Analysis of the pc Algorithm in the Sample Setting with an Application to the Study of Allosteric Regulation in Proteins

Katharina Dannenberg

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

From the Institute of Theoretical Computer Science
of the Universität zu Lübeck
Director: Prof. Dr. Rüdiger Reischuk

Analysis of the pc Algorithm in the Sample Setting with an Application to the Study of Allosteric Regulation in Proteins

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the Universität zu Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by

Katharina Dannenberg

Lübeck, 26. April 2021

First examiner:	Prof. Dr. Maciej Liśkiewicz
Second examiner:	PD Dr. Amir Madany Mamlouk
Date of oral examination:	05.07.2021
Approved for printing:	06.07.2021

Zusammenfassung

Auch wenn es utopisch klingen mag, ist kausale Inferenz aus Beobachtungsdaten innerhalb gewisser Grenzen möglich. In dieser Arbeit werden die Grenzen und Möglichkeiten des Lernens von kausalen Relationen untersucht, und zwar mithilfe eines klassischen Algorithmus, des PC Algorithmus.

Das Herzstück kausaler Inferenz ist eine Struktur, die die kausalen Beziehungen der betrachteten Variablen wiedergibt. Es wird oft angenommen, dass kausale Struktur ein gerichteter azyklischer Graph (DAG) ist. Auf der Basis von Beobachtungsdaten ist es jedoch typischerweise nicht möglich, *den einen* wahren Graphen zu lernen. Stattdessen wird eine Menge von ununterscheidbaren DAGs identifiziert und kompakt durch einen so-genannten CPDAG repräsentiert.

Im Orakel-Setting, in dem die zu Grunde liegenden Informationen über die bedingten Unabhängigkeiten zwischen den Variablen fehlerfrei sind, entspricht die Ausgabe des PC Algorithmus dem korrekten CPDAG. In dieser Arbeit wird jedoch gezeigt, dass der gelernte Graph im Sample Setting, das heißt, wenn die bedingten Unabhängigkeiten aus Daten abgeschätzt werden, selten auch nur zu einem DAG erweiterbar ist. Das bedeutet, dass der Ausgabegraph überhaupt keinen DAG repräsentiert, der die Daten erklären kann.

Ein weiteres Problem, das im Sample-Setting auftritt, aber in der Literatur meist vernachlässigt wird, ist, dass die gelernten Kantenrichtungen, die Kausalität repräsentieren, oft zu Widersprüchen führen. Es wurde vorgeschlagen, solche Widersprüche durch eine spezielle Art von Kanten zu repräsentieren, die hier Konfliktkanten genannt werden. Die Frage, wie diese Konfliktkanten im Weiteren zu behandeln sind, wurde jedoch bisher noch nicht untersucht.

Neben der Analyse der Häufigkeit der oben genannten Probleme und wie diese zu Stande kommen, werden hier Lösungsansätze vorgeschlagen. Hierfür wird die Wahl des Kalibrierungsparameters α untersucht, des Signifikanzniveaus der einzelnen Tests auf bedingte Unabhängigkeit. Ein neues Modell liefert unmittelbar eine Abschätzung für einen guten Wert von α basierend auf der Anzahl von Variablen, der Stichprobengröße und der gewünschten Dichte des Graphen. Außerdem wird ein Algorithmus vorgestellt, mit dem ein Graph mit Konfliktkanten in Polynomialzeit zu einem DAG erweitert werden kann.

Der praktische Nutzen dieser neuen Methoden wird in einer Anwendung aus der Biologie gezeigt, die von großer medizinischer Bedeutung ist. Der PC Algorithmus wird genutzt, um Netzwerke allosterischer Kommunikation in Proteinen zu lernen und es wird gezeigt, dass diese weitgehend konsistent mit veröffentlichten Ergebnissen sind. Außerdem ermöglichen kausale Methoden, über das hinauszugehen, was mit anderen modernen Methoden möglich ist: Basierend auf einem kausalen Modell ist es möglich, „theoretische„Experimente durchzuführen, indem die Effekte von Interventionen berechnet werden. Es wird hier gezeigt, dass dies auch auf die Vorhersage der Effekte von Mutationen anwendbar ist, was ein mächtiges Werkzeug für die Untersuchung der Funktionsweise von Proteinen liefert.

Abstract

Although it might seem utopian, causal inference from observational data is possible within a certain scope. In this thesis, the limitations and opportunities of the discovery of causal relationships are explored in terms of a classical algorithm, the PC algorithm.

At the heart of causal reasoning is a structure that represents the causal relationships between the variables under consideration. Often, this causal structure is assumed to be a directed acyclic graph (DAG). However, on the basis of observational data only, it is typically not possible to learn *the* true DAG; instead, a collection of indistinguishable DAGs is identified and succinctly represented by a so-called CPDAG.

In the oracle setting, when the given conditional independence information is perfect, the output of the PC algorithm is always the correct CPDAG. Nevertheless, it is shown in this thesis that the estimated graph is rarely even *extendable* in the sample setting, that is, when the conditional independences are assessed from data. This means that the output graph does not represent *any* DAG that explains the data.

Another problem that arises in the sample setting but is usually disregarded in the literature is that the inferred edge directions, which symbolise causality, often lead to contradictions. These contradictions have been proposed to be represented by a special type of edges, which are called conflict edges here. However, the question how these edges should subsequently be treated has not been studied so far.

In addition to an analysis of the frequency of the above mentioned problems and the reasons for their occurrence, this thesis provides approaches for their resolution. For this, the choice of the tuning parameter α, the level of significance for the individual conditional independence tests, is studied. A new model yields an immediate estimate of a good choice for α based on the number of variables, sample size, and target density of the graph. On top of that, an algorithm is introduced with which it is possible to extend a graph with conflict edges in polynomial time.

The practical benefit of these new methods is shown in a biological application of high medical relevance. The PC algorithm is used to learn networks of allosteric communication in proteins, which are shown to be largely consistent with published results. On top of that, causal methods permit to go beyond what is possible with other state-of-the-art methods: Based on a causal model, it is possible to perform "theoretical experiments" by calculating the effects of interventions. This is shown here to be applicable to the prediction of the effects of mutations in proteins, providing a powerful tool for the study of protein function.

Acknowledgements

First of all, I would like to thank my supervisor Maciej for his patience and support. I am also thankful to my colleagues, most notably Marcel for sharing his ideas with me and always being ready for discussions. In particular, the idea for the proof of Lemma 4.2 is due to him. And thanks to Max, not only for the technical support. I also thank Alvaro Mallagaray for introducing me to the fascinating world of proteins. A special thanks to Julia and Alex for their molecular biological counselling. I am also very grateful for the mental support by Esa and the rest of my family. And to Urs, for patiently improving this thesis, for life, the universe, and everything …

Contents

1

Introduction – Causal Reasoning with Observational Data

Exactly 100 years ago, Sewell Wright [312] first used a causal, not correlation-based method to determine "path coefficients" – today we would say, causal effects. Wright's quest was to determine the "relative importance of heredity and environment" in the development of coat colors in guinea pigs [313]. For this, he used what we today would call a causal structures.

This model-based approach was an innovation that made it possible to transcend from correlations to causality, a notion that was deemed unscientific in those times. Wright therefore had to face a general opposition and it would take about half a century until the notion of causality received a larger audience in the philosophy and computer science community. But still in 1990, when the predecessor of the PC algorithm, the SGS was first published, there were still strong reservations against not model-based analysis. [261]

The insight that Wright's work build upon was that it is possible to make some basic assumptions about the system studied. For example, it is obvious that a guinea pig's coat color does not have a causal influence on its parents' coat colors. In this way, he derived what is probably the first published causal structure and is reproduced in Figure 1.1). Wright found that when such a structure of assumptions on the causal relationships of the variables at hand is available, it is possible to estimate the strengths of *causal effects*, not mere correlations, from data. [206]

Today, we can go one step further. Not only can we determine causal effects from data when we are given the causal structure; we can even learn the causal structure or a class of equivalent causal structures from the data.

This makes us capable of answering causal questions like "was it the drug that cured the patient" from merely observational data, that is, without performing a randomised controlled trial (RCT), at least under certain assumptions and in certain situations.

This is particularly valuable if experiments like RCTs are unethical or infeasible. A popular example is the question of the effects of smoking on people's health [206]. Today, the question might sound obsolete, but from the first observations of the connection between smoking and lung cancer in the beginning of the 20th century [78], it took decades until smoking was generally accepted as a cause of diseases like lung cancer and – not least because of the opposition of eminent statisticians like Ronald Aylmer Fisher (e.g. [92, 93]) and Joseph Berkson (e.g., [20], [19]), whose scientific merits we will encounter later in this thesis. Although the correlation between smoking and lung cancer was mostly accepted, many doubted that smoking really was the cause in this relation-

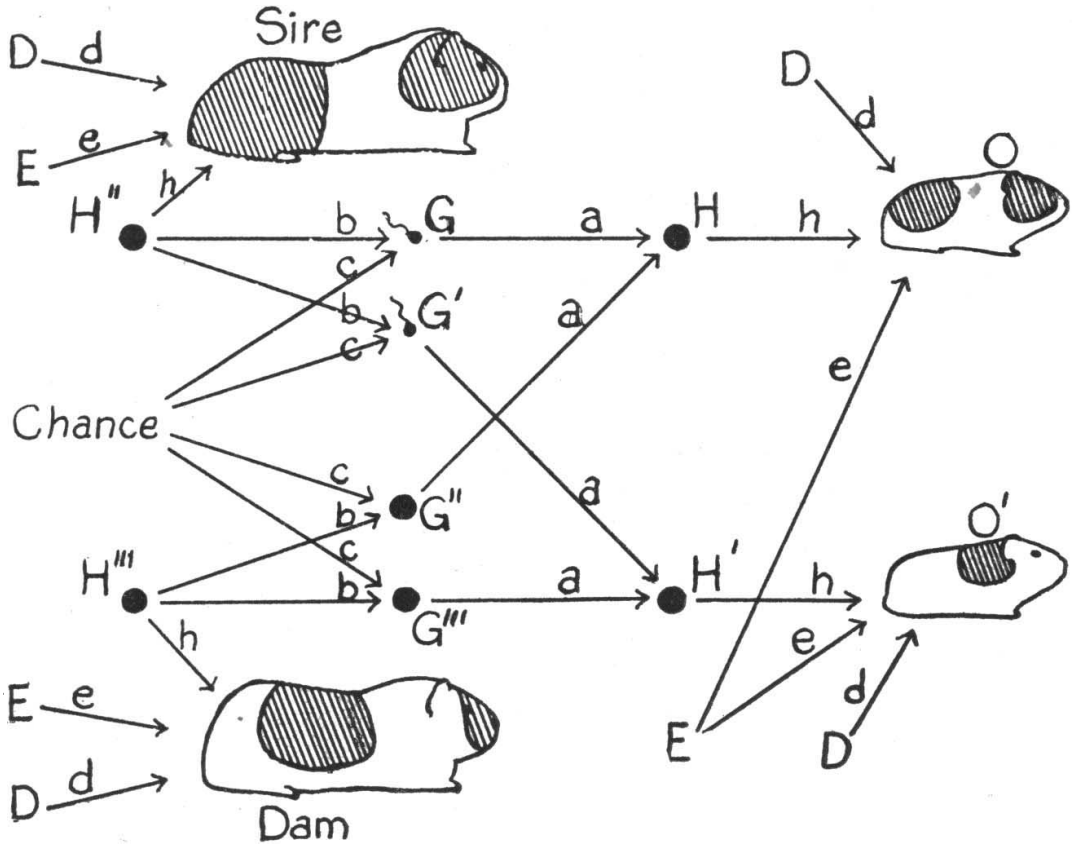


Figure 1.1: The graphical model used by Wright to express his assumptions about genetic and environmental factors of the development of fur colours in guinea pigs

ship. After all, the correlation might as well have been produced by the inverse causal relationship, or a common cause, for example genetic predispositions that lead to an increased susceptibility to cancer, as well as to an inclination to smoking [93], maybe because of nicotine craving. Had randomised controlled experiments been conducted on the matter, the statistical scientific society could surely have been convinced and, as Pearl and Mackenzie [206] formulates it, "millions of lives [...] been saved and lifespans lengthened".

