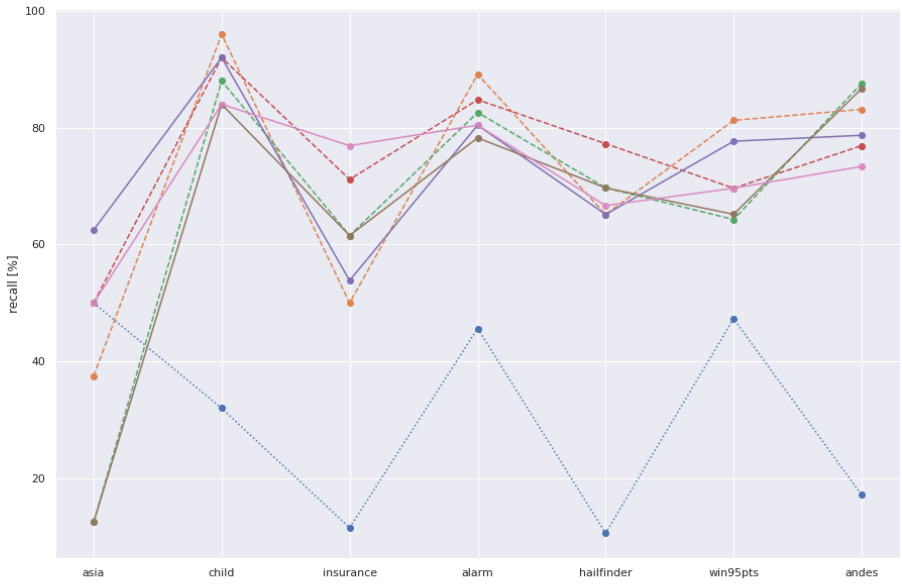


Figure 5.7: TPR for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks.

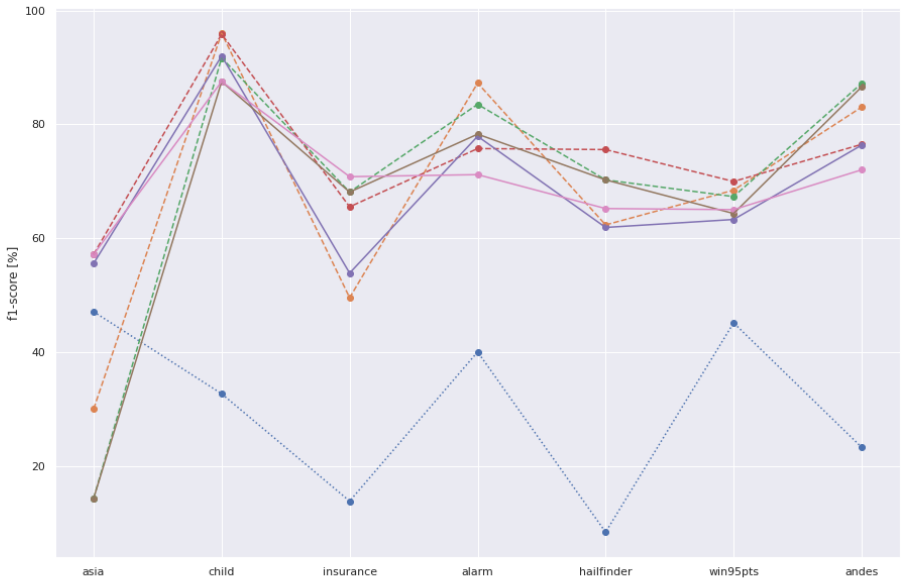


Figure 5.8: F_1-Score for the DAGs obtained by the *NOTEARS*, *FGES* and *Reg-FGES*, for all Bayesian networks.

In order to get a clearer picture of the results in figure 5.5, see tables 5.8, 5.9 and 5.10, which show the *SHD* values of the DAGs obtained by the algorithms using a specific score function for *FGES* and *Reg-FGES*. Note that for SHD, smaller is better.

Table 5.8: SHD values for the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using as score function the *bdeu-score*.

SHD	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	9	33	75	63	154	129	383
FGES/bdeu	14	**2**	53	**12**	**52**	**84**	**115**
R-FGES/bdeu	**8**	4	**48**	21	53	101	165

Table 5.9: SHD values for the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using as score function the *disc-bic-score*.

SHD	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	**9**	33	75	63	154	129	383
FGES/db	12	**4**	**30**	**15**	**39**	**70**	**87**
R-FGES/db	12	6	**30**	20	**39**	81	91

Table 5.10: SHD values for the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using as score function the *degen-gauss-bic*.

SHD	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	9	33	75	63	154	129	383
FGES/dgb	**6**	**2**	39	**25**	**33**	**67**	**160**
R-FGES/dgb	**6**	6	**33**	30	47	84	193

In spite of the relatively poor results of *NOTEARS* in all metrics, *Reg-FGES* displays similar results to *FGES*. Though generally poorer than in the non-regularized version, *Reg-FGES* still manages to achieve the best or on-par results for many of the Bayesian networks, for the various score functions. In addition, even when it obtains a worse score, it is often close to the best achieved score, for instance compare the scores for the *Hailfinder* network on table 5.8, or the ones for the *Andes* network on table 5.9.

While the regularization of *FGES* did not translate into significant improvements on the various metrics, this should be taken with a grain of salt, for:

- On the one hand, since this is a non-convex optimization problem, there might be local optima near the starting graph of *Reg-FEGS*, which contains the set of common directed edges detected by both *NOTEARS* and *FGES*.
- On the other hand, the arbitrariness of the method used to extract a DAG from the obtained CPDAG strongly influences the final DAG, as can be seen from the comparison of the precision values of the DAG's edges with the precision values of its adjacencies.

Nonetheless, even if the meta-algorithm did not achieve its full promise and the quality of the edges from the obtained CPDAGs is not markedly better, the validity of the starting hypothesis is already a valuable development.

By discovering that the directed edges common to both *NOTEARS* and *FGES* have a higher likelihood of being present in the goal DAG, our knowledge of the previously unknown system, of which all that was known were observed instances of its variables, is now enriched with the discovery of some highly likely relationships.

5.2 Real Data

In order to illustrate the applicability of the Bayesian network model to real-world data, the previously described Bayesian network structure learning algorithms were tested on some publicly available data sets. However, one should note that these data sets contain a set of mixed random variables, so in order to stay consistent with the focus on discrete random variables, the continuous variables were discretized.

While in section 5.1, we generated data samples from known Bayesian networks, now we are dealing with purely observational data, so the underlying DAG is unknown. Therefore, it is then impossible to apply the metrics from section 4.2 to evaluate and compare the DAGs obtained by the various algorithms.

This makes measuring the quality of the obtained DAGs based solely on their structure not trivial, especially when dealing with data from fields of study of which we do not possess expert knowledge. For this reason, we will follow the method explained in section 4.4 so as to be able to evaluate and compare the learned Bayesian networks.

On this section, certain edges are particularly important and so they should be highlighted. In order to illustrate this, consider the graphs in figure 5.9.

(a) CPDAG

(b) DAG

Figure 5.9: Example CPDAG and DAG.

The color-code used for the edges is the following:

- Black edges correspond to normal edges, *i.e.,* edges that were only detected by a single algorithm. They are present in both CPDAGs and DAGs, *e.g.,* the $(A \rightarrow B)$ edge from both figures 5.9a and 5.9b.
- Green edges correspond to directed edges that were detected by both the *NOTEARS* and the *FGES* algorithms. They are present in both CPDAGs and DAGs, *e.g.,* the $(D \rightarrow C)$ edge from both figures 5.9a and 5.9b.
- Blue edges correspond to undirected edges and thus are only present in CPDAGs, *e.g.,* the $(D - E)$ edge from figure 5.9a.
- Yellow edges correspond to directed edges that were previously blue undirected edges, thus they are only present in DAGs extracted from CPDAGs, *e.g.,* the $(D \rightarrow E)$ edge from figure 5.9b.

Note that the graph obtained by the *NOTEARS* algorithm will only contain black edges, since:

- The *NOTEARS* algorithm outputs a DAG, therefore it will not contain neither blue nor orange edges.
- It is unwise to highlight directed edges it has in common with the *FGES* algorithm. This is due to the fact the CPDAGs obtained from *FGES* algorithm using different score functions may result in different sets of directed edges in common with *NOTEARS*, therefore it makes no sense to pick a set over the others. For this reason, the DAG obtained by the *NOTEARS* algorithm will not contain green edges.

Over the course of this section we will focus on each data set, giving an overview of its subject matter, detailing the required data transformations and showing the DAG of the learned Bayesian network that was "closest" to the real distribution underlying the data. It will conclude with the results of the learned Bayesian networks obtained by the various Bayesian network structure learning algorithms and some final remarks on these same results.

5.2.1 Cardiovascular Disease Data Set

This data set was obtained from *Kaggle* [56]. It is comprised of $70,000$ records of patient data collected at the moment of a medical examination. Each record notes eleven features, which can be classified as either being:

- *Objective*, when expressing factual information. This type of features includes:
  - Age, measured in days.

  - Height, measured in cm.
  - Weight, measured in kg.
  - Gender, classified as:
    * Male.
    * Female.
- *Examination*, when expressing results of the medical examination. This type of features includes:
  - Systolic blood pressure (AP_{hi}), measured in mmHg
  - Diastolic blood pressure (AP_{lo}), measured in mmHg
  - Cholesterol levels, classified as:
    * Normal.
    * Above normal.
    * Well above normal.
  - Glucose levels, classified as:
    * Normal.
    * Above normal.
    * Well above normal.
- *Subjective*, when expressing information given by the patient. This type of features includes:
  - Smoking, whether the patient smokes or not.
  - Alcohol intake, whether the patient drinks alcohol beverages or not.
  - Physical activity, whether the patient is physically active or not.

Based on these features, the goal is to diagnose the patient in regards to whether or not he suffers from cardiovascular disease.

Feature transformation

The features that measure continuous values were discretized in the following manner:

- Age: First it was converted into years and then classified according to table 5.11.

Table 5.11: Age discretization.

Age	Age [years]
Young Adult	$[20, 29]$
Adult	$[30, 39]$
Middle Age	$[40, 59]$
Early Elder	$[60, 79]$

- Height/Weight: Combined into Body Mass Index (BMI), according to equation 5.1, then classified according to table 5.12.

$$\text{BMI} = \frac{\text{Weight}_{kg}}{\text{Height}_{m}^{2}} \tag{5.1}$$

Table 5.12: Body Mass Index discretization.

BMI	BMI
Underweight	< 18.5
Normal Veight	$[18.5, 25[$
Overweight	$[25, 30[$
Obese	≥ 30

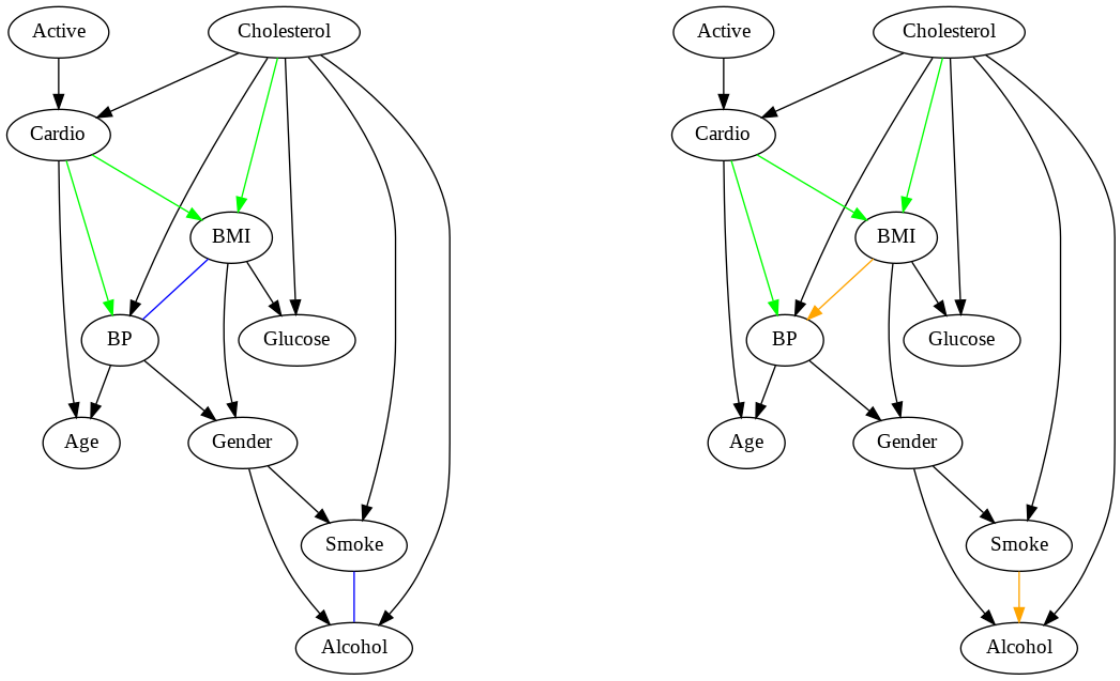
- Systolic blood pressure/Diastolic blood pressure: Combined into Blood Pressure (BP), which is classified according to table 5.13.

Table 5.13: Blood pressure discretization.

BP	AP_{hi}	Relationship	AP_{lo}
Normal	< 120	$\wedge$	< 80
Elevated	$[120, 130[$	$\wedge$	< 80
Hypertension 1	$[130, 140[$	$\vee$	$[80, 90[$
Hypertension 2	$[140, 180[$	$\vee$	$[90, 120[$
Hypertensive Crisis	≥ 180	$\vee$	≥ 120

Learned Bayesian network

The algorithm/score function pair that resulted in the Bayesian network that best fit the observed data was the *FGES* algorithm using the *degen-gauss-bic* score function, see figure 5.10.



(a) CPDAG (b) DAG

Figure 5.10: Obtained CPDAG by the *FGES* algorithm using the *degen-gauss-bic* score function and the extracted DAG for the *Cardio* data set.

The learned CPDs allow us to analyze the influence that parent nodes have on their child node. We can find out how much a specific parent influences its child node by marginalizing the latter's CPD of all other parent nodes. For instance, regarding the node:

- *BMI* – We find that if a patient is diagnosed with cardiovascular disease, then it is much more likely that his *BMI* is above normal weight, rather than being underweight, as seen in table 5.14. We also discover that when considering patients with higher levels of cholesterol, the chances that they are underweight or have a normal *BMI* decreases, while the opposite is true for obese patients, as seen in table 5.15.

Table 5.14: Marginalized CPD of node *BMI* for analyzing $P(\textit{BMI} \mid \textit{Cardio})$.

$P(\textit{BMI} \mid \textit{Cardio})$	*BMI* = Underweight	*BMI* = Normal weight	*BMI* = Overweight	*BMI* = Obese
Cardio = False	0.011	0.374	0.35	0.265
Cardio = True	0.005	0.259	0.362	0.374

Table 5.15: Marginalized CPD of node *BMI* for analyzing $P(\textit{BMI} \mid \textit{Cholesterol})$.

$P(\textit{BMI} \mid \textit{Cholesterol})$	*BMI* = Underweight	*BMI* = Normal weight	*BMI* = Overweight	*BMI* = Obese
Cholesterol = Normal	0.01	0.395	0.362	0.233
Cholesterol = Above normal	0.009	0.285	0.371	0.335
Cholesterol = Well above normal	0.005	0.268	0.336	0.391

- *BP* – From tables 5.16 and 5.17, we find that most of the patients' blood pressure is on the Hypertension 1 level, seeing as how it barely changes regardless of whether we know their *BMI* or if they have cardiovascular disease. Meanwhile, the chances that they are in a Hypertensive Crisis are slim, also remaining relatively unaffected by knowing their *BMI*, though there is a significant increase if we observe that the patient has cardiovascular disease. The observations for these specific *BP* levels on table 5.16 likely indicate the bias of the data collection process. This data set contains the records of patient data collected at a medical examination, so it is possible that healthier individuals would not feel the need to attend, thus skewing the data towards patients with medical problems. On the other hand, individuals in a Hypertensive crisis would likely be examined at a hospital's emergency visit, rather than a simple routine checkup.

Table 5.16: Marginalized CPD of node *BP* for analyzing $P(\textit{BP} \mid \textit{BMI})$.

$P(\textit{BP} \mid \textit{BMI})$	*BP* = Normal	*BP* = Elevated	*BP* = Hypertension 1	*BP* = Hypertension 2	*BP* = Crisis
BMI = Underweight	0.328	0.042	0.474	0.149	0.007
BMI = Normal weight	0.161	0.052	0.599	0.185	0.003
BMI = Overweight	0.114	0.038	0.589	0.255	0.004
BMI = Obese	0.079	0.026	0.552	0.335	0.008

Table 5.17: Marginalized CPD of node *BP* for analyzing $P(\textit{BP} \mid \textit{Cardio})$.

$P(\textit{BP} \mid \textit{Cardio})$	*BP* = Normal	*BP* = Elevated	*BP* = Hypertension 1	*BP* = Hypertension 2	*BP* = Crisis
Cardio = False	0.262	0.045	0.575	0.116	0.002
Cardio = True	0.078	0.034	0.533	0.346	0.009

- *Cardio* – From table 5.18 we find that a patient's active lifestyle has a very small positive effect on his chance of having a cardiovascular disease. On the other hand, the higher his cholesterol levels, the higher his chance of having cardiovascular disease. This is in line with medical literature [57]–[59], where proper nutrition is pointed out as the most important factor in regard to preventing heart disease.