Unfortunately such an experiment cannot be conducted for ethical reasons, as this would require to force a large enough, randomly composed group of people to smoke (or not smoke) for a considerable time span – with all the effects on health that this would have. There are also variables that cannot be "set" by an intervention or policy. For example, we might want to investigate the connection between sleep disorder and depression, which have been found to be correlated. [227] However, we can neither randomly assign research participants to suffer from depression nor sleep problems, so we have to do this study using observational data from patients that have these conditions for reasons we do not know [244].

That is where causal discovery from observational data comes into play. Given a large enough amount of data on all relevant variables and with some assumptions on the nature of the graphical model, causal effects can today be identified or at least, bounds on their

sizes can be estimated [172]. However, this branch of causality must still be considered in their infancy, as many problems can arise in practice. This thesis can hopefully contribute to the development of methods that yield good results in practice, by analysing the potential and limitations of the PC algorithm [263], a canonical method that also forms the basis of many other causal structure learning algorithms.

1.1 Structure of this Thesis

Causal analysis based on observational data is usually subdivided in two steps: causal structure learning and causal effect estimation.

This thesis provides an extensive introduction to causal modelling but is concerned mainly with the first step, causal structure learning, for which the canonical PC algorithm is applied. I evaluate the accuracy and many other properties of the learned graph and cover the choice of the tuning parameter α. In a case study, I show the applicability of the algorithm to the study of protein function.

Moreover, I identify some limitations of the algorithm that are usually disregarded in the literature. Generally, the problem is that sampling errors and violations of the assumptions can give rise to inconsistencies. These do not only impair the accuracy to the estimated graph, they also cause it to be invalid as a causal representation in many cases.

The thesis begins with an introduction to the foundation of causality on the basis of observational data in Chapter 2. In particular, Section 2.2.2 explains the rather surprising emergence of directed edges in models that are learned on the basis of symmetric conditional independence tests. Another key concept that is introduced in this chapter is Markov equivalence. On the one hand, the Markov property (defined in Section 2.3.1) is the means to describe the compatibility of a graph and distribution that is observed in terms of data. On the other hand, Markov equivalence of graphs (see Section 2.4.1) characterises the indistinguishability of causal structures given data, which defines the bounds of what we can hope to achieve by causal structure learning. Since a Markov equivalence class can be described by a so called CPDAG (for the definition see Section 2.4.2), this is the intended output of many causal structure learning algorithms. Although we are often only concerned with the causal structure, that is the graph learned by the PC algorithm, the causal model also describes the functional relations of the dependence of variables. Such a model is called a structural causal model and defined in Section 2.5. The generative character of SCMs permits to answer causal questions on their basis, the second above-mentioned step of causal analysis. This includes interventional and counterfactual reasoning, as briefly introduced in Section 2.5.1.

Chapter 3 details the problem of learning the model from data. Besides a characterisation of the problem, the underlying assumptions and its complexity, the main part of this chapter is an overview over existing causal structure learning methods (Section 3.5). The two basic approaches to this are to learn the graph based on independences identified in the data (like the PC algorithm), or based on a score that measures the fit of structure and data. Beyond that, other methods exist that combine these two approaches or apply a completely different method. Some algorithms are devised for a particular setting, for

example in the presence of missing data, or when a causal ordering of the variables or interventional data are available.

Having learned the causal structure, one can derive the direct causes (or parents) of the respective variables (up to indistinguishability). On the basis of a presupposed functional form of the dependence of a variables on their parents, it then remains to determine the individual parameters of these relationships as described in Section 3.3, which usually is straightforward compared to the identification of the structure. In the end of the Chapter 3, some common measures for the evaluation of the learned causal structures are given. Section 3.6 gives an introduction to the above mentioned scores that measure the fit of a causal structure and given data. In Section 3.6.2 measures are presented which can be used to compare different causal structures.

Chapter 4 gives a detailed introduction to the PC algorithm and its properties. Most of the results in this chapter are new. The first one is a theoretical characterisation of the order-dependence of the oracle version of the PC algorithm, that is the setting where all conditional independence information that the algorithm obtains is correct. Although the PC algorithm is consistent, that is, always returns the correct CPDAG in the oracle setting, the intermediate results in fact depend on the ordering of the input variables, as shown in Section 4.2.

The rest of the section consist of empirical evaluations of the PC algorithm in the sample setting, where the conditional independences are estimated from data by a conditional independence test. The starting point of this analysis is a result on the accuracy of the PC algorithm presented by [148]. I am able to show that this result is significantly idealised by the ordering of the variables in the input. Furthermore I investigate the the validity of the estimated graph. It turns out, that in many settings only a vanishing fraction of the estimated graphs are CPDAGs, and many of the resulting graphs are not even acyclic or contain conflict edges. Section 4.3.4 confirms that these problems become more frequent with increasing density of the estimated graph.

Chapter 5 is mainly concerned with the validity of the PC-estimated graphs including "symptoms" on invalidity like cycles and conflict edges. A new characterisation of the result of the PC algorithm as a general partially directed graph (GPDG) is proposed in Section 5.1, and the internal structure of this class of graphs is characterised. After analysing the reasons for such symptoms in Section 5.2, I consider two main approaches to deal with them. On the one hand, parameter tuning can not only improve the accuracy of the result, but – reducing sampling errors – also improves the validity. In Section 5.3.2, I propose a new model by which the value for α can be determined dependent on the number of variables, the sample size and the estimated density of the true graph. While for this, the PC algorithm is considered as a "black box", the individual p-values of the conditional independence tests are considered in Section 5.3.3. In a supervised setting, the properties of an optimal choice of α are determined. The resulting high values of α (often far above $\alpha = 0.5$) emphasise the importance of regarding also the type-II-error of erroneously removing edges. On the other hand, in Section 5.4 an algorithm is presented to obtain a DAG that is as similar to a GPDG as possible. This algorithm is based on a generalisation of an algorithm by Dor and Tarsi [79] and allows to extend graphs that contain conflict edges. This algorithm is able to retain the polynomial running time of the original algorithm.

Finally, in Chapter 6 the PC algorithm and the new extensions are applied to learn networks of allosteric communication in proteins. For this, NMR data are used in Section 6.1 and evolutionary data in Section 6.2. In the latter case the computation of the causal structure is complemented with an estimation of effects in the resulting graph using a method called IDA [172]. In both cases it is possible to partly reproduce and complement published results. The main results of this thesis are summarised in in Chapter 7.

2

Preliminaries and Causality Backgrounds

What is causality? Although it is easy to explain the difference between a *directed* causal relationship and an *undirected* correlation, it is not obvious where the methods of causality theory differ from the ones used in "ordinary" statistics: After all, causal methods rely on observational data, use statistical tests to judge the significance of the coupling of variables and partial correlations to assess its strength. Nevertheless, they are capable of finding de novo causal relationships, to predict the effects of "interventions" – a utopian notion for many scientists.

The main tool in the field of causality are graphical models, which can be seen as a union of statistics and graph theory. They are a manifestation of the belief that statistical data must be interpreted on the basis of a model because "data cannot speak for themselves" [130].

The aim of this section is to explain how causal discovery can be possible.

First of all, we have to deduce something from the given data that can give us information about causality. What we can see in the data is that variables behave similarly. Intuitively, one can think of one variable always increasing, or decreasing, when another variable increases.

However, even if we assume that there is no correlation without *some* causality – the common cause principle first postulated by Reichenbach [221] – we do not know which variable is the cause or whether there is another variable that is the common cause of both.

That's why we have to regard more than two variables and measure not only dependences between pairs of them (so called marginal dependences), but also conditional dependences – are the variables still correlated if we keep some other variables fixed? For example, if we hold the common cause constant, this would render the two effected variables independent (assuming there are no other common causes). Using an example from [203, p. 116], height and reading ability of children are certainly correlated: The taller the children, the better they read. However, this is only due to the common cause of height and reading education – age. If we only regard children with a fixed age (say, eight years), we will probably not find the taller children to be the better readers. Luckily, we can also estimate *conditional* independences from data – although this becomes increasingly hard when the set of variables that are held constant becomes larger.

In the next two sections, conditional dependence and independence of variables will be introduced: In Section 2.1, we will first take a look at how to assess them in data –

the statistical component. In Section 2.2 we consider independences in graphs. In Section 2.3 we will assemble the two components and be able to fully describe the aim that is pursued from the next chapter throughout the rest of this thesis – learning causal structures from data. We will see, however, that graphs can be indistinguishable in the light of data and investigate this issue and its implications in Section 2.4. Finally higher level type of model will be presented in Section 2.5, which combines both data generation and corresponding graphs and allows causal inference in the form of theoretical experiments – just what we need to answer questions like "Does smoking cause lung cancer?" or "Why are depression and sleep disorders" correlated?

2.1 Independences in Statistics

In this section, I will define some probabilistic and statistical notions that are of importance in this thesis. For a general introduction see, for example, the textbook by Casella and Berger [36].

In theoretical considerations, we usually regard discrete variables that have a joint multinomial distribution. In practice, data are often continuous. Sometimes this is handled by discretising the data, that is binning them into intervals and regarding these intervals as a discrete domain, which, however, always entails some discretisation error. Luckily, many of the results for multinomial data carry over, at least in some form, if continuous data is assumed to have a Gaussian distribution. This assumption also has some statistical advantages as we will see later in this thesis. Since continuous data are used in the empirical parts of this thesis, we focus on Gaussian distributions when considering conditional independence tests in Section 2.1.4.

2.1.1 Probabilistic Independences

Throughout this thesis, variables will be denoted by capital letters, often X, Y, Z or $X_1, X_2, \ldots$ and their assumed values with the corresponding small letters. Sets of variables are written in bold letters, e. g. $\mathbf{X}$. In a slight abuse of notation, the braces around sets will sometimes be omitted writing, for example, $P(X, Y)$.Also, a continuous probability distribution P is identified with its probability density function $P(X = x)$ for variables X and their assumed values x (for discrete variables $P(X = x)$ refers to the probability mass function instead). We use the notation $P(x)$ as an abbreviation for $P(X = x)$. Recall that conditional probabilities are defined as follows:

$$P(X \mid Z) = \frac{P(X, Z)}{P(Z)}.$$

The expected value of a variable measures its mean value, weighted by the corresponding probabilities of the individual values. For a discrete variable X with a finite domain $\{x_1, x_2, \dots x_n\}$ it is computed as follows:

$$\mathrm{E}(X) = \sum_{x_i} P(x_i) \cdot x_i.$$

A key concept in this thesis is independence of variables. Two random variables X and Y are independent of one another (written $X \perp\!\!\!\perp Y$) if their joint probability equals the product of the individual probabilities, that is, $P(X, Y) = P(X) \cdot P(Y)$. This notion of independence can be extended to conditional probabilities: Two variables X and Y are conditionally independent given another variable Z, when the product of the individual conditional probabilities is equal to the joint conditional probability [108]. Formally we define

$$(X \perp\!\!\!\perp Y \mid Z) \quad \text{if and only if} \quad P(X, Y \mid Z) = P(X \mid Z) \cdot P(Y \mid Z). \tag{2.1}$$

Except for the edge case when $P(Y \mid Z) = 0$ and thus also $P(X, Y \mid Z) = 0$, this can be transformed into the following, by using the definition of conditional probabilities:

$$(X \perp\!\!\!\perp Y \mid Z) \quad \text{if and only if} \quad P(X \mid \mathbf{Y}, \mathbf{Z}) = P(X \mid Z), \text{assuming } P(Y \mid Z) \neq 0. \tag{2.2}$$

If X and Y are not independent (given Z), they are dependent, which is written as $X \not\perp\!\!\!\perp Y$. In these definitions, the variables X, Y and Z can also be replaced by sets of variables. The set $\mathbf{Z}$ is then referred to as the conditioning set . The edge case $P(X \mid \emptyset)$ is then defined to be equal to the probability of X, that is $P(X)$. Referring to the independences that hold in a graph or a distribution, the dependences are usually also implied. Likewise, the question whether a pair of nodes is dependent or independent given a set of other nodes $\mathbf{Z}$ is referred to a conditional independence query or simply a conditional independence. The order of a conditional independence is the cardinality of the set $\mathbf{Z}$ above. For unconditional independences, which are also called marginal independences, it is zero. Likewise conditional independences in which the condition is the empty set, are interpreted as marginal independences, that is $(X \perp\!\!\!\perp Y \mid \emptyset) \leftrightarrow (X \perp\!\!\!\perp Y)$. We will used this notion of the *order* also for conditional independence tests: The order of a conditional independence test is the order of the conditional independence that is tested.