Table 5.18: Marginalized CPD of node *Cardio* for analyzing $P(\textit{Cardio} \mid \textit{Active})$.

$P(\textit{Cardio} \mid \textit{Active})$	*Cardio* = False	*Cardio* = True
Active = False	0.374	0.626
Active = True	0.403	0.597

Table 5.19: Marginalized CPD of node *Cardio* for analyzing $P(\textit{Cardio} \mid \textit{Cholesterol})$.

$P(\textit{Cardio} \mid \textit{Cholesterol})$	*Cardio* = False	*Cardio* = True
Cholesterol = Normal	0.543	0.457
Cholesterol = Above normal	0.383	0.617
Cholesterol = Well above normal	0.241	0.759

5.2.2 Acute Inflammation Data Set

This data set was obtained from the *UCI Machine Learning Repository* [60]. It was created by a medical expert as a data set to test the expert system, which will perform the presumptive diagnosis of two diseases of the urinary system [61]:

- Acute inflammation of the urinary bladder.
- Acute nephritis of renal pelvis origin.

Acute inflammation of the urinary bladder is characterized by a sudden occurrence of pains in the abdomen region and the urination in form of constant urine pushing, micturition pains and a higher temperature of the body, though generally not above $38\ ^{\circ}$C.

Acute nephritis of renal pelvis origin is characterized by a high fever, sometimes exceeding $40\ ^{\circ}$C, accompanied by shivers and lumbar pains. The symptoms of acute inflammation of the urinary bladder are also common, while nausea and burning of urethra are less frequent symptoms.

Each entry in the data set represents a potential patient and is comprised of eight attributes, which can be split into:

- *Symptoms*:
  - Temperature of the patient, measured in degrees Celsius (°C) and ranging from $[35, 42]$.
  - Occurrence of nausea, either yes or no.
  - Lumbar pain, either yes or no.
  - Urine pushing, *i.e.,* whether the patient has a continuous need for urination.
  - Micturition pains, either yes or no.
  - Burning of urethra, either yes or no.
- *Diagnosis*:
  - Inflammation of urinary bladder, either yes or no.
  - Nephritis of renal pelvis origin, either yes or no.

Feature transformation

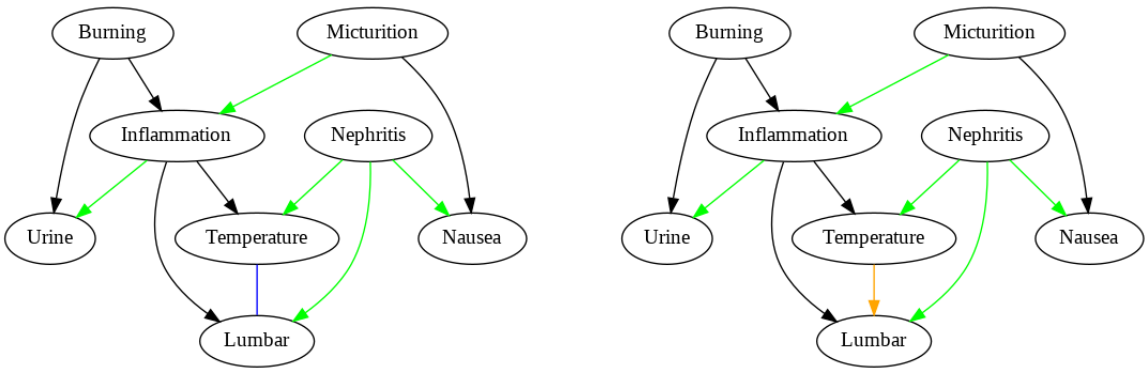
Most data set features can be described as boolean variables, except for the patient's temperature which requires discretizing according to table 5.20.

Table 5.20: Temperature discretization.

Temperature	Temperature [°C]
Low	< 36
Normal	$[36, 37.5]$
Fever	$[37.5, 40]$
High Fever	$[40, 42]$

Obtained Network Structure

The algorithm/score function pair that resulted in the Bayesian network that best fit the observed data was the *Reg-FGES* algorithm using the *bdeu-score* as the scoring function, see figure 5.11.



(a) CPDAG (b) DAG

Figure 5.11: Obtained CPDAG by the *Reg-FGES* algorithm using the *bdeu-score* as the scoring function and the extracted DAG for the *Diagnosis* data set.

Similarly to the analysis that was made for the previous data set on section 5.2.1, we can inspect the marginalized CPDs of certain nodes in order to extract specific observations made possible by the learned Bayesian networks. For instance, regarding the node:

- *Temperature* – From table 5.21, we find that patients that have been diagnosed with nephritis of renal pelvis origin are very likely to have abnormally high temperatures, which is in line with the expert knowledge provided by the creator of the data set. However, on table 5.22, we find that patients diagnosed with acute inflammation of the urinary bladder have a 50% chance of having a high fever, while the expert knowledge indicates that temperatures above 38^{o}C are uncommon.

Table 5.21: Marginalized CPD of node *Temperature* for analyzing $P($*Temperature* | *Nephritis*$)$.

$P($*Temperature* | *Nephritis*$)$	*Temp* = Low	*Temp* = Normal	*Temp* = Fever	*Temp* = High fever
Nephritis = False	0.046	0.504	0.283	0.167
Nephritis = True	0.0	0.0	0.161	0.839

Table 5.22: Marginalized CPD of node *Temperature* for analyzing $P($*Temperature* | *Inflammation*$)$.

$P($*Temperature* | *Inflammation*$)$	*Temp* = Low	*Temp* = Normal	*Temp* = Fever	*Temp* = High fever
Inflammation = False	0.033	0.217	0.245	0.505
Inflammation = True	0.012	0.288	0.2	0.5

- *Inflammation* – From table 5.23, we find that patients that haven't reported experiencing micturition pains are unlikely to be diagnosed with acute inflammation of the urinary bladder, which is in line with the knowledge provided by the medical expert, since it is one of the symptoms of this disease. On the other hand, the learned Bayesian network identifies the relationship ($\textit{Burning} \rightarrow \textit{Inflammation}$), while the expert knowledge states that the burning of the urethra is a less frequent symptom of the same disease, which is not in line with the results on table 5.24.

Table 5.23: Marginalized CPD of node *Inflammation* for analyzing $P($*Inflammation* | *Micturition*$)$.

$P($*Inflammation* | *Micturition*$)$	*Inflammation* = False	*Inflammation* = True
Micturition = False	0.875	0.125
Micturition = True	0.167	0.833

Table 5.24: Marginalized CPD of node *Inflammation* for analyzing $P($*Inflammation* | *Micturition*$)$.

$P($*Inflammation* | *Burning*$)$	*Inflammation* = False	*Inflammation* = True
Burning = False	0.542	0.458
Burning = True	0.5	0.5

5.2.3 Marital Depression Data Set

This data set was obtained from *Kaggle* [62]. It is comprised of the answers of married individuals from Istanbul to an online form, which aims to examine the influence of certain demographic factors on depression. In each form, an individual inputs his personal information and then answers the Beck Depression Inventory, which is a 21-question multiple-choice questionnaire widely used for measuring the severity of depression, focusing on the individual's thought regarding certain statements [63]. For a specific question, each possible answer has a specific score, ranging from $[0, 3]$. The overall score of the questionnaire is the sum of the scores of the answers for each question.

Each entry in the data set represents a married individual from Istanbul and is comprised of 27 features, which can be split into:

- *Information*:
  - Survey key, which is used to identify each specific individual without encroaching on their privacy.

  - Gender, which can be either:
    - Male.
    - Female.
  - Education, which can be either:
    - Primary.
    - High school.
    - Bachelor.
    - MsC/PhD.
  - Working status, which can be either:
    - Employed.
    - Unemployed.
  - Marriage style, which can be either:
    - Arranged marriage.
    - Courtship marriage.
  - Child, which identifies whether the individual has children or not.
- *Questionnaire*, for the individual scores of the answers each of the 21 questions.

Feature transformation

Since we are interested in analyzing the final score of the Beck Depression Inventory, all of the data set's *questionnaire-type* features were combined into a single feature, referred to as BDI

$$\text{BDI} = \sum_{i=1}^{21} \text{bdi}_i \,, \tag{5.2}$$

where bdi_i is the score of the answer given by a specific individual to the i-th question of the Beck Depression Inventory questionnaire.

After computing the BDI score, we can characterize the individual's level of depression according to table 5.25.

Table 5.25: BDI score discretization.

BDI	BDI
Normal	$[0, 10]$
Mild Mood Disturbance	$[11, 16]$
Borderline Clinical Depression	$[17, 20]$
Moderate Depression	$[21, 30]$
Severe Depression	$[31, 40]$
Extreme Depression	> 40

Note also that the Survey key feature was dropped, since is not particularly interesting for our purposes of learning a Bayesian Network from the data.