Conditional independence relations have some properties which help to understand the notion [69, 68, 207]:

Symmetry:	If $X \perp\!\!\!\perp Y \mid Z$, then $Y \perp\!\!\!\perp X \mid Z$.	
Decomposition:	If $W \perp\!\!\!\perp X, Y \mid Z$, then $W \perp\!\!\!\perp X \mid Z$.	
Weak Union:	If $W \perp\!\!\!\perp X, Y \mid Z$, then $W \perp\!\!\!\perp X \mid Y, Z$.	(2.3)
Contraction:	If $W \perp\!\!\!\perp Y \mid Z$ and $W \perp\!\!\!\perp X \mid Y, Z$, then $W \perp\!\!\!\perp X, Y \mid Z$.	
And, in strictly positive probability distributions:		
Intersection:	If $W \perp\!\!\!\perp X \mid Y, Z$ and $W \perp\!\!\!\perp Y \mid X, Z$, then $W \perp\!\!\!\perp X, Y \mid Z$.	

Pearl [201] remarks that these properties also hold for the relation "all paths from X to Y are intercepted by Z" in a graph, which provides a good intuition, but unfortunately applies in undirected models only (see the corresponding paragraph on page 27). For our

purposes, the connection between conditional independence and graphs will be established in terms of d-separation in Section 2.2.3.

A joint or multivariate distribution is a distribution that assigns a probability to all combinations of values for several variables. Given a joint distribution $P(X_1, X_2 \dots, X_n)$, there are two ways to eliminate of some variables X_i: If the value x_i of this variable is known, or only one value x_i is relevant, then one can condition on it and regard the probability $P(X_1, X_2, \dots, X_{i-1}, X_{i+1}, \dots, X_n \mid X_i = x_i)$. If, on the other hand, a variable X_i is of no interest, one can sum over all values for this variable according to the law of total probability, thus removing it from the distribution:

$$P(X_1, X_2, \dots, X_{i-1}, X_{i+1}, \dots, X_n) = \sum_{x_i} P(X_1, X_2, \dots, X_{i-1}, X_i = x_i, X_{i+1}, \dots, X_n).$$

Conceptually, this process of marginalisation amounts to averaging over the variable, which is then called marginalising it out.

2.1.2 Assessing Independences in Data

In the previous section, conditional independence was defined from the point of view of probability theory. Can we use the formula given there in practice, after estimating the relevant probabilities from the data?

There are several problems with this approach. Let us first consider discrete variables. Assume that we want to evaluate if $P(X, Y \mid Z) = P(X \mid Z) \cdot P(Y \mid Z)$. In this case, for every value z that Z can assume, it must hold that $P(X, Y \mid Z = z) = P(X \mid Z = z) \cdot P(Y \mid Z = z)$. But what if there is a value z that is not represented in the data? We cannot simply remove z from the domain of Z, since it might well play a role for other configurations of the remaining variables: Assume that there is some value z_1, which the variable Z_1 never assumes when $Z_2 = z_2$ but well when $Z_2 = z_2'$. Then, $P(X \mid Z_1 = z_1, Z_2 = z_2)$ is not defined, but removing z_1 from the domain of Z_1 would also prevent us from regarding $P(X \mid Z_1 = z_1, Z_2 = z_2')$.

On top of that, the case of undefined probabilities is not rare, as one can easily see with a bit of combinatorics: Already for n variables with only two states, one needs at least 2^n samples to incorporate all combinations of variable values into the sample. As this would not be very conclusive statistically, true samples would have to be much larger.

Another problem is pointed out in [214, p. 218]: Not only must all combinations of values *occur* in the data, they must also all be considered, which becomes very costly in high-order queries. For example, a variable X is independent of a variable Y given the variables $Z_1, Z_2, \dots, Z_l$ if the variables are independent *for all* combinations of the values that the variables Z_i can assume. Even if they all are binary, this results in 2^l conditional independences which must be evaluated.

For continuous data, which can be seen as categorical data with infinitely many categories, the situation theoretically becomes even worse. However, assumptions about the variables' distribution, most of the times Gaussianity, can be made if the variable domains are ordered. This makes the conditional independence testing considerably more efficient, but can also lead to problems when the assumptions are not justified. This issue is further discussed in Section 5.2.1.

Another problem related to the above considerations is that for conditional independence tests with discrete data, only a subset of the samples can be used for each test – the ones for which the condition applies. This leads to decreasing power of statistical test, especially for discrete, but also with continuous data [266].

In practice, however, the limited sample size is not only problematic because it can entail undefined probabilities, but also causes problems regarding the accuracy of estimates. When estimated probabilities are used, the joint probability will never be exactly equal to the product of the individual probabilities; we thus need a statistical test that, considering the sample size and other features of the data, decides for us whether the observed semi-equality can justify the assumption of independence or not.

Statistical independence tests often do not evaluate this directly according to the definition, but by means of measures of the variable association, like the correlation or covariance. These will be explained in the next section.

2.1.3 Variable Association

As causal discovery eventually is based on *symmetric measures* of the connection between variables, we will first define these. The two basic measures of association between variables are covariance and correlation.

The covariance between two variables X and Y quantifies to what extent they deviate similarly from their respective expected values. It is defined as the expected value of the product of these deviations:

$$\mathrm{Cov}(X, Y) = \mathrm{E}((X - \mathrm{E}(X)) \cdot (Y - \mathrm{E}(Y))).$$

The expected value of a variable $\mathrm{E}(X)$ is usually not known, but can be estimated from data in terms of the sample mean $\overline{x}$ and $\overline{y}$. Given a sample $x_1, x_2, \ldots x_s$, the sample mean is simply $\overline{x} = \frac{1}{s}\sum_{i=1}^{s} x_i$. This yields the sample covariance $\overline{\mathrm{Cov}}(X, Y)$:

$$\overline{\mathrm{Cov}}(X, Y) = \frac{1}{s}\sum_{i=1}^{s}(x_i - \overline{x})(y_i - \overline{y}).$$

Since the deviation of a variable from its mean can be arbitrarily large, this measure is not bounded from above. To achieve comparability of the association across variables and data sets, the covariance is normalised to obtain the correlation between the variables:

$$\mathrm{Corr}(X, Y) = \rho(X, Y) = \frac{\mathrm{Cov}(X, Y))}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}} = \frac{\mathrm{E}((X - \mathrm{E}(X)) \cdot (Y - \mathrm{E}(Y)))}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}}.$$

Accordingly, when estimated from a sample:

$$\overline{\mathrm{Corr}}(X, Y) = \frac{\frac{1}{n}\sum_{i=1}^{s}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\overline{\mathrm{Var}}(X)} \cdot \sqrt{\overline{\mathrm{Var}}(X)}} = \frac{\sum_{i=1}^{s}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{s}(x_i - \overline{x})^2 \cdot \sum_{i=1}^{s}(y_i - \overline{y})^2}}.$$

This type of correlation ist called Pearson or Bravais-Pearson correlation [28, 312]. It measures the linear correlation between the variables. If the correlation is assumed or known to be non-linear, it might be more compute the correlation of the ranks – not the variable values – with the above formula for Pearson correlation. The result is referred to as the Spearman correlation and more robust to outliers at the expense of discarding the actual values of the variables. The Spearman correlation is maximal (-1 or $+1$) when the connection between the variables is monotone (but not necessarily linear). Another example for a correlation coefficient that is based on a ranking of the variable values is Kendell's τ.

The core of causal reasoning is to assess direct effects between variables with the effects of all other variables eliminated. Although not causal itself, the partial correlation is an important tool for this purpose. It quantifies the association of two variables X and Y, with the influence of all variables in a set Z disregarded. It can be computed in one of three ways:

Recursively

On the basis of the Pearson correlation $\rho(X, Y) = \rho_{X,Y|\emptyset}$, the partial correlation $\rho_{X,Y|Z}$ can be computed by successively adding elements of Z using the following formula:

$$\rho_{X,Y|Z\cup Z'} = \frac{\rho_{X,Y|Z} - \rho_{X,Z'|Z} \cdot \rho_{Z',Y|Z}}{\sqrt{1 - (\rho_{X,Z'|Z})^2} \cdot \sqrt{1 - (\rho_{Z',Y|Z})^2}}.$$

In the special case of a singleton set $Z = Z$, this entails [186]

$$\rho_{X,Y|Z} = \frac{\rho(X, Y) - \rho(X, Z) \cdot \rho(Z, Y)}{\sqrt{1 - (\rho(X, Z))^2} \cdot \sqrt{1 - (\rho(Y, Z))^2}}.$$

Based on this, a partial correlation of order k can be computed in time $O(k^3)$ using dynamic programming [300].

From the Precision Matrix

It is also possible to compute the partial correlation $\rho_{X_i,X_j|\mathcal{V}\setminus\{X_i,X_j\}}$ of two variables given all the others directly. For this purpose, one needs to invert the correlation matrix $M = (\rho(X_i, X_j,))$ – this is possible since the matrix is positive definite – to obtain the precision matrix $P = M^{-1} = (p_{ij})$. Then,

$$\rho_{X_i,X_j|\mathcal{V}\setminus\{X_i,X_j\}} = -\frac{p_{ij}}{\sqrt{p_{ii} \cdot p_{jj}}}.$$

By Linear Regression

Another option for computing partial correlations is to regress both variables X and Y on the variables in Z [311]. This gives us two vectors of residuals of length s, which contain the deviations of the actual values of X and Y from the ones predicted by the regression. Let w^X and w^Y be the vectors that contain the regression weights, including the constant offsets w_s^X and w_s^Y. To obtain a vector of equal dimension, we augment the vector of values for the variables Z, z_i with $i \in \{1, 2, \dots, s-1\}$, with the entry $z_s = 1$, corresponding to w_s.

Then the residuals are $\epsilon_i^X = x_i - w^X \cdot z_i$ and $\epsilon_i^Y = y_i - w^Y \cdot z_i$, for $i \in \{1, 2, \dots, s\}$, where $w^X \cdot z_i$ and $w^Y \cdot z_i$ denote scalar products, respectively. The partial correlation is then equal to the Pearson correlation between the residuals $\mathrm{E}^X = (\epsilon_i^X)$ and $\mathrm{E}^Y = (\epsilon_i^Y)$:

$$\rho_{X,Y|Z} = \rho(\mathrm{E}^X, \mathrm{E}^Y). \tag{2.4}$$

It is important to note that, like Pearson correlation, which it is based on, the partial correlation $\rho_{X,Y|Z}$ measures the *linear* dependence of X and Y conditional on Z only if the variables have a Gaussian distribution. It might yield inconsistent results for non-Gaussian variables or non-linear associations.

2.1.4 Conditional Independence Testing

After having measured variable association in the data, we can define tests that turn these association measures into boolean statements – are the variables (likely) dependent or not? The basic principle of tests for conditional independence are outlined here, followed by the description of a test that is used in the empirical evaluations in this thesis.

In general, a conditional independence test (like any other statistical test) is implemented by computing a p-value and comparing it to a threshold α, the level of significance of the test. Then the null null hypothesis H_0 that the variables are independent is rejected when p-value $< \alpha$ and kept otherwise.[1]

The p-value is computed in terms of a test statistic T, which projects the data D to a scalar value. This statistic is then scaled to ensure that the probability of a type I error, i.e. that the test rejects a true independence statement, actually equals the user-specified level of significance α. The other type of error, keeping a false independence statement, is referred to as a type II error and its probability, that depends on the choice of α and the properties of the test statistic is denoted by β. It is also referred to as the power of the test.

Fisher's Z-test

A test statistic that is often used for Gaussian data is based on Fisher's $\mathcal{Z}$-transform, which is defined as

$$\mathcal{Z}(X, Y \mid Z) = \frac{1}{2} \log\left(\frac{1 + \rho_{X,Y|Z}}{1 - \rho_{X,Y|Z}}\right). \tag{2.5}$$

and tests vanishing partial correlation (see Equation (2.4)). In other words, it assesses if the partial correlation $\rho_{X,Y|Z}$ is zero [14, 148].

From this, the test statistic $T = \sqrt{s - |Z| - 3} \cdot |\mathcal{Z}(X, Y \mid Z)|$ is derived, where the s is the sample size. Then, the p-value can be computed as

$$-2\Phi(T) + 2,$$

[1] Usually, in statistical testing the null hypothesis is rejected when the p-value is smaller or equal to α (i.d. p-value $\leq \alpha$); however, following Kalisch and Bühlmann [148] and the implementation of the PC algorithm in the R package PC algorithm, in this thesis independence is assumed when the p-value is strictly *smaller* than α (i.d. p-value $< \alpha$).

where Φ denotes the cumulative distribution function of $\mathcal{N}(0,1)$. This entails that the null hypothesis is rejected if and only if

$$-2\Phi(\sqrt{s-|\mathbf{Z}|-3}\cdot|\mathcal{Z}(X,Y\mid\mathbf{Z})|)+2<\alpha. \tag{2.6}$$

This test is applied to evaluate conditional independence throughout this thesis unless stated otherwise. It is the default test used in the R package `pcalg`, which contains the implementation of the PC algorithm that is used for the empirical studies in Chapter 4 and 5. In `pcalg`, Fisher's $\mathcal{Z}$-test is implemented in the functions `gaussCItest` and `condIndFisherZ`.

A downside of tests for zero partial correlation is that it assumes the variables to have a Gaussian distribution. Vanishing partial correlation is neither sufficient nor necessary for conditional independence for non-Gaussian variables, as shown in [214, Example 7.9]. As the Gaussianity assumption is often farfetched or not justified at all, non-parametric tests, that do not assume the data to have a certain distribution, are often a better choice, at least theoretically. Unfortunately, such tests are difficult to perform when the sample size is finite, as has been illustrated with the aid of the examples in the beginning of previous section (see page 7).

A general approach for dealing with data of an unknown distribution is to use ranks of observations instead of the observations themselves. Particularly for the PC algorithm, there is are extensions for Gaussian copulas [122] (rank-PC or RPC), and for mixed discrete and continuous data [62] (copula-PC) (see Section 4.1.3).

Kernel-based Conditional Independence Tests

Recently, another kind of non-parametric conditional independence tests has gained a lot of attention: kernel-based methods, in particular methods based on an injective mapping onto reproducing kernel Hilbert spaces [234]. Many such methods are based on the Hilbert-Schmidt independence criterion (HSIC) [116], which can be efficiently evaluated and seem to yield superior results. A generalisation and an R-implementation in the package `dHISC` have been provided by [215]. There are also extensions of HSIC for continuous variables [104, 318]. For an introduction to such methods see [214] or [5].

2.2 Independences in Graphs

2.2.1 Graph Theory Basics

In this section, some basic or particularly important concepts of graph theory are presented. For a more profound introduction see, for example, Diestel [77].

A graph is a tuple $(\mathcal{V},\mathcal{E})$ consisting of a set of nodes (or vertices) $\mathcal{V}$ and a set of edges $\mathcal{E}$. In this thesis, we always denote the number of nodes in a graph by n. The edges form connections between the nodes which can be undirected or directed, in which case they sometimes are called *arcs* or *arrows*.

Because we often regard partially directed graphs (PDGs), that contain both directed and undirected edges, we will model the latter not as sets, but as the union of two opposite directed edges. This is done as follows: An edge is undirected (depicted as $X - Y$) if $(X, Y) \in \mathcal{E}$ and $(Y, X) \in \mathcal{E}$, and directed (depicted as $X \to Y$) if $(X, Y) \in \mathcal{E}$ and $(Y, X) \notin \mathcal{E}$. A directed or undirected graph comprises only edges of the respective type. The skeleton of a partially directed graph is the undirected graph obtained by replacing all directed edges with undirected edges.

Especially in computations, graphs are often represented as by their adjacency matrix. This is an $n \times n$-matrix $A = (a_{ij})_{i=1,2,\dots,n;j=1,2,\dots,n}$ in which an entry is set, $a_{ij} = 1$, if there is an edge $X_i \to X_j$ in the graph. According to our definition, an undirected edge $X_i - X_j$ is then represented by $a_{ij} = a_{ji} = 1$. We will later also regard a special type of edges called conflict edges, which are modelled in the adjacency matrix by $a_{ij} = a_{ji} = 2$.

In weighted graphs, every edge $X_i \to X_j$ is assigned a weight $w_{ij} \in \mathbb{R}$. We assume these weights to be different from zero, so that we can set $w_{ij} = 0$ for non-existing edges $X_i \to X_j$, and obtain a weighted adjacency matrix $W = (w_i j)_{i=1,2,\dots,n;j=1,2,\dots,n}$. Then it is required for consistency, that for undirected edges $X_i - X_j$, $a_{ij} = a_{ji}$. Conflict edges must then be recorded separately or the unweighted adjacency matrix A is kept in addition.

An edge is incident to the nodes which it connects (and vice versa), and two nodes connected by a common edge are adjacent. Such nodes are also called neighbours. Consequently, the neighbourhood $\mathcal{N}(X)$ of a node X is the set of all of its neighbours. If edges are directed, neighbours can be divided into parents and children, with $P \to C$ pointing from a parent P to a child C. The set of parents of a node X_i in a graph $\mathcal{G}$ is referred to $Pa_i^{\mathcal{G}}$, or Pa_i when the graph is unambiguous. Spouses are defined to be parents of common children.

The density of a graph is a general notion that describes how many edges the graph has in comparison with its nodes. A sparse graph has few edges and a dense graph has many. To measure the density of a graph, we use its degree. The degree $\mathrm{d}_{\mathcal{G}}(X)$ or simply $\mathrm{d}(X)$ of a node X in a graph $\mathcal{G}$ is number of neighbours that this node has. In directed graphs this can be further subtilised. The in-degree of a nodes is the number of parents that it has, the out-degree the number of children.

The degree of a graph, which measures its density, is the mean degree of its nodes, which can be computed as $\mathrm{d}(\mathcal{G}) = {}^{2|\mathcal{E}|}/_{|\mathcal{V}|}$. (Note that every edge contributes to the degree of its two incident nodes). The degree of a graph is also referred to by the variable $\hat{d}_G$ or simply $\hat{d}$, if the graph is unambiguous.

In the graph sampling methods that we use here, edges are sampled independently between all pairs of nodes with a probability p_e. The expected degree d of the resulting graph is then $d = p_e \cdot (|\mathcal{V}| - 1)$. The maximal degree d_{max} of a graph is the maximal degree of one of its nodes.

A walk is a sequence of nodes such that consecutive nodes are adjacent in the graph (regardless of the edge directions). A walk $(X_1, \dots, X_n)$ is said to be a walk from X_1 to X_n or between X_1 and X_n, or to start in X_1 and end in X_n. In a directed walk, consecutive nodes X_i, X_{i+1} are connected by a directed edge $X_i \to X_{i+1}$. A (directed) path is a (directed) walk in which every node appears at most once. A (directed) path that starts and ends in the same node is called a (directed) cycle. A semi-directed cycle arises from a directed cycle by replacing some edges by undirected edges.

A directed graph is said to be acyclic if it does not contain a directed cycle, an undirected graph is acyclic if it does not contain a cycle. For partially directed graphs acyclicity is linked to extendability, which is defined later (see Definition 10).

In this thesis, acyclic graphs play in important role, especially directed acyclic graphs (DAGs). These are graphs with only directed edges which are required not to contain a *directed* cycle. In a DAG, one or more nodes do not have any parents. These are referred to as sources. There are also always one or more sinks, nodes without any children. A topological ordering $X_1, X_2, \dots, X_n$ of the nodes of a directed graph $\mathcal{G}$ is an ordering so that for every edge $X_i \to X_j$ in $\mathcal{G}$ it holds that $i < j$. A directed graph has a topological ordering if and only if it is acyclic, that is if and only if it is a DAG. Other relevant graph models are given in the Section 2.4.

An induced subgraph of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with respect to a set of nodes $\mathcal{V}' \subseteq \mathcal{V}$ is the graph that results when all other nodes $\mathcal{V} \setminus \mathcal{V}'$ are removed along with their incident edges. What remains then is the graph between the nodes $\mathcal{V}'$, including *all* the edges between those nodes. An important graph pattern that we will be concerned with in this thesis, $X \to Y \leftarrow Z$, is called a v-structure. A graph is said to contain a v-structure if it has an *induced* subgraph that is a v-structure. Note that this means that the edge $X - Z$ does not exist, since the induced subgraph of $\{X, Y, Z\}$ contains all the edges between these nodes.

Historically, v-structures have also been termed immoralities, interpreting them as a pair of non-adjacent, that is, unmarried, parents with a common child, We will avoid this term in this thesis, except in the case of moral graph extensions (see Definition 10). The middle node of a v-structure (in this case Y) is called a collider. A triple of nodes that form a chain $X - Y - Z$ – especially when directed as $X \to Y \leftarrow Z$ – is said to be shielded when an edge between X and Z (either directed or undirected) exists, such that no v-structure is formed.

The pattern of a partially directed or undirected graph is a graph with the same skeleton that is undirected apart from all directed edges that belong to one or several v-structures.

As has been mentioned before, the graphical models that we are concerned with in this thesis are DAGs. They have a straightforward causal interpretation. If there is a directed edge $X \to Y$, this signifies that X is one of the direct causes for Y. As we will see later, such DAGs are in general only partially identifiable from observational data, which is why we will regard classes of observationally indistinguishable graphs. These can be represented by so-called CPDAGs (see the definition in Section 2.4.2). We will also regard unrestricted partially directed graphs (PDGs), which will be relevant to model the learning algorithms' practical results.

CPDAGs, as well as DAGs, are chain graphs [103, 11]: Partially directed graphs which do not contain semi-directed cycles. A chain component of a chain graph is a connected component of the graph that is obtained when all the directed edges are removed. In this thesis we will not restrict the use of this term to chain graphs. An undirected graph, and in particular a chain component, is chordal if every cycle of length at least four has a chord, that is two nonconsecutive adjacent nodes. Another type of graph that is often regarded in the literature are partially directed acyclic graphs (PDAGs), that do not contain directed cycles but may contain undirected and semi-directed cycles. We will not

regard them here, but characterise PDAGs as PDGs instead (dropping the acyclicity claim altogether).

2.2.2 The Emergence of Direction

In Section 2.1.3, we have seen how the association between variables can be measured. It is not surprising that this information can be used to define tests for independence of variables and that these can be generalised to assess conditional independences (see Section 2.1.4). It is maybe also intuitive to build undirected causal models in which edges represent not marginal correlations, but correlations conditional on all other variables or all subsets of them, as in Gaussian Markov random fields (see Section 2.4.3).[2]

What seems to be astonishing for most people, however, is how it can be possible to infer edge directions from data – and this is in fact the crucial part of causal discovery in my opinion. The basic principle presented here was pointed out by Verma and Pearl [297] and Spirtes et al. [261, Principle II]. It might have been common knowledge before, as the generalisation of this notion, d-separation (introduced in the next section), predates these sources.

Consider the graphs in Figure 2.1. The first three graphs yield the same (in)dependences: A and C are dependent unconditionally, but independent when conditioning on B, so one could say that B mediates the correlation between A and C. The last graph behaves completely contrary: Unconditionally, A and C are independent. Knowing B however, renders A and C dependent, because both are (potential) causes.