Obtained Network Structure

For this data set, it was the *NOTEARS* algorithm that resulted in the Bayesian network that best fit the observed data, see figure 5.12. This is surprising, since for all data sets generated from well-known Bayesian networks in section 5.1, this algorithm was by far the one with the weakest performance on the used metrics.

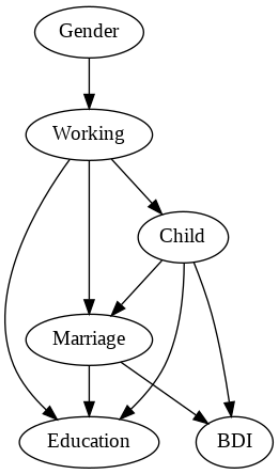


Figure 5.12: Obtained DAG by the *NOTEARS* algorithm for the *Marriage* data set.

Note that no edge is highlighted green due to the *FGES* algorithm using the various score functions having detected different edges between themselves.

However, it is important to point out that both *FGES/bdeu-score* and *FGES/disc-bic-score* detected the (*Gender* $\rightarrow$ *Working*) edge, which is also present in figure 5.12. This edge appears to be important and adjusted to reality, for this data set involves married individuals from Istanbul, Turkey. Here, a traditionally Muslim country, only 30.3% of women are part of the workforce, the lowest in the OCDE and one of the lowest in the world [64]. This value drops to 20.5% when considering only married women, which is the case for all females involved in the data set.

Similarly to the analysis that was made for the previous data sets on sections 5.2.1 and 5.2.2, we can inspect the marginalized CPDs of certain nodes in order to extract specific observations made possible by the learned Bayesian networks. For instance, regarding the node:

- *Gender* – The values on table 5.26 show that the individuals that participated on the questionnaire are not representative of the population. This is due to the fact that observing that a given participant is female has small bearing on the chances of her being employed, which is contrary to the expert knowledge stated above when analyzing just the obtained DAG. Meanwhile, if the individual is male there is a high likelihood that he is unemployed. Similarly to section 5.2.1, this is likely an indicative of the bias inherent to the data collection process. Since Turkey is a highly patriarchal society, it is possible that the over-representation of unemployed males in a questionnaire used to measure depression in married couples is due to their own perceived failure in meeting society's expectations.

Table 5.26: Marginalized CPD of node *Working* for analyzing $P($*Working* $|$ *Gender*$)$.

$P($*Working* $|$ *Gender*$)$	*Working* $=$ Unemployed	*Working* $=$ Employed
Gender $=$ Female	0.55	0.45
Gender $=$ Male	0.927	0.073

- *BDI* – Tables 5.27 and 5.28 display that observing whether a specific individual has a child shows a similar effect on the *BDI* score as observing whether that person's marriage was arranged or not. This is in line with expert knowledge which states that depression affects people from all walks of life and is caused by a multitude of factors [65].

Table 5.27: Marginalized CPD of node *BDI* for analyzing $P($*BDI* $|$ *Child*$)$.

$P($*BDI* $|$ *Child*$)$	*BDI* $=$ Normal	*BDI* $=$ Mild	*BDI* $=$ Borderline	*BDI* $=$ Moderate	*BDI* $=$ Severe	*BDI* $=$ Extreme
Child $=$ No	0.639	0.118	0.063	0.164	0.008	0.008
Child $=$ Yes	0.599	0.189	0.075	0.112	0.025	0.0

Table 5.28: Marginalized CPD of node *BDI* for analyzing $P(\textit{BDI} \mid \textit{Marriage})$.

$P(\textit{BDI} \mid \textit{Marriage})$	*BDI* = Normal	*BDI* = Mild	*BDI* = Borderline	*BDI* = Moderate	*BDI* = Severe	*BDI* = Extreme
Marriage = Arranged	0.661	0.108	0.064	0.152	0.015	0.0
Marriage = Courtship	0.576	0.198	0.075	0.124	0.019	0.008

5.2.4 Titanic Data Set

This data set was obtained from *Kaggle* [66]. It contains the data of passengers who boarded the *RMS Titanic* which sank after striking an iceberg in the North Atlantic Ocean in 1912. Out of the original 2,201 passengers, only 711 survived [67]. At the time, it became the deadliest peacetime sinking of a cruise ship.

Each entry in the data set represents a passenger and is comprised of 8 features:

1. Survived, which identifies whether the passenger survived the whole ordeal or not.
2. Passenger class (PClass), which can be either:
    - Upper class.
    - Middle class.
    - Lower class.
3. Gender, which can be either:
    - Male.
    - Female.
4. Age, which corresponds to the passenger's age in years.
5. Siblings/Spouse (SibSp), which corresponds to the number of siblings and spouse of the passenger that were also aboard the ship.
6. Parents/Children (ParCh), which corresponds to the number of parents and children of the passenger that were also aboard the ship.
7. Fare, which corresponds to the ticket fare paid by the passenger to board the ship.
8. Embarked, which identifies where the passenger boarded the ship, and can be either:
    - Cherbourg.
    - Queenstown.
    - Southampton.

Feature transformation

The features that measure continuous values were discretized in the following manner:

- Age: The thought-process was to characterize each passenger's age according to his ability to help others and his own frailness. Thus, classifying each passenger according to table 5.29.

Table 5.29: Age discretization.

Age	Age [years]
Child	$[0, 8]$
Youngster	$[8, 16]$
Adult	$[16, 45]$
Middle-Age	$[45, 65]$
Elder	> 65

- Siblings/Spouse and Parents/Children: For both features, the idea was to classify each passenger according to the help they could expect and organization needed for their group during the catastrophe. Thus, passengers traveling alone would only need to fend for themselves, yet would only

be able to count on the help of strangers. On the other hand, passengers traveling in large family groups would have to focus a lot more on their group's integrity, despite having better chances of getting help than solo travelers. Therefore, the passengers hypothesized to have a better chance of surviving would be those traveling in small family groups. Then, each passenger was classified according to table 5.30.

Table 5.30: Siblings/Spouses and Parents/Children discretization.

SibSp/ParCh	SibSp/ParCh
Solo	0
Small Group	$[1, 3]$
Large Group	> 3

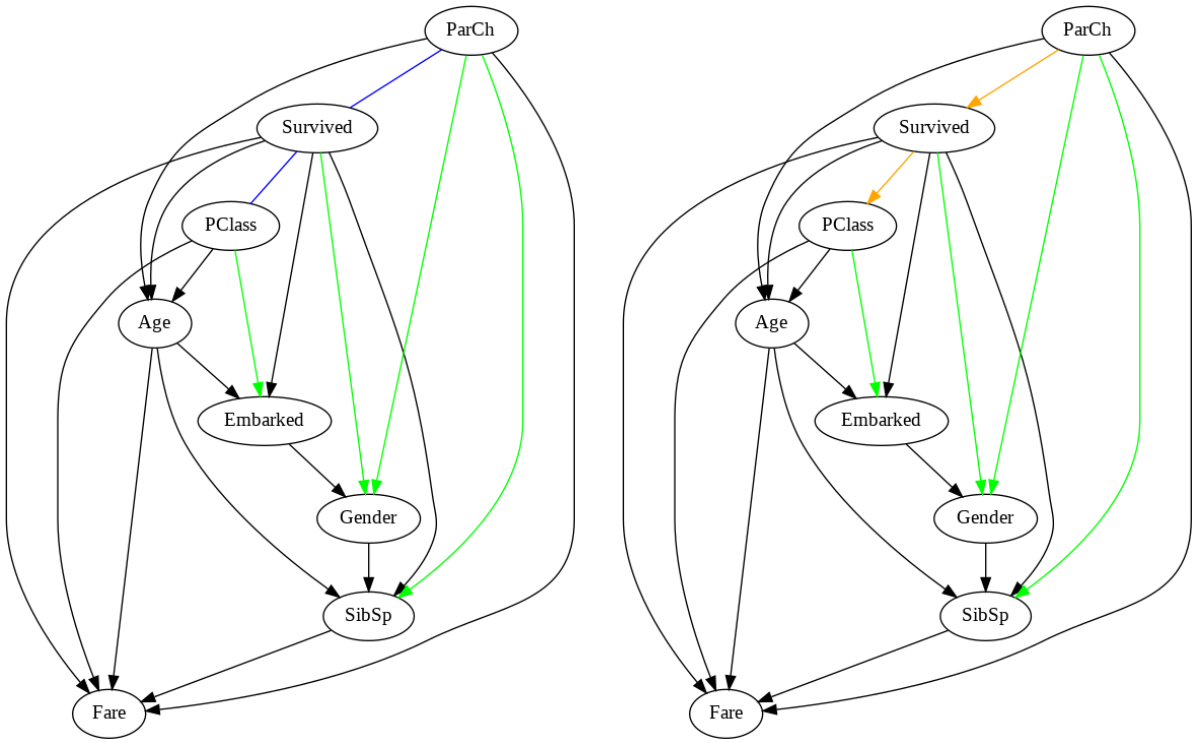
- Fare: According to the data distribution, we can identify the classes of fares paid by passengers seen in table 5.31.