For example, we assume that B is a very rare event, say wet sand in the Sahara, and C is a very rare cause of this event, say someone running a grass sprinkler in this desert. Let A be another cause of B, e. g. rain. Then, knowing that the sand is wet (B), makes A and C dependent, for then the fact that it has not rained (A) suddenly makes it much more probable that C has occurred.

A subgraph like the last graph in Figure 2.1 is called a v-structure. By analysing the conditional (in)dependences of the nodes in the graph, the v-structures can be identified as induced subgraphs and the edges can be directed accordingly. Some of the other edges can subsequently be directed to avoid cycles and v-structures that are not justified by the data. A complete set of rules for this has been provided by Meek [181] (see Figure 2.2).

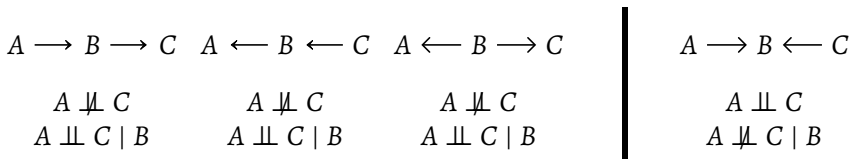


Figure 2.1: Origin of Direction

[2] Yet more surprisingly, it is also possible to regard and even learn causal models with only two variables, as discussed, for example, in [214, Chapters 3-4]

2.2.3 Assessing Independences in Graphs: d-Separation

We have seen in the previous section that v-structures play an important role in determining whether two nodes that are indirectly connected via a third variable are dependent or independent. This observation can be generalised to two arbitrary nodes in an arbitrary graph, leading to the notion of d-separation, which was first described in [199] and later formalised in [203]. It is introduced as a "direction-dependent criterion of connectivity" that extends the notion for undirected models, namely that Z intercepts all paths between the nodes of X and Y [203, p. 92].

Definition 1 *(d-separation).*
In a DAG G, a walk is blocked by a set Z if it contains a collider C that is not in Z or a non-collider X that is in Z. Otherwise it is open with respect to Z.

Two nodes X and Y are d-separated given a (possibly empty) set of nodes Z, written $X \perp\!\!\!\perp_G Y \mid Z$, if every walk between X and Y is blocked by Z (see below). Otherwise they are not d-separated, written as $X \not\perp\!\!\!\perp_G Y \mid Z$. (If the graph is unambiguous, and no distinction from an independence in a probability distribution is necessary, d-separation is also written using "$\not\perp\!\!\!\perp$" or "$\perp\!\!\!\perp$" without an index.)

In an alternative definition with paths instead of walks, colliders can not only be unblocked by being in Z themselves, but also if one of their descendants is in Z. In our definition this case is elegantly covered, because the walk can then be extended to the dependent and back to the original collider – which on this walk then appears twice but never as a collider. Instead, the descendent then is a collider on the walk – but, as it is in Z, does not block it.

Both criteria are formulated in [273, Section 1.8.1], where one can also find the equivalent d-separation criteria by Lauritzen *et al.* [163, 58], called the moralisation criterion, and a criterion independently suggested by Massey [180] and Darwiche [65].

The d-separation criterion captures exactly the dependance and independence information in the graph [110], [203], [296], [109] and Meek [185].

Extending the graph theoretical interpretation of the independence axioms by Pearl and Paz [207] (see Equation (2.3)) d-separation can be seen as a special version of separation in a graphs (see, for example, [77]).

If $X \perp\!\!\!\perp_G Y \mid Z$ (in a distribution or by d-separation in a graph), we will say that Z separates X and Y and call it a separating set, avoiding the graph theoretic term *separator*, which is used in the original sense (not with regard to d-separation), for example in [309]. Whenever we denote a set in the form S_{XY}, we refer to a set that separates X and Y. Whenever two variables X and Y are not d-separated given some set Z, there is an open walk between them. This walk can be simplified to a path (matching the path definition of d-separation). We will call such a path a d-path (between X and Y with respect to Z).

There is an elegant and efficient algorithm for deciding d-separation, which was given by Shachter [243]. It can be thought of as a modified breadth-first search, and has the same linear asymptotic running time (in the number of edges).

2.3
The Consistency of Independences in Distributions and Graphs

Having defined conditional independence in probability distributions (in Section 2.1) and graphs (in Section 2.2), our aim is now to find a graph that represents the independences in the data; more precisely, the independences which are inferred to hold in the distribution from which the data have been generated. The correspondence between the independences (and dependences) in distribution and graph is made up of two components:

1. All the dependences that hold in the distribution hold in the graph.
2. All the independences that hold in the distribution hold in the graph.

As the negation of dependence is an independence and vice versa, the contrapositions of these statements, which are logically equivalent, read as follows:

1. All the independences that hold in the graph hold in the distribution.
2. All the dependences that hold in the graph hold in the distribution.

Statement 1 is referred to as the Markov property (see also Definition 3) in the literature and constitutes the basic assumption about causal models. It is presented in all its canonical formulations in the next section.

Statement 2, known as faithfulness (see Definition 5), is not as self-evident, which will be shown in Section 2.3.2. Anyway, it is a common assumption of causal structure learning algorithms, which may be the cause of some problems we encounter. This is discussed in Section 5.2.1.

In accordance with the second, contrapositional formulation above, Pearl [203, S. 92] uses the term dependence map for models that fulfil the Markov property and independency map for faithful models. A graph model that has both properties is then said to be a perfect map of the probability distribution.

2.3.1
The Markov Property

In its general meaning, the Markov property describes "memorylessness": The future development of a process does not depend on the past, but only on the present. Applied to directed causal models, this property is usually formulated as "every node is independent of its non-descendants given its parents". That is to say, every causal influence into a node is mediated by the parents. Since every walk from a node X to its non-descendants must pass through one of its parents Pa_i, which never constitute a colliders on such a path due to the direction of the edge from a parent to X, this property can easily be shown to hold in a DAG via d-separation.

The Markov property becomes more meaningful in this context when it is interpreted as a measure of the compatibility of a graph and a probability distribution:

Definition 2 *(Local Markov property).*
A probability distribution P has the local Markov property with respect to a DAG G (and

vice versa) if each variable is independent (in P) of its non-descendents given its parents (in G).

As we have seen, the non-descendants of a node X_i are d-separated from X_i by its parents Pa_i. It can be shown [259] that the non-descendants can be replaced by *all nodes that X_i is d-separated from*, without changing the criterion (see Theorem 2.1). That means that the local Markov property is equivalent to the global Markov property in DAGs: Each variable is independent (in P) of all the variables it is d-separated from (in G) given some set Z. More precisely, this can be formulated as follows.

Definition 3 *(Global Markov property).*
A probability distribution P has the global Markov property with respect to a DAG G (and vice versa) if every d-separation $X \perp\!\!\!\perp_G Y \mid Z$ in G implies the conditional independence $X \perp\!\!\!\perp Y \mid Z$ in P.

Factorisation

If we recursively apply the definition of a conditional probability, or its equivalent form known as the chain rule, we find that for any permutation of the variables $X_1, X_2, \dots X_n$,

$$\begin{aligned} P(X_1, X_2, \dots X_n) &= P(X_1 \mid X_2, X_3 \dots X_n) \cdot P(X_2, X_3 \dots X_n) \\ &= P(X_1 \mid X_2, X_3 \dots X_n) \cdot P(X_2 \mid X_3, X_4 \dots X_n) \cdot P(X_3, X_4 \dots X_n) \\ &= P(X_1 \mid X_2, X_3 \dots X_n) \cdot \dots \cdot P(X_{n-2} \mid X_{n-1}, X_n) \cdot P(X_{n-1} \mid X_n). \end{aligned} \tag{2.7}$$

The right-hand side of the formula is called a factorisation of the joint probability distribution. Because it is valid for every permutation of the variables, every joint probability distribution has many different factorisations. If we choose a reverse topological ordering of the variables (as is possible in DAGs), every variable X_i appears in the factorisation conditional on a set of variables that does not contain a descendant of X_i. Thus, the local Markov property yields independence of all these variables – except for the parents – given the parents:

$$X_i \perp\!\!\!\perp \{X_{i+1}, X_{i+2}, \dots, X_{i+n}\} \setminus pa_i \mid pa_i.$$

According to the definition of conditional independence we can therefore drop all these variables $\{X_{i+1}, X_{i+2}, \dots, X_{i+n}\} \setminus pa_i$ from the conditions (assuming that all the conditional independences are non-zero) and are left with the factors $P(X_i \mid pa_i)$, which yields the factorisation formulation of the Markov property.

Definition 4 *(Markov factorisation property).*
A probability distribution with the density P has the Markov factorisation property with respect to a DAG G if it factorises according to G, that is:

$$P(X_1, X_2, \dots X_n) = \prod_{i=1}^{n} P(X_i \mid pa_i).$$

This Markov factorisation property is equivalent to the above Markov properties in DAGs [164, Section 4]. Hence it is true in general that a probability distribution P has the global Markov property with respect to a DAG G if and only if it has the local Markov property with respect to G, which it has if and only if its density factories according to G.

This is summarised in the following theorem, which is proven for example in [163]:

THEOREM 2.1.
Given a probability distribution P and a DAG *G, the following are equivalent:*

1. *P has the local Markov property with respect to G.*
2. *P has the global Markov property with respect to G.*
3. *P factorises according to G.*

Since we do not need to distinguish between the different incarnations of the Markov condition, we will simply say, that P and G are Markovian to each other if they fulfil any of the properties in the theorem.

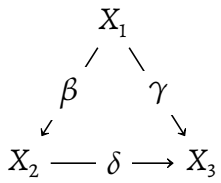
2.3.2 FAITHFULNESS

In the previous section, we have seen a basic property that describes the compatibility of a probability distribution and a DAG, the Markov property, in several formulations. However this property constitutes only half of what we would demand for a graph to be a perfect map of a probability distribution.

We also require the converse of the global Markov property to hold:

Definition 5 (Faithfulness).
A probability distribution P is faithful to a DAG G, if every dependence $X \not\perp\!\!\!\perp_G Y \mid Z$ in G implies the conditional dependence $X \not\perp\!\!\!\perp Y \mid Z$ in P.

While this claim seems as basic as the Markov property, it is not as self-evident. For example, it is problematic in cases where paths in the graph cancel out like in the following example. Consider this graph:



If $\gamma = -\beta\delta$, meaning that the direct effect of X_1 on X_3 is reverse but equal to the indirect effect mediated by X_2, then the effects along the two paths from X_1 to X_3, $X_1 \to X_3$ and $X_1 \to X_2 \to X_3$ would neutralise each other so that the total effect would be zero (in the Gaussian case that we consider here, direct effects correspond to edge weights; the total effect is computed over all directed, causal paths and is mathematically defined in Equation 2.10). This would entail an independence in the probability distribution, although in the graph X_1 and X_3 are marginally dependent (measured by d-separation).

Faithfulness is a very debated assumption that is the basis of many constraint-based causal structure learning algorithms, including the PC algorithm. This issue and the resulting problems will be further discussed in Section 3.2.1 and 5.2.1

2.4 The Consistency of Independences in Graphs

Although it had been known for a long time that probabilistic models can be statistically indistinguishable, such that their parameters [91] or structures [297] cannot be uniquely determined from data, it was only around the year 2000 that research focused on the advantages of succinct representations for these sets of indistinguishable models or *Markov equivalence classes* [11, 43]. The use of these representations has in turn been a major improvement for the computational efficiency and generality of many methods, not only in causal structure learning [43] but also in causal effect estimation [294, 210].

2.4.1 Markov Equivalence of Graphs

We have seen in Section 2.3.1 that every DAG trivially fulfils the (local) Markov property with respect to itself via d-separation, in the sense that each variable is independent, also with respect to d-separation in G, of its non-descendents given its parents. In fact, it is possible that two different DAGs behave completely equally with respect to d-separation and thus are Markovian to each other. Such DAGs, which represent exactly the same set of conditional independences and thus are Markovian to exactly the same set of distributions, are then said to be Markov equivalent to each other. For example, the three graphs on the left in Figure 2.1 are Markov equivalent.