Table 5.31: Fare discretization.

Fare	Fare [GBP]
Normal	< 50
Expensive	$[50, 100]$
Very Expensive	> 100

Obtained Network Structure

The algorithm/score function pair that resulted in the Bayesian network that best fit the observed data was the *FGES* algorithm using the *degen-gauss-bic* score function, see figure 5.13.



(a) CPDAG (b) DAG

Figure 5.13: Obtained CPDAG by the *FGES* algorithm using the *degen-gauss-bic* score function and the extracted DAG for the *Titanic* data set.

Similarly to the analysis that was made for the previous data sets on sections 5.2.1, 5.2.2 and 5.2.3,

we can inspect the marginalized CPDs of certain nodes in order to extract specific observations made possible by the learned Bayesian networks. For instance, regarding the node:

- *Age* – Table 5.32 shows that for ages up until middle-age, the likelihood of passengers that survived being in those age brackets is higher than for passengers that perished. While, the opposite is true for middle-aged and elder passengers.

Table 5.32: Marginalized CPD of node *Age* for analyzing $P(\textit{Age} \mid \textit{Survived})$.

$P(\textit{Age} \mid \textit{Survived})$	*Age* = Child	*Age* = Youngster	*Age* = Adult	*Age* = Middle-age	*Age* = Elder
Survived = False	0.072	0.066	0.516	0.28	0.066
Survived = True	0.166	0.095	0.543	0.134	0.062

- *Gender* – Table 5.33 shows if we know that a passenger survived, then it is much more likely that it is a female, while the opposite is true for male passengers.

Table 5.33: Marginalized CPD of node *Gender* for analyzing $P(\textit{Gender} \mid \textit{Survived})$.

$P(\textit{Gender} \mid \textit{Survived})$	*Gender* = Male	*Gender* = Female
Survived = False	0.593	0.407
Survived = True	0.324	0.676

- *PClass* – Table 5.34 shows that if we know that a passenger perished, then it is much more likely that he was traveling in third class, rather than in a higher class.

Table 5.34: Marginalized CPD of node *PClass* for analyzing $P(\textit{PClass} \mid \textit{Survived})$.

$P(\textit{PClass} \mid \textit{Survived})$	*PClass* = 1st Class	*PClass* = 2nd Class	*PClass* = 3rd Class
Survived = False	0.146	0.177	0.677
Survived = True	0.398	0.254	0.348

In the aftermath of the Titanic catastrophe, there was a great outrage over the fact that wealthier passengers were provided with more assistance than common folk. On the other hand, women and children were also benefited at the cost of the lives of the male passengers [67]. Therefore, the observations obtained from analysis of the marginalized CPDs are in line with the expert knowledge.

5.2.5 Overall Results

The relative entropy is used to measure how close the learned joint probability distribution is to the original distribution, being equal to zero when they are the same, therefore the lower its value the better.

The values of the relative entropy between the joint probability distribution underlying the data set and the joint probability distribution encoded by the Bayesian network learned by the various algorithms and score functions can be found on table 5.35.

Table 5.35: Relative entropy (KL-divergence) values of the distributions encoded the learned Bayesian networks for all data sets.

$\boldsymbol{D(P\|Q)}$	*Cardio*	*Diagnosis*	*Marriage*	*Titanic*
NOTEARS	0.694	0.191	**0.287**	0.765
FGES/bdeu	0.532	0.226	0.49	0.774
FGES/db	0.515	0.356	0.491	0.845
FGES/dgb	**0.512**	0.279	0.493	**0.67**
R-FGES/bdeu	0.532	**0.185**	0.49	0.783
R-FGES/db	0.528	0.356	0.491	0.845
R-FGES/dgb	0.69	0.273	0.493	0.678

The obtained results show:

- When using the same score function, the *FGES* and *Reg-FGES* algorithms generally achieve similar relative entropy values, except for the *degen-gauss-bic* score function on the *Cardio* data set.
- Unlike with the generated data sets of section 5.1 and apart from the *Cardio* data set, the *NOTEARS* algorithm generally proved to be competitive with the other algorithms, even achieving the best result on the *Marriage* data set.
- By comparing the best results for each data set, we find that the Bayesian networks obtained for the *Diagnosis* and *Marriage* data sets fit their respective observed data better than the ones obtained for the *Cardio* and *Titanic* data sets.

It is important to note that all used data sets are conducive to *classification* tasks, *i.e.,* where we want to predict the values of specific random variables based on knowing the values of the rest. These random variables are known as *decision variables*. For instance, consider a system for medical diagnosis, where the variables that represent specific symptoms would then influence the decision variable, which represents the diagnosis.

For causal inference, *i.e.,* determining causal links, the direction of the edges would then be from the other nodes to the decision node. However, barring the *Marriage* data set, on the DAGs obtained for the various data sets, the decision nodes were more akin to the root node of the naive Bayes network, see figure 2.6. In other words, the edges are outgoing from the decision nodes, therefore the various algorithms proved to be inadequate for causal inference.

In order to interpret the meaning of the edges in the obtained DAGs, we should think of $X \to Y$ not as "X *causes* Y", but instead as "knowing X *influences* Y".

To illustrate this, consider the DAG in figure 5.13b. This graph was obtained for the *Titanic* data set, where the decision variable is represented by the node *Survived*. Consider its outgoing edges, in particular, the ($Survived \to Age$) and ($Survived \to Gender$) edges. It should be obvious that whether a passenger survived or not does not change his age or gender. What the Bayesian network is actually implying is that, for instance, *knowing* that a passenger survived increases the likelihood of the passenger being of a certain age and gender. The code of conduct used to determine whose lives should be saved with the limited available lifeboats was "women and children first". This resulted in the survival of 74.35% of the women and 52.29% of the children, while only 20.27% of the men survived [67].

The same type analysis can be done for the decision variables of the rest of the data sets, which can be considered an advantage probabilistic graphical models have over neural networks, the latter being much more opaque in their reasoning.

Chapter 6

Conclusions

This chapter concludes the developed work, providing some observations on the subject matter of the preceding chapters and suggesting future lines of research.

6.1 Closing Thoughts

This work addressed the problem of learning Bayesian networks from observational data alone, focusing especially in the Bayesian network structure learning sub-problem. As we saw in chapter 3, the number of possible DAGs grows super-exponentially as the number of nodes increases, see table 3.1.

In order to deal with the combinatory nature of the search-space, traditional algorithms seek to reduce its size so as to speed-up the search, which comes at the cost of not guaranteeing the optimality of the obtained solution. On the other hand, the recent continuous optimization approach followed by the *NOTEARS* algorithm [10] is subject to the pitfalls of non-convex optimization, therefore it is also not optimal. In spite of this, the results obtained in chapter 5 show that these algorithms still achieve worthwhile solutions.

In order to extract a DAG from the CPDAG obtained by the *FGES* or *Reg-FGES* algorithms, we used the method described in section 4.3. This method is random in nature, which leads to possibly different output DAGs for the same input CPDAG.

In chapter 4, we formulated the hypothesis that certain relationships are so strongly evident in the data, that they are present across the network structures learned by different Bayesian network structure learning algorithms. Later, in section 5.1, we showed that this hypothesis appeared to be valid, see table 5.2.

Motivated by these results, in section 4.1, we proposed a meta-algorithm that combined the new Bayesian network structure learning approach established by the *NOTEARS* algorithm with the state-of-the-art score-based approach *FGES* algorithm. This meta-algorithm used the directed edges common to the graphs obtained by the *NOTEARS* and *FGES* algorithms as prior knowledge and then applied the *FGES* algorithm again, now using this set of edge as its starting graph.

The results in section 5.1 show that the proposed solution consistently outperformed *NOTEARS* and achieved on-par results *FGES* on the smaller sized networks. However, for larger networks it achieved slightly worse results, likely due to local optima of the used score-function and the randomness of the method used to extract a DAG from the obtained CPDAG.

While the *Reg-FGES* did not supplant the *FGES* algorithm, the directed edges common to the graphs obtained by both the *NOTEARS* and *FGES* algorithms were shown to be very likely to be correct edges, as seen in tables 5.5, 5.6 and 5.7.

In section 5.1, the traditional metrics we used show that the *FGES* and *Reg-FGES* algorithms achieved better DAGs than the *NOTEARS* algorithm, as can be seen in figures 5.5, 5.6, 5.7, 5.8. For instance, in table 5.3 we see that the former algorithms achieved DAGs with smaller SHD values than their number of edges, therefore they unequivocally identified correct relationships between the random variables.

While in section 5.2, we found that despite *NOTEARS* being the worst-performing algorithm when it came to Bayesian networks structure learning, it still managed to learn the Bayesian network that encoded the probability distribution closest to the real distribution underlying the observed data for the *Marriage* data set 5.2.3.