Definition 6.
Two DAGs are Markov equivalent if they are Markov to exactly the same set of probability distributions.

As this relation is an equivalence not only by name, it is reflexive, transitive and symmetric [43] and thus partitions the space of DAGs into equivalence classes. Andersson, Madigan, and Perlman [11] summarise it as follows: "The collection of all possible [DAGs] for a given set of variates naturally coalesces into one or more classes of Markov-equivalent [DAGs], where all [DAGs] within a Markov equivalence class determine the same statistical model."

Definition 7.
The set of DAGs that are Markov equivalent to a DAG G is called the (Markov) equivalence class, sometimes abbreviated as MEC, of G.

Markov equivalence is a very important notion, especially in causal structure learning, as it entails indistinguishability: If two DAGs are in the same equivalence class, they are Markovian to the same distributions and thus explain these distributions equally well. We cannot distinguish between them only by the information given in the distribution.[3] That is why many causal structure learning algorithms, including the PC algorithm, actually do not learn a graph, but a set – in particular an equivalence class – of graphs. The characterisation of the indistinguishability of models by data was a major advancement especially in score-based learning (see Section 3.5.2), for which the introduction of a succinct representations for Markov equivalence classes was crucial.

[3] Spirtes et al. [263, Chapter 4] distinguish varies types of indistinguishability and term the type that we regard here *Strong Statistical Indistinguishability*.

In order to obtain such a representation we will first characterise the equivalence. In the following theorem, a directed edge $X_i - X_j$ is said to be covered (also referred to as *legally reversible*, e. g., in [273, Section 1.8.5]) if, apart from this edge, the parent sets of X_i and X_j are equal, that is to say, $Pa_j = Pa_i \cup X_i$.

THEOREM 2.2.
Given two DAGS *G and G′, the following statements are equivalent.*

1. *G and G′ are Markov equivalent*
2. *G and G′ have the same skeleton and v-structures*
3. *There is a sequence of reversals of covered edges by which G can be transformed into G′ and vice versa.*

The characterisation of Markov equivalence 2. was found by Pearl and his coauthors (e. g. Verma and Pearl [297, Theorem 1]); for the broader class of chain graphs it was independently discovered by Frydenberg [103, Theorem 5.6]. A detailed proof of the equivalence of 1. and 2. can be found in [195, Theorem 2.4]. The characterisation 3. and its equivalence to 1. is due to Chickering [40]. Kočka, Bouckaert, and Studený [155] provide an alternative, constructive proof of the equivalence of all three statements.

2.4.2
Representations for Markov Equivalent Graphs

Along with the above characterisation 2 of Markov equivalence, Verma and Pearl [297] define the notions of *rudimentary* and *completed patterns*. Rudimentary patterns are what we call *patterns* in this thesis (following, e. g., [181]): partially directed graphs made up of the a graph's skeleton in which only v-structures are directed.

We first need to introduce some terminology regarding patterns. Like [195], we will call the edges that are oriented in the same direction in all DAGS in the equivalence class compelled edges. (This includes the edges that are part of v-structures.) The other edges are said to be reversible here. Note, however, that these edges are not identical to *covered* edges (that are called *legally reversible* in [273]) since Chickering's transformational characterisation (3 in Theorem 2.2) does not state that all covered edges can be flipped straight away. It might be necessary to flip some other edges first before a reversible edge becomes covered and can be flipped. Reversibility thus describes a global property with respect to the class of DAGS, namely that there is a DAG in the class where the respective edge has the reverse direction. In contrast, it is a local property for an edge to be covered since this means that the edge can be flipped in the current graph without "leaving the equivalence class".

Now, each edge of a pattern is either compelled or reversible. Verma and Pearl remark that since the construction of a (rudimentary) pattern matches the characterisation [195] of Markov equivalent DAGS (in Theorem 2.2), such a pattern uniquely represents a Markov equivalence class. However, there is not a one-to-one correspondence between *directed* or *undirected* and *compelled* or *reversible*: There might be undirected edges in the pattern that still are uniformly directed in all members of the equivalence class.

The fact that a (rudimentary) pattern uniquely identifies a Markov equivalence class implies that the direction of the compelled edges that are not part of v-structures must be completely determined by the structure of the pattern, along with the characteristics /

traits of the procedure by which a DAG is extracted from the representation. This is done by directing undirected edges subject to the following requirements:

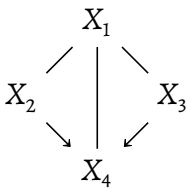
1. No new v-structures may form and
2. no directed cycles may emerge.

Accordingly, Verma and Pearl call the structure in which all compelled edges are directed a *completed pattern*. This structure is referred to as an *essential graph* in, for example, [11] and as a *DAG pattern* in [195] . Following [43] and most of the modern literature, we will call it a CPDAG (completed partially directed acyclic graph) in this thesis.

Verma and Pearl reduce the above requirements (for directing the compelled non-v-structure edges) to two conditions: An edge $X - Y$ is directed to $X \to Y$

1. if there is a node Z that is not adjacent to Y but is a parent of X (because otherwise a v-structure would form), or
2. if there is a directed path from X to Y (because otherwise a directed cycle would form).

It is obvious from the justifications in brackets, that these rules are sound, but it can be shown by a counterexample that they are not complete. Consider the following pattern:



Here, none of Verma and Pearl's rules applies: No v-structures can form from one of the undirected edge and one of the existing directed edges (because the triples $X_2 - X_4 - X_1$ and $X_3 - X_4 - X_1$ are shielded), and there are no directed paths of more than one edge, so rule 2 does not apply either. However, there is only one way to direct the edge $X_1 - X_4$ and that is $X_1 \to X_4$. Otherwise, both edges $X_2 - X_1$ and $X_3 - X_1$ would have to be directed towards X_1 to avoid cycles, which in turn, would produce a v-structure in $X_2 \to X_1 \leftarrow X_3$.

Meek's Rules

A set of complete rules for gradually turning a pattern into a CPDAG, including the above example as the third rule, was provided by the same authors some time later [295]. However, the completeness of these rules was only proven by Meek [182, 181][4], a few years later. He also showed that the fourth rule given by Verma and Pearl is necessary only if the initial graph is not a pattern, but contains some directed edges that are not part of v-structures. This can occur, for example, when some edge directions are known from beforehand (so called background knowledge), in addition to the v-structures determined from the data. All three remaining rules, which are now known as "Meek's rules", are shown in Figure 2.2. Whenever one of the configurations on the left appears as an induced subgraph in the regarded graph, an edge can be directed to create the right subgraph. When they are recursively applied, beginning with a pattern, the result is a CPDAG.

[4] This is the topic of [182], but is stated again in [181]. I will only cite the conference version [181] in the following, in case of redundancies.

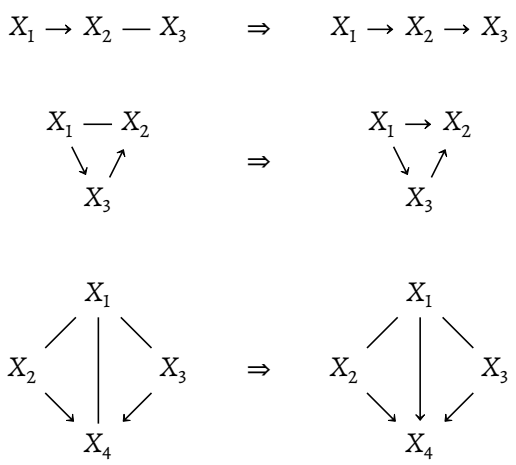


Figure 2.2: The complete and sound set of rules given by Meek [181] to transform a pattern into a CPDAG. When the induced substructures on the left are exhaustively replaced by the corresponding structures on the right (thereby directing one edge), any pattern is transformed into a CPDAG.

Another downside of Verma and Pearl's original rules that is remedied by Meek's rules is that the former are not local. This is important algorithmically because these rules have to be applied repeatedly to the graph, that is, to each undirected edge in it. Searching the whole DAG for a directed path every time can become very inefficient compared to only having to regard the two nodes incident to the respective edge and their neighbours, as sufficient for Meek's rules.

So far, we have approached the concept of a CPDAG, without giving a compact definition. To this end, Andersson, Madigan, and Perlman [11] give a complete local characterisation, which seems to be based on the characterisation of Markov equivalence by Chickering [40] (3 in Theorem 2.2), but regards the converse of covered edges, which they term protected: An edge $X_i \to X_j$ is protected if $Pa_i \neq Pa_j \setminus \{X_i\}$.

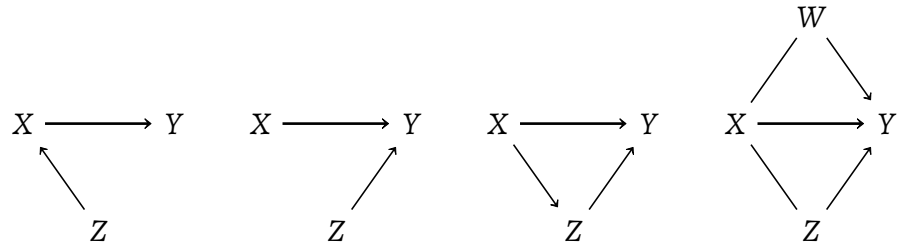
Since every edge that is covered can be reversed immediately (by Chickering's characterisation), the compelled edges must be among the protected edges. Andersson et al. first study the possible induced subgraphs containing a protected edge for DAGs and CPDAGs. In the next step, they analyse in which of these subgraphs v-structures or cycles might be induced by flipping the edge. While v-structures can easily be taken account of, since they are inherently local, it is rather surprising that Andersson et al. find a way to characterise cycles locally as well. Their approach can be explained as follows. We assume that there is an edge that is irreversible only because it blocks some directed cycle (which thus would have to be identified), and show that it would then be detected as non-reversible by local criteria, without having to consider directed paths p of arbitrary length.

Consider an edge $X \to Y$ that is irreversible only because it blocks some directed cycle. Because of the previous analysis (and because the edge is protected), Y must have a parent Z that is not a parent of X. Andersson et al. regard the case that Z is the last node on the path p from X to Y which would be closed to a cycle if the edge $X \to Y$ was flipped. Since we assume that no v-structures prevent the reversibility of the edge, there has to

be an edge $Z - X$ because otherwise there would be a v-structure $X \to Y \leftarrow Z$ which would be destroyed by flipping $X \to Y$. Now if the edge $Z - X$ was directed towards X, it would form a directed cycle with p, which is precluded in a DAG. It would thus have to be directed as $X \to Z$. This leaves us with a local cycle that is intercepted by $X \to Y$. Therefore, it is sufficient to regard induced subgraphs of this form, and not necessary to search for cycles of arbitrary length blocked by $X \to Y$.

The result of the analysis is the definition of induced subgraphs that render an edge strongly protected:

Definition 8.
An edge $X \to Y$ is strongly protected in a graph G if it occurs in at least one of the following induced subgraphs of G (where W and Z are different nodes):



In summary, we obtain the following CPDAG characterisation [11, Theorem 4.1]:

Theorem 2.3 ([11], Theorem 4.1.).
A graph is a CPDAG for some class of DAGs, if all of the following hold:

1. *It is a chain graph.*
2. *Every chain component is chordal.*
3. *The configuration $X \to Y - Z$ does not occur as an induced subgraph.*
4. *Every edge $X \to Y$ is strongly protected.*

Recall that a chain graph is a mixed graph without semi-directed cycle. Furthermore, as defined above, the chain components are the connected components of the undirected graph obtained by removing all the directed edges. It is necessary that these components are chordal because it is not possible to direct a non-chordal cycle of length at least four without producing either a directed cycle or a v-structure.

By construction, a CPDAG represents a class of Markov equivalent DAGs. Given a CPDAG G, we will denote this class by MEC(G).