We also noted that, the Bayesian networks learned in section 5.2 are inadequate for inference tasks. This is due to the use of *MLE* for Bayesian network parameter estimation, which overfits the models to the data sets, as discussed in section 2.3.3. Therefore, these learned Bayesian networks do not generalize well to unseen data. Since our goal was to evaluate Bayesian network structure learning algorithms, this type of parameter estimation was the best-suited for it. However, if we want to use the learned models for inference further downstream, then we should use Bayesian parameter estimation once we have obtained the various Bayesian network structures.

As discussed in section 2.3, one of the advantages of Bayesian networks is that they are able to represent causal relationships. However, the results we obtained in section 5.2 show that the edges in the learned Bayesian networks do not necessarily represent causal relationships. When learning a Bayesian network from observational data alone, we generally found that the direction of certain edges of the learned graph structures was the opposite of the expected causal direction.

For instance, for the *Cardio* data set of section 5.2.1, the learned Bayesian network structure contains the edge ($\textit{Cardio} \rightarrow \textit{Age}$), as seen in figure 5.10b. From a causality viewpoint, we would expect the reverse edge ($\textit{Age} \rightarrow \textit{Cardio}$) since a patient's age is not influenced by whether or not they suffer from cardiovascular disease, while the older a person is the more likely they are to be diagnosed this disease [58].

Despite the discussed Bayesian network structure learning algorithms proving to be insufficient for causal reasoning, they still offer valuable insight on the nature of the data sets by identifying important relationships between the random variables. As mentioned in section 5.2.5, the edges in the learned Bayesian network structure should be interpreted as knowing the value of the source influences the value of the target.

Furthermore, the analysis conducted on section 5.2 demonstrates an advantage that Bayesian networks (and probabilistic graphical models, in general) have over neural networks. The former's graphical nature allows us to understand the learned model's reasoning, whereas for the latter this is not trivial.

On the next and final section, we will also offer possible lines of future work that should enhance the proposed meta-algorithm and the developed methods.

6.2 Future Work

While developing the proposed solution presented in chapter 4, several choices were made that left unexplored avenues.

First, let us consider the possible changes involving the meta-algorithm proposed in section 4.1.

On the one hand, we chose to use the *FGES* algorithm due to previous research showing that it achieves better results than the alternatives that were discussed in chapter 3 [48]. However, it would be interesting to apply the meta-algorithm using other Bayesian network structure learning algorithms.

A possible reason for the similarity of the results achieved by *FGES* and *Reg-FGES* is that the search method both use already achieves relatively good results, being the state-of-the-art of traditional approaches. Thus, it is possible that for a worse performing traditional algorithm, regularizing it using the meta-algorithm would result in a noticeable improvement in performance. Note that there is a wide array of algorithms that belong to the traditional approaches, but for the sake of brevity we focused on the most renowned.

On the other hand, the meta-algorithm joins a traditional score-based structure learning algorithm, *FGES*, with the modern continuous optimization approach of the *NOTEARS* algorithm. However, as described in section 4.1.1, the latter is clearly the most computationally costly step of the meta-algorithm. Since its inception, several improved versions of this algorithm have been proposed, managing to alter the acyclicity constraint so that the algorithm's complexity becomes $\mathcal{O}(n^2)$. For a recent review of *NOTEARS*' variants and other algorithms that fall into the continuous optimization approach, see [22].

There are also alternatives to the methods described in sections 4.3 and 4.4.

As was previously mentioned, using the method described in section 4.3 to extract a DAG from the obtained CPDAG does not guarantee the DAG that best fits the data, due to the method's random nature. Though DAGs encoded by the same CPDAG are score-equivalent and encode the same *d-separation* facts, their causal implications are not the same, *e.g.,* $X - Y$ encodes both $X \rightarrow Y$ and $X \leftarrow Y$, which from a causal inference perspective means X *causes* Y and Y *causes* X, respectively.

Generally, it is not trivial to induct causal directionality from observation data alone [22]. However, for particular scenarios where certain assumptions on the nature of the real distribution underlying the data hold, there are methods that explore asymmetries in order to identify the direction of a structural relationship. These methods are *local* in nature, seeing as how they can only test a single edge at a time, *i.e.,* bivariate networks [46], or variable triplets, where the third variable is latent, *i.e., unobserved* [68]. These methods can be extended to construct full graphs by iteratively testing pairwise relationships [69]. Therefore, one possible alternative to extract a DAG from a CPDAG would be to use these methods to assign a direction to the undirected edges. However, their applicability is limited to CPDAGs with a relatively small number of undirected edges, due to their computational complexity. For an extensive review of these methods, see [70].

Regarding the method used to evaluate the DAGs learned from the real data sets described in section 4.4, while adequate to our purposes, it still only provides a rough estimate of the closeness distribution encoded by the learned Bayesian network to the real distribution underlying the data. An alternative path would be to use the state-of-the-art exact solver [41] to obtain a DAG that we would then consider as the real DAG. This way we could use the same metrics described in section 4.2, performing a similar type of analysis as the one that was done for the generated data sets in section 5.1. However, this approach is only possible for relatively small networks, since exact solvers do not scale to domains with a large number of variables [40].

Yet another way to evaluate and compare the DAGs obtained by the various Bayesian network structure learning algorithms would be to use *cross-validation*. This would have us split the data set into a:

- *Training set*, which would be used to learn the full Bayesian network, *i.e.,* both the network's structure and its parameters.
- *Testing set*, which would be used for inference on the learned Bayesian network. For each sample, we would find the value of a decision variable, Y, based on the values of the rest of the feature variables, $\boldsymbol{X}$. Then, we would measure the accuracy of the values inferred by the Bayesian network by comparing them to the values of Y in the *testing set*. These accuracy values would then allow us to evaluate and compare the DAGs obtained by the various algorithms.

Note that instead of using MLE to estimate the network's parameters, we would have to use *Bayesian parameter estimation* with a uniform *prior*, otherwise the learned Bayesian networks would be *overfit* to the *training set*, as discussed in section 2.3.3. This would manifest into worse results when evaluating the accuracy of the inferences of the learned Bayesian network on the *testing set*.

In summary, the proposed lines of future work are:

- With the aim of achieving markedly better results using the regularized form of the traditional algorithm as opposed to its original form, **extend the meta-algorithm by using other traditional Bayesian network structure learning algorithms**, such as the ones mentioned in chapter 3 (*e.g., PC* [9], *MMHC* [36]) or others (*e.g., LiNGAM* [71], *FCI* [9]), instead of just using the *FGES* algorithm.
- In order to reduce the computational complexity of the meta-algorithm, **use an improved version of *NOTEARS* or any other continuous optimization Bayesian network structure learning algorithm**, see [22] for a recent review of algorithms that follow this approach.
- To overcome the random nature of the method used for extracting a DAG from the CPDAG obtained by the *FGES* and *Reg-FGES* algorithms, **use causal directionality discovery methods to assign a direction to the undirected edges**, see [70] for an extensive review of these methods.
- For real data sets, where the underlying distribution is unknown, **use as ground truth the DAG obtained by the exact solver, *GOBNILP*** [41]. This would allow the use of the standard metrics

(*ACC, PPV, TPR, F_1-Score, SHD*) to compare the DAGs obtained by the various Bayesian network structure learning algorithms, similarly to the type of analysis made on section 5.1.

- Another alternative to evaluate and compare the obtained DAGs for real data sets would be to **use *cross-validation*, learning the full Bayesian network from the *training data* and using the *testing data* to find the accurary of the learned models**.

Bibliography

[1] B. Abramson, J. Brown, W. Edwards, A. Murphy, and R. Winkler, "Hailfinder: A Bayesian system for forecasting severe weather", *International Journal of Forecasting*, vol. 12, no. 1, pp. 57–71, 1996.

[2] I. Beinlich, H. Suermondt, R. Chavez, and G. Cooper, "The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks", in *In Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, Springer-Verlag, 1989, pp. 247–256.

[3] C. J. Needham, J. R. Bradford, A. J. Bulpitt, and D. R. Westhead, "A Primer on Learning Bayesian Networks for Computational Biology", *PLOS Computational Biology*, vol. 3, no. 8, pp. 1–8, Aug. 2007. DOI: 10.1371/journal.pcbi.0030129.

[4] M. Beaumont and B. Rannala, "The Bayesian Revolution in Genetics", *Nature Reviews Genetics*, vol. 5, pp. 251–261, 2004. DOI: 10.1038/nrg1318.

[5] M. Voronenko, D. Nikytenko, J. Krejci, O. Naumov, N. Savina, E. Topalova, V. Filippova, and V. Lytvynenko, "Dynamic Bayesian Networks Application for Economy Competitiveness Situational Modeling", in *Advances in Intelligent Systems and Computing V*, vol. 1293, Springer International Publishing, 2021. DOI: 10.1007/978-3-030-63270-0_14.