From DAGs to CPDAGs

Theorem 2.3 also entails a polynomial time algorithm for constructing CPDAGs, which however has an asymptotic running time of $O(n^6)$ for graphs with n nodes. Given a DAG one can construct the CPDAG that represents the DAG's Markov equivalence class as follows: As long as there is a change in the graph, check all directed edges in the graph and convert every edge that is not strongly protected into a undirected one. [11, Section 5]

Another possibility with the same asymptotic running time to construct this CPDAG is to convert all edges that are not part of v-structures into undirected edges and then apply Meek's rules as long as possible.

From CPDAGs to DAGs

The reverse process is described by the consistent extension of a CPDAG, or, in general, any partially directed graph. This is defined in two steps as follows:

Definition 9.
An orientation $\vec{G}$ is obtained from a partially directed graph G by substituting all the undirected edges with directed ones. Then $\vec{G}$ is said to extend G.

Definition 10.
A (consistent) extension of a partially directed *acyclic* graph P is a DAG D that extends P and and contains exactly the same v-structures as P. The latter claim is also referred to as morality. That is, a consistent extension is a moral extension, and the resulting DAG D is moral.(Of course, being a DAG, it must also be acyclic.) If such an extension exists, we call P extendable, otherwise non-extendable. We will denote the set of DAGs that extend a partially directed graph G by $\text{EXT}(G)$.

Theorem 2.4.
The set of consistent extensions of a CPDAG G is exactly the Markov equivalence class represented by G, that is $\text{EXT}(G) = \text{MEC}(G)$.

An algorithm for deciding if a DAG has a consistent extension has been given by Dor and Tarsi [79]. The authors claim that it runs in time $O(n^3)$ but do not explain how this can be achieved. Chickering [43, Page 454f] analyses the computational complexity obtains a "a very loose upper bound" of $O(|\mathcal{V}|d_{max}^2 + |\mathcal{E}|d_{max}k^2)$, for a partially directed graph $G = (\mathcal{V}, \mathcal{E})$ with a maximal node degree d_{max}, so an implementation in time $O(n^4)$ is possible [308]. The function is implemented in `cextend` in the *R* package `bnlearn`. A version for exhaustive enumeration of these DAGs is given in the function `pdag2allDags` in the `pcalg`. Wienöbst, Bannach, and Liśkiewicz [308] propose a faster extension algorithm that runs in time $O(n^3)$.

Combinatorial Properties of Equivalence Classes

Chickering [43] emphasises the gain in computational efficiency when searching over the space of equivalence classes instead of the space of DAGs in score-based causal structure learning algorithms (see Section 3.5.2). A natural question is: *By which factor?* This can be answered by counting the (average) number of DAGs in an equivalence class. This is also relevant in causal inference methods that aggregate over Markov equivalence classes, e. g. IDA [172], where the feasibility depends directly on the size of the classes. On top of that, there are methods that rely on the sizes of Markov equivalence classes in their decisions, for example metrics that determine the best target for a systematic intervention, in order to learn causal structures more precisely than would be possible with observational data only [124, 123, 111].

In 2002, Gillispie and Perlman [112] computed the average size of Markov equivalence classes for DAGs with up to $n = 10$ nodes, obtaining a value of about 3.7. (The number of nodes considered might sound a bit meagre, but note that for $n = 10$ there are already 1118902054495975141 classes over which to average.) Wienöbst, Bannach, and Liśkiewicz [309] gave the first polynomial time algorithm to count the number of elements of Markov equivalence classes. To be precise, they provide algorithms to count $|\text{EXT}(G)|$ and sample from $\text{EXT}(G)$ for graph G. If G is a CPDAG.This yields the size of the equivalence class that G represents.

Similar counting problems concern for example the size of Markov equivalence classes [125] or, given a skeleton, the number of Markov equivalence classes by size or number of immoralities are [217].

2.4.3 Classification and Types of Graphical Models.

Although we generally model causal relationships as DAGs in this thesis, we have also seen some other graphical models so far: CPDAG, which represent sets of Markov equivalent DAGs and chain graphs, which generalise CPDAGs and can, for instance, represent subsets of Markov equivalence classes. All these graphs are partially directed graphs.

Mixed Graph Models

When considering latent confounders, i. e. variables that are not observable and thus not included in the data, usually mixed graphs are used to model the relationships between the variables. These do not only comprise directed and undirected edges but also bidirected edges, which represent the influence of a latent confounder that influences the two incident nodes (see the paragraph on causal sufficiency in Section **??**).

Ancestral graphs are mixed graphs that fulfil certain criteria, regarding, for example, acyclicity, designed to represent causality with confounding, that come with a generalised type of d-separation, called m-separation. A common subtype of ancestral graphs are *maximally ancestral graphs* (MAGs) [222]. Equivalence classes of MAGs can be represented by *partially ancestral graps* (PAGs) [316].

A good short overview and introduction to related graphical models, in particular ancestral graphs, can be found in [293].

All the causal structures described so far are a subtypes of *Bayesian Networks*. In a Bayesian Network which consists of a graph G and an probability distribution P, G describes how the distribution *factorises* (as in Equation (3.4)) and represents the independences in P (which we also saw in Section 2.3) [158]. The only difference in causal models is that we attribute a causal, meaning to the directed edges – they do not only describe a probabilistic connection. The undirected edges represent uncertainty, ambiguity or indifference about the edge direction.

Bayesian Networks can also be used for non-causal graphical models, but in this case, there are also undirected approaches, which will be briefly described in the next paragraph.

Undirected Graphical Models

The most prevalent group of undirected models are *Markov random fields*, including their various subtypes, such as the Gibbs random field and the Ising model [145, 121]. These models can be regarded as the predecessors of directed models [304]. As the name indicates, also in Markov random fields special versions of the Markov property hold: The pairwise Markov property – non-adjacent variables are conditionally independent –, the local Markov property – a variable is conditionally independent of all other variables given

its neighbours –, and the global Markov property – any two subsets of variables are conditionally independent given a separating subset. Depending on the subtype of model that is regarded, these properties might or might not be equal.

In Gaussian models, the covariance matrix measures the *marginal* dependence of the variables, i. e., the dependence without conditioning on any of the other variables. The inverse of the covariance matrix then describes the dependences conditioned on the set of all other variables. An undirected graphical model can be obtained by connecting each pair of nodes by an edge that have a non-zero entry in the inverse covariance matrix (and setting the edge weight to the value of this entry.)

Detailed introductions can be found for example in the books by Lauritzen [163], Edwards [85], Koller and Friedman [158] and Cox and Wermuth [59].

2.5
Structural Causal Models

The graphical models that we have considered so far have been structural models only, which represented the relationship of the variables in the observed distribution, but disregarded the processes that give rise to it. Structural causal models (SCMs) are more involved in that they also describe these *generative* processes. They are not types of graphs in essence, but closely related to graphical representations as we will see. As they also entail a defined distribution, they constitute an elegant, straightforward and intuitive way to establish the connection between graph structure and statistics. In this section, I will introduce SCMs and their main properties, roughly following Peters, Janzing, and Schölkopf [214].

The foundations of Structural Causal Models were laid by Sewell Wright in studies on the heredity of fur colours in guinea pigs (see the Introduction.) Although they have been used in particular fields, for example in economics [118] and the social sciences [81] since then, it was first in the 1980's that computer scientists and philosophers made these models popular in their quest to establish causality as a field of science.

An SCM describes the functional relationships between a set of variables, which constitute the nodes in the corresponding graph. However, in accordance with the causal interpretation, the relationships are not symmetrical. Hence, one should rather talk about assignments than equations. This is emphasised by the symbol $\leftarrow$ instead of $=$ as in the following definition.

Definition 11.
A structural causal model for a set of variables $\mathcal{V}$ consists of assignments for all nodes in $X_i \in \mathcal{V}$ which are dependent on some other nodes, namely the parents Pa_i of node X_i and noise:

$$X_i \leftarrow f_i(Pa_i, \varepsilon_i) \tag{2.8}$$

Here, the noise terms ε_i reflect exogenous influence on the variables, which, due to the assumption of causal sufficiency, are mutually independent. We also assume the assignments to be acyclic. Therefore, there is a variable ordering $X_1, X_2, \dots, X_n$, called the topological ordering of the variables, such that X_i only depends on (a subset of) $X_1, \dots, X_{i-1}$, for all $i \in \{1, \dots n\}$; that is, $Pa_i \subseteq \{X_1, X_2, \dots, X_{i-1}\}$.

Important subtypes of SCMs are obtained when the assignments are restricted to a certain form. In additive noise models (ANMs), the assignments are of the form

$$X_i \leftarrow f_i(Pa_i) + \varepsilon_i,$$

in causal additive models (CAMs) of the form

$$X_i \leftarrow \sum_{Y \in Pa_i} f_i(Y) + \varepsilon_i;$$

in linear Gaussian models, that we mainly regard in this thesis, they have the form:

$$X_i \leftarrow \sum_{Y_j \in Pa_i = \{Y_1, Y_2, \ldots, Y_l\}} \beta_{ij} \cdot Y_j + \varepsilon_i, \tag{2.9}$$

with Gaussian noise terms ε_i.

Pearl [200] remarks that this way of modelling processes of nature corresponds to the Laplacian concept of probability, in which randomness amounts to the effects of processes that are not understood, modelled or resolved (as opposed to, for example, modern quantum mechanics).

SCMs can be visualised as graphs with the nodes of nodes $\mathcal{V} = \{X_1, X_2, \ldots, X_n\}$ and, for each node X_i, a directed edge from each parent in Pa_i to X_i. An example is given in Figure 2.3. Since we assume the assignments to be acyclic, this graph will always be a DAG, with the topological ordering given in the definition of an SCM. However, Bongers et al. [22] recently regarded more general SCMs, which involve some inconvenient properties: For instance, they do not always have a solution and do not always entail unique distributions.

SCMs describe the generating processes of a distribution over these variables, a distribution that we would observe in experiments. In fact, an SCM uniquely defines such a distribution, which we will call the entailed distribution. This is the distribution of samples generated from the SCM in the following way, called ancestral sampling: First, the noise terms are sampled from their (predefined) distribution. Because of the acyclicity of the model, we can now proceed through the variables in their topological ordering and compute their values according to the assignments, as all noise terms and all their predecessors will have been evaluated beforehand.

$$\begin{aligned} X_1 &\leftarrow \varepsilon_1 \\ X_2 &\leftarrow \beta X_1 + \varepsilon_2 \\ X_3 &\leftarrow \gamma X_1 + \delta X_2 + \varepsilon_3 \end{aligned} \qquad \begin{array}{ccc} & X_1 & \\ \swarrow & & \searrow \\ X_2 & \longrightarrow & X_3 \end{array}$$

Figure 2.3: A simple linear SCM.

Now, the crucial finding is that the graph and the distribution entailed by an SCM are compatible: They are Markovian to each other.

Theorem 2.5 ([201, 296]).
Let P be the probability distribution entailed by an SCM with the graph G. Then P is Markovian to G.

Finally, SCMs not only unite graphs and distributions, they also express causation in a very intuitive way. In fact, the parents of a variable in an SCM are its *direct causes*. In linear SCMs, the coefficients of the parents describe the strength of the direct causal effect from the parent on the child. Based on this, we can carry on to don causal inference, and estimate total causal effects. This will be outlined in the next Section.

2.5.1 Outlook: Causal Inference in Structural Causal Models

While the consolidation of graphs and distributions in SCMs is convenient, their real merit is to represent causality, not only properties of observed distributions.[5] In fact, SCMs are one of the two main approaches to causality, besides the *potential-outcome framework* [229, 134, 226].

Because of this, they allow us to do inference on all levels of Pearl's metaphoric ladder of causation [206]. It is obvious that we can describe associations in a system as it is based on the assignments (rung 1). But we can also make predictions about the how the system would react to interventions (rung 2). I will briefly describe how even counterfactuals, which are located on rung 3, can be modelled with SCMs.