[6] P. Whitney, A. White, S. Walsh, A. Dalton, and A. Brothers, "Bayesian Networks for Social Modeling", *Social Computing, Behavioral-Cultural Modeling and Prediction*, Lecture Notes in Computer Science, vol. 6589, 2011. DOI: 10.1007/978-3-642-19656-0_33.

[7] R. M. A. Valdés, V. F. G. Comendador, A. R. Sanz, E. S. Ayra, J. A. P. Castán, and L. P. Sanz, "Bayesian Networks for Decision-Making and Causal Analysis under Uncertainty in Aviation", *Bayesian Networks - Advances and Novel Applications*, 2018. DOI: 10.5772/intechopen.79916.

[8] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, 1st ed., ser. Adaptive Computation and Machine Learning Series. The MIT Press, 2009.

[9] P. Sprites, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, 2nd ed., ser. Adaptive Computation and Machine Learning. The MIT Press, 2001.

[10] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing, "DAGs with NO TEARS: Continuous Optimization for Structure Learning", in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018. arXiv: 1803.01422 [cs.LG].

[11] M. Kaiser and M. Sipos, *Unsuitability of NOTEARS for Causal Graph Discovery*, 2021. arXiv: 2104.05441 [stat.ML].

[12] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2003.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[14] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Computation and Machine Learning. The MIT Press, 2012.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning. The MIT Press, 2016.

[16] J. Peters, D. Janzing, and B. Schölkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*, ser. Adaptive Computation and Machine Learning. The MIT Press, 2017.

[17] C. Glymour, K. Zhang, and P. Spirtes, “Review of Causal Discovery Methods Based on Graphical Models”, *Frontiers in Genetics*, vol. 10, p. 524, 2019, ISSN: 1664-8021. DOI: 10.2289/fgene.2019.00524.

[18] D. Margaritis, “Learning Bayesian Network Model Structure from Data”, Ph.D. dissertation, Carnegie Mellon University, May 2003.

[19] M. F. Bari, “Bayesian Network Structure Learning”, in *Proceedings of the 4th Annual Meeting of Asian Association for Algorithms and Computation*, Jan. 2011.

[20] W. Zucchini, “An Introduction to Model Selection”, *Journal of Mathematical Psychology*, vol. 44, no. 1, pp. 41–61, 2000. DOI: https://doi.org/10.1006/jmps.1999.1276.

[21] D. Madigan and A. E. Raftery, “Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam’s Window”, *Journal of the American Statistical Association*, vol. 89, pp. 1535–1546, 1994.

[22] M. J. Vowels, N. C. Camgoz, and R. Bowden, *D’ya like DAGs? A Survey on Structure Learning and Causal Discovery*, 2021. arXiv: 2103.02582 [cs.LG].

[23] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd ed. Cambridge University Press, 2009.

[24] N. Shajoonnezhad and A. Nikanjam, *A Sparse Structure Learning Algorithm for Bayesian Network Identification from Discrete High-Dimensional Data*, 2021. arXiv: 2108.09501 [cs.LG].

[25] W. Lam and F. Bacchus, “Learning Bayesian Belief Networks: An Approach Based on the MDL Principle”, *Computational Intelligence*, vol. 10, no. 3, pp. 269–293, 1994. DOI: 10.1111/j.1467-8640.1994.tb00166.x.

[26] Z. Liu, B. Malone, and C. Yuan, “Empirical Evaluation of Scoring Functions for Bayesian Network Model Selection”, *BMC Bioinformatics*, vol. 13, S14, Sep. 2012. DOI: 10.1186/1471-2105-13-S15-S14.

[27] C. P. Robert, N. Chopin, and J. Rousseau, “Harold Jeffreys’s Theory of Probability Revisited”, *Statistical Science*, vol. 24, no. 2, May 2009, ISSN: 0883-4237. DOI: 10.1214/09-sts284.

[28] D. Heckerman, D. Geiger, and D. M. Chickering, “Learning Bayesian Networks: The Combination of Knowledge and Statistical Data”, *Machine Learning*, vol. 20, pp. 197–243, 2004.

[29] W. L. Buntine, “Theory Refinement on Bayesian Networks”, in *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, Morgan Kauffman, 1991, pp. 52–60. DOI: https://doi.org/10.1016/B978-1-55860-203-8.50010-3.

[30] T. Silander, P. Kontkanen, and P. Myllymaki, “On Sensitivity of the MAP Bayesian Network Structure to the Equivalent Sample Size Parameter”, in *Proceedings of the 23rd Conference on Uncertainty in Machine Learning*, 2012, pp. 360–367. arXiv: 1206.5293 [cs.LG].

[31] B. D. McKay, F. E. Oggier, G. F. Royle, N. J. A. Sloane, I. M. Wanless, and H. S. Wilf, “Acyclic Digraphs and Eigenvalues of 0,1 Matrices”, *Journal of Integer Sequences*, vol. 7, pp. 1–5, 2003.

[32] D. Heckerman, “A Tutorial on Learning with Bayesian Networks”, in *Innovations in Bayesian Networks: Theory and Applications*. Springer Berling Heidelberg, 2008, pp. 33–82. DOI: 10.1007/978-3-540-85066-3_3.

[33] M. Chickering, “Optimal Structure Identification with Greedy Search”, *Journal of Machine Learning Research*, vol. 3, pp. 507–554, 2002.

[34] C. Meek, “Causal Inference and Causal Explanation with Background Knowledge”, in *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, ser. UAI’95, Morgan Kaufmann Publishers Inc., 1995, pp. 403–410.

[35] S. E. Fienberg, *The Analysis of Cross-Classified Categorical Data*, 2nd ed. Springer, 2007.

[36] I. Tsamardinos, L. Brown, and C. Aliferis, “The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm”, *Machine Learning*, vol. 65, pp. 31–78, Oct. 2006. DOI: 10.1007/s10994-006-6889-7.

[37] O. Banerjee, L. E. Ghaoui, and A. d’Aspremont, “Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data”, *Journal of Machine Learning Research*, vol. 9, pp. 485–516, 2008.

[38] M. Grant and S. Boyd, "Graph Implementations for Non-Smooth Convex Programs", in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds., Springer-Verlag Limited, 2008, pp. 95–110.

[39] J. Peters, D. Janzing, and B. Schölkopf, "Identifying Cause and Effect on Discrete Data using Additive Noise Models", in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9, PMLR, 2010, pp. 597–604.

[40] M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon, "Learning Bayesian Networks with Thousands of Variables", in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://doi.org/10.1007/s10994-018-5701-9.

[41] J. Cussens, "Bayesian Network Learning with Cutting Planes", in *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, Morgan Kauffman, 2012, pp. 153–160. arXiv: 1202.3713 [cs.AI].

[42] A. Nemirovski, *Lecture Notes in Optimization II: Standard Numerical Methods for Nonlinear Continuous Optimization*. Technion - Israel Institute of Technology, 1999.

[43] K. Zhong, I. E. H. Yen, I. S. Dhillon, and P. Ravikumar, "Proximal Quasi-Newton for Computationally Intensive l_1-Regularized M-Estimators", in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, MIT Press, 2014, pp. 2375–2383.

[44] Y. Yu, J. Chen, T. Gao, and M. Yu, "DAG-GNN: DAG Structure Learning with Graph Neural Networks", in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019. arXiv: 1904.10098 [cs.LG].

[45] M. Scutari, C. Vitolo, and A. Tucker, "Learning Bayesian Networks from Big Data with Greedy Search: Computational Complexity and Efficient Implementation", *Statistics and Computing*, vol. 29, no. 5, pp. 1095–1108, 2018. DOI: 10.1007/s11222-019-09857-1.

[46] P. O. Hoyer, D. Janzing, J. Mooij, J. Peters, and B. Schölkopf, "Nonlinear Causal Discovery with Additive Noise Models", in *Proceedings of the 21st International Conference on Neural Information Processing Systems*, ser. NIPS'08, Red Hook, NY, USA: Curran Associates Inc., 2008, pp. 689–696.

[47] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed., ser. Wiley Series in Telecommnunications and Signal Processing. Wiley-Interscience, 2006.

[48] B. Aragam and Q. Zhou, "Concave Penalized Estimation of Sparse Gaussian Bayesian Networks", *Journal of Machine Learning Research*, vol. 16, pp. 2273–2328, 2015. arXiv: 1401.0852 [stat.ME].

[49] J. Ramsey, M. Glymour, R. Sanchez-Romero, and C. Glymour, *A Million Variables and More: The Fast Greedy Equivalence Search Algorithm for Learning High-Dimensional Graphical Causal Models, With an Application to Functional Magnetic Resonance Images*, 2017. [Online]. Available: https://doi.org/10.1007/s41060-016-0032-z.

[50] C. (Wongchokprasitti, H. Hochheiser, J. Espino, E. Maguire, B. Andrews, M. Davis, and C. Inskip, *Bd2kccd/py-causal v1.2.1*, version v1.2.1, 2019. DOI: 10.5281/zenodo.3592985.

[51] B. Andrews, J. Ramsey, and G. F. Cooper, "Learning High-Dimensional Directed Acyclic Graphs with Mixed Data-types", in *Proceedings of Machine Learning Research*, vol. 104, PMLR, 2019, pp. 4–21. [Online]. Available: https://proceedings.mlr.press/v104/andrews19a.html.

[52] S. Lauritzen and D. J. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems", *Journal of the royal Statistical Society. Series B (Methodological)*, vol. 50, no. 2, pp. 157–224, 1988. [Online]. Available: http://www.jstor.org/stable/2345762.

[53] D. J. Spiegelhalter and R. G. Cowell, "Learning in Probabilistic Expert Systems", in *Bayesian Statistics 4*. Claredon Press, Oxford, 1992.

[54] J. Binder, D. Koller, S. Russel, and K. Kanazawa, "Adaptive Probabilistic Networks with Hidden Variables", *Machine Learning*, vol. 29, no. 2-3, pp. 213–244, Nov. 1997. DOI: 10.1023/A:1007421730016.

[55] S. Conati, A. Gertner, K. VanLehn, and M. Druzdzel, "On-Line Student Modeling for Coached Problem Solving Using Bayesian Networks", in *Proceedings of the 6th International Conference on User Modeling*, Springer-Verlag, 1997, pp. 231–242.

[56] S. Ulianova, *Cardiovascular Disease dataset*, Jan. 2017. [Online]. Available: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset/version/1.

[57] T. C. Campbell and T. M. Campbell II, *The China Study: Revised and Expanded Edition: The Most Comprehensive Study of Nutrition Ever Conducted and the Startling Implications for Diet, Weight Loss, and Long-Term Health*. BenBella Books, 2016.

[58] C. B. Esselstyn Jr., *Prevent and Reverse Heart Disease*. Avery, 2007.

[59] M. Greger and G. Stone, *How Not to Die: Discover the Foods Scientifically Proven to Prevent and Reverse Disease*. Flatiron Books, 2015.

[60] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2019. [Online]. Available: http://archive.ics.uci.edu/ml.

[61] J. Czerniak and H. Zarzycki, "Application of Rough Sets in the Presumptive Diagnosis of Urinary System Diseases", in *Artificial Intelligence and Security in Computing Systems, ACS'2002 9th International Conference Proceedings*, Kluwer Academic Publishers, 2003, pp. 41–51. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Acute+Inflammations.

[62] babyoda, *Depression in Married Couples in Istanbul*, Nov. 2020. [Online]. Available: https://www.kaggle.com/babyoda/depression-in-married-couples/version/2.

[63] A. Beck, C. Ward, M. Mendelson, J. Mock, and J. Erbaugh, "An Inventory for Measuring Depression", *Archives of General Psychiatry*, vol. 4, no. 6, pp. 561–571, 1961. DOI: 10.1001/archpsyc.1961.01710120031004.

[64] D. Kağnıcıoğlu, "The Role of Women in Working Life in Turkey", *Sustainable Development and Planning IX*, WIT Transactions on Ecology and the Environment, pp. 349–358, Jun. 2017. DOI: 10.2495/SDP170301.

[65] R. Bailey, J. Mokonogho, and A. Kumar, "Racial and Ethnic Differences in Depression: Current Perspectives", *Neuropsychiatric Disease and Treatment*, vol. 15, pp. 603–609, 2019. DOI: 10.2147/NDT.S128584.

[66] Kaggle, *Titanic - Machine Learning from Disaster*. [Online]. Available: https://www.kaggle.com/c/titanic/data.

[67] J. C. Bigham, A. Gough-Calthorpe, A. Clarke, F. Lyon, J. Biles, and E. C. Chaston, *Report on the Loss of the Titanic*, 1912. [Online]. Available: https://web.archive.org/web/20140103020756/http://www.titanicinquiry.org/BOTInq/BOTReport/BOTRepSaved.php.

[68] P. O. Hoyer, S. Shimizu, A. J. Kerminen, and M. Palviainen, "Estimation of Causal Effects Using Linear Non-Gaussian Causal Models with Hidden Variables", *International Journal of Approximate Reasoning*, vol. 49, no. 2, pp. 362–378, 2008, Special Section on Probabilistic Rough Sets and Special Section on PGM'06. DOI: https://www.sciencedirect.com/science/article/pii/S0888613X080000212.

[69] D. Janzing, J. Mooij, K. Zhang, J. Lemeire, J. Zscheischler, P. Daniušis, B. Steudel, and B. Schölkopf, "Information-Geometric Approach to Inferring Causal Directions", *Artificial Intelligence*, May 2012. DOI: 10.1016/j.artint.2012.01.002.

[70] J. M. Mooij, J. Peters, D. Janzing, J. Zscheischler, and B. Schölkopf, "Distinguishing Cause from Effect Using Observational Data: Methods and Benchmarks", *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1103–1204, Jan. 2016, ISSN: 1532-4435.

[71] S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen, "A Linear Non-Gaussian Acyclic Model for Causal Discovery", *Journal of Machine Learning Research*, vol. 7, pp. 2003–2030, Dec. 2006. DOI: 10.5555/1248547.1248619.

Appendix A - Overview of the Method used for Comparing DAGs without Ground Truth

Here is a visual overview of the method developed for evaluating the quality of the Bayesian networks learned from purely observation data, when there is no ground truth.

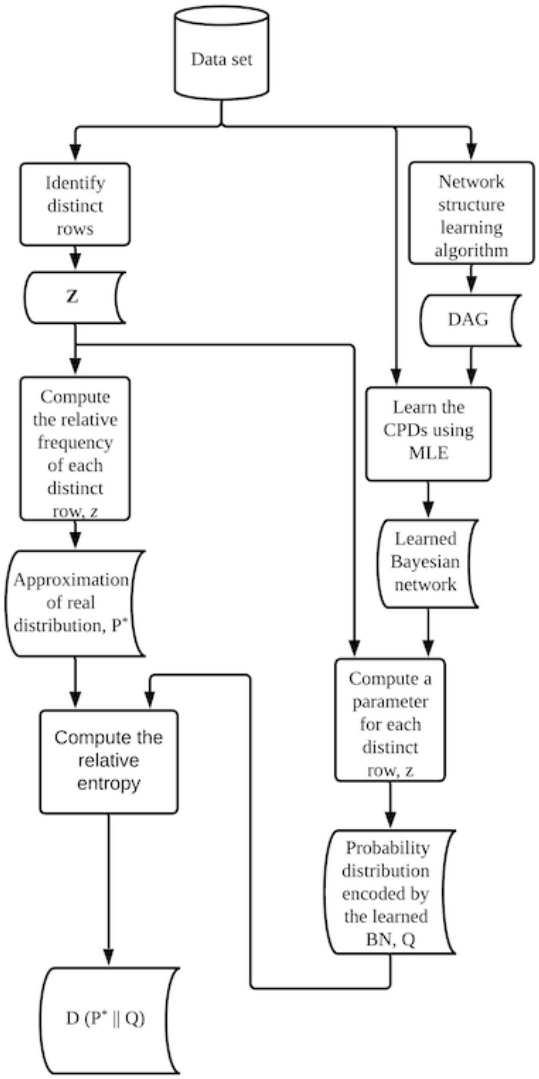


Figure 6.1: Process used to find the relative entropy between the real distribution underlying the data and the joint probability distribution encoded by the learned Bayesian network.

Appendix B - Strategies non-convex optimization

A potential problem for score-based structure learning algorithms is getting stuck at a local maximum. In order to avoid this, we can use the following methods.

Random restarts

A popular method for escaping local maxima is greedy search with random restarts, where we:

1. Apply greedy search until we hit a local maximum.
2. Randomly perturb the network structure, and repeat the process for some manageable number of iterations.

Simulated annealing

Another popular method for escaping local maxima, where we:

1. Initialize the system at some temperature T_0.
2. Pick some eligible change *e* at random, and evaluate $p = \exp(\Delta(e)/T_0)$.
    - If $p > 1$, then we make the change.
    - Otherwise, we make the change with probability *p*.
3. Repeat this selection and evaluation process α times or until we make β changes.
    - If we make no changes in α repetitions, then we stop searching.
    - Otherwise, we lower the temperature by multiplying the current temperature T_0 by a decay factor $0 < \gamma < 1$, and continue the search process.

We stop searching if we have lowered the temperature more than δ times. Thus, this algorithm is controlled by five parameters: T_0, α, β, γ and δ. To initialize this algorithm, we can start with the empty graph, and make T_0 large enough so that almost every eligible change is made, thus creating a random graph. Alternatively, we may start with a lower temperature and use one of the initialization methods described for local search.