Interventions

In science, we are not only interested in describing systems, but we want to understand them. This often means to know how the system would react to a certain intervention. For example in medicine, we are not content when we now that a certain disease is associated with, say, an increased cholesterol level. We want to find out that if we lower the cholesterol level, the disease abates. (One could also say, we are interested in causality not association.) The same holds for chemistry, physics, economy, and other disciplines – we want to learn how we can interact with systems in order to employ them to our benefit. Apart from science, causal queries are typical in marketing, legal liability cases, policy making, as well as probabilistic expert systems. [158, 263].

Interventions to the mechanism that govern a regarded system can easily be modelled when just these mechanism constitute the model: In an SCM, interventions to single variables are modelled by replacing the original assignment for this variable with a new assignment that describes the mechanism by which the variable should be generated in the new system. This might even involve the set of parents of the variable, i. e. the graph, to change (otherwise, the intervention is said to be *imperfect*). To guarantee that the result of an intervention in an SCM is itself a proper SCM, the new graph needs to be acyclic. In addition, the error terms of all the assignments, changed or not, must be mutually independent. A very common way of intervening in a system is to set some variable to a fixed value – this is called an *atomic or ideal intervention*, as opposed to a *stochastic intervention*. For example, in a randomised controlled trial the different groups might receive different amounts of a drug, that is to say the variable *dose* would be set to different values. When considering proteins, one can model a mutation as setting some position to a certain amino acid. In the popular grass sprinkler example, where the season effects whether the grass sprinkler is turned on, which in turn effects whether the grass is wet,

[5] This fact is sometimes disregarded in the literature; see the Discussion in [200, Chapter 5].

an intervention could be to turn the sprinkler on. Such an intervention would supersede all the mechanism that would otherwise control the state of the grass sprinkler. In terms of the graph, all ingoing edges would be removed [262], or, more generally, replaced by a new set of parents.

In particular atomic interventions can be expressed mathematically with the so-called do-operator [200]. For example, turning the grass sprinkler on (an intervention) can be written as $\mathrm{do}(S = 1)$, where S is the variable that describes the state of the sprinkler. This is fundamentally different from seeing that the grass sprinkler is on $S = 1$ because the latter admits conclusions about the variables that are causes for S. For example if we also consider the variable W which describes whether it is winter, $P(W = 1 \mid S = 1)$ is certainly very small, while $P(W = 1 \mid \mathrm{do}(S = 1))$ is equal to $P(W = 1)$ because turning the grass sprinkler on does not have an effect on the season.

After the intervention, the new SCM thus generally has a different graph. It also entails a new distribution, the so-called *interventional distribution*, which represents the results of a theoretical experiment. Since the resulting model still is a valid SCM, we can easily determine its graph and distribution as described above.

In particular, causal effects are defined using interventions. There is a total causal effect of a variable X on a variable Y if and only if X and Y are not independent in the interventional distribution that arises when an atomic intervention is made on variable X. This in turn is equivalent to the existence of two values x' and x'' which, when X is set to either of them via intervention, yield different distributions for Y. While in general the size of the total causal effect depends on the intervention that is performed, it does not in the Gaussian case. Instead, in this case it is defined as the expected change in Y when X is increased by 1, that is,

$$\mathrm{E}(Y|\,\mathrm{do}(X = x + 1)) - \mathrm{E}(Y|\,\mathrm{do}(X = x)). \tag{2.10}$$

The classical method to calculate probabilities that include interventions like $\mathrm{do}(X = x)$ is the *do-calculus* or *intervention calculus* by Pearl [200]. The crucial observation for this is that, as we have seen, an intervention at a variable X eliminates all the previous causes of X.

Counterfactuals

While interventional distributions answer questions of the type "What if I ...?", counterfactual questions ask about what would have been. ...". The difference is that the question concerns a particular instantiation of the general system described by the SCM, which has already been observed. Peters, Janzing, and Schölkopf [214] give the example of a game of poker. The rules of the game define the links of the SCM, but the cards dealt to the players and the *flop* – the community pool in the middle of the table – define the instantiation of the game or, in statistical terms, the noise distribution. In poker, a player can, instead of raising his bet, drop out ("fold") at various stages with different amounts of knowledge about their cards and thus, chances. Now, they might wonder after having gained more information, for example after having seen the *flop*: "What would my chances be now if I had not folded?"

Formally, this amounts to retrieving information about the noise distributions (by conditioning), and then performing an intervention to change ones previous behaviour. This can also be seen as an intervention in a new, counterfactual SCM; it results in yet a

different, counterfactual distribution, which can be used to answer the counterfactual question. A good introductory example with discrete variables is given in [214, Example 6.18].

Although not a topic of this thesis, being able to model interventions and evaluate counterfactual questions is the main motivation for research with causal methods. It is discussed extensively, for example, in [201]. Since a possibility to evaluate basic interventional distributions is given by Pearl's intervention calculus, recent research has focused on more involved cases, for instance in the presence of unobserved and thus unalterable variables. Methods that can be used in such cases include covariate adjustment [210] in DAGs, CPDAGs, MAGs and PAGs, and instrumental variables in SCMs [70, 71]. If the true DAG is not known, a method called "intervention-calculus when the DAG is absent" (IDA) can be used. This method will be described and applied in Section 6.2.2.

3

Causal Structure Learning

In the previous chapter, it was discussed how probability distributions, and in particular their entailed conditional independences, can be modelled by graphs. We have also seen structural causal models (SCMs) which describe the structural relationships of a set of variables and thus entail a graph between them. (In the following, the more general term *graph* is used interchangeably for *causal structure*.) Furthermore, SCMs explicitly represent the pertaining generative processes, so that they can naturally express all types of causality, including interventions and counterfactual statements.

This chapter, like the rest of the thesis, is concerned with the first phase of empirical causal research – learning the model, the SCM, from given data. This is not only necessary for subsequent causal inference, but especially the causal structure, which expresses the causal relationships in a very intuitive way, can be useful per se in many applications. Causal structure learning is also the crucial step in the determination of the model; estimating the parameters to recover the full functional form of the model is generally comparatively easy.

Unfortunately, it is in general not possible to learn *the* true graph. Instead, there are classes of graphs that are indistinguishable given data, so-called Markov equivalence classes. In the case of DAGs, such a Markov equivalence class can be represented by a so-called CPDAG. Our aim is thus to learn a CPDAG, or more generally a partially directed graph, a PDG.

All steps of the causal structure learning process are described in this chapter. First, I will formally define the setting in Section 3.1. This is complemented by an overview of the assumptions that we make about the setting in Section 3.2 – assumptions that are frequently made, but seldom discussed or questioned. Before we start to learn the structure of the model, in Section 3.3 we first consider how we can learn its parameters when we already *know* the structure. Then we tackle the more difficult problem of learning the structure. The complexity of this problem is examined in Section 3.4, followed by a description of different algorithms that can solve it, in part heuristically, in Section 3.5. Finally, Section 3.6 describes how the learned graphs can be compared and evaluated, which will be one of the main topics of the next chapter.

3.1 Problem Definition

In this thesis we assume that we are given a data set in the form of a matrix with s observations for n variables. Contrary to some specific studies described in Section 3.5.4), we assume that there are no missing values in the data.

We often consider high-dimensional settings, in which the number of variables n is (much) larger than the number of observations in the data. For example, Kalisch and Bühlmann [148] assume that the number of variables can increase exponentially with a linear increase in the number of observations.

To avoid confusion, note that our notation for the sample size and the number of variables – that is, nodes in the structural model – is inspired by graph theory, where n typically stands for the number of nodes. Authors that are more affiliated with statistics, where n usually refers to the sample size, rather denote the number of variables by p and use n for the sample size. In statistics, high-dimensional settings are sometimes referred to in terms of the "large p, small n" paradigm [306, 242].

The data values can be of arbitrary type, discrete (ordered or categorical) or continuous. In this thesis, we are usually concerned with continuous data, although it might be more convenient to regard discrete, sometimes binary variables in examples. Many causal structure learning algorithms are general with respect to the type of input data, but sometimes, including in the PC algorithm, all variables are required to have the same distribution. In constraint-based methods, the interface between the data and the algorithm is constituted by the conditional independence test, which must be chosen appropriate to the data. In score-based methods, a scoring function evaluates the fit of graph and data; the type of data must thus be taken into account in the design of this function.

The result of a causal structure learning algorithm is a representation of the inferred causal relationships between the variables given in terms of a graph. Because not all causal graphs can be distinguished on the basis of observational data, such a graph usually represents a class of causal structures, in this thesis, a class of DAGs. A class of indistinguishable DAGs can be represented by a so-called CPDAG. Since in practice, it is necessary to take errors into account, we will relax the claim of learning a CPDAG, and do not assume the output to be acyclic. On top of that, we will allow an additional type of edges in the output graph, referred to as *conflict edges* (see Algorithm 1 below). These convey information about errors that were detected during the computation. As detailed in Section 5.2.1, conflict edges do not only make these errors explicit, but also allow handling them in a rational way.

Sometimes the true underlying graph is known, for example when the data were explicitly generated from a DAG by ancestral sampling (see Section 2.5). We then refer to it as the true graph or true DAG . The graph output by the learning algorithm is then termed the estimated or learned graph or DAG .

Conceptually, conflict edges represent edges whose direction could not be established, similar to undirected edges. However, unlike the latter, conflict edges have been found to *necessarily* (not potentially) be directed in both the one and the other direction – a contradiction. In this thesis, we will graphically represent conflict edges like this: $\leftrightsquigarrow$.

Formally, the problem of causal structure learning is a "model learning" problem [158]. It is customary to view learning as an optimisation problem: The *hypothesis space* is in this

case constituted by the candidate models, and can be equipped with an objective function to measure a model's quality.

As discussed in [158, Section 16.2], it is not self-evident how such an objective function should be chosen. The best model might for example be the one that entails a probability distribution closest to the observed distribution in the data. One could also define a specific probability that is of certain interest in the application, and try to reproduce this probability as well as possible. With due regard to indistinguishability (confer Section 2.4), it also makes sense to learn a class of models instead of a single best model. If interventional data are available or can be obtained, it is possible to obtain a more fine-tuned partition into Markov equivalence classes and thus more precise identifiability. This is not pursued further here, but is investigated in the literature, e. g., [123, 314, 154, 252].

Another approach would be to learn a density function over all models rather than only its optimum. Although this seems infeasible when considering the size of the space of models (see Section 3.4), there are so called Bayesian model averaging methods that pursue this approach. [158, Section 18.5].

However, when learning from limited data we face well-known problems in learning theory, such as *overfitting*. If we do not restrict the model class, we might learn a model that fits the training data perfectly, but does not capture its characteristic features. In the structure learning case, this corresponds to learning a complete graph, which through its high number of parameters is able to express more distributions than any other model.

The above considerations are closely connected to so-called score-based causal structure learning methods (see Section 3.5.2), in which a regulation or penalty term prevents overfitting. In constraint-based methods, like the PC algorithm, overfitting is not an issue. Instead, the sparsity of the model can and must be regulated by means of a hyperparameter. This is a parameter not of the model itself, but of the algorithm that learns it, and it must be prespecified by the user. It is usually denoted by α, because it constitutes the level of significance for the individual conditional independence tests.

3.2 Assumptions

The main objective of this thesis is to evaluate and analyse the performance of the PC algorithm in practice. To interpret the results and understand how errors come about, it is important to keep in mind which assumptions are implicitly presumed in the PC algorithm and other causal structure learning algorithms. The most important assumptions are listed and briefly discussed in this section. Violations of some of the assumptions listed here will be an issue for the analysis of errors occurring in the PC algorithm in Section 5.2.

When considering all the assumptions, many of which are often only poorly justifiable in applications, one might think that methods with weaker assumptions are generally preferable. However, one should rather see the assumptions as a means to remedy the underdetermination of the problem. Therefore, the weaker the assumptions, the less fine-grained the model that can be identified and the more vage the predictions that can

be made with it. This trade-off between assumptions and underdetermination is discussed for example by Eberhardt [82].

The assumption that is the very basis of our approach to causality is that the global Markov property defined in Definition 3 holds. This assumption is referred to as the causal Markov assumption. It follows immediately from our definition of SCMs, provided the error terms are independent [223]. Whether it should be considered an assumption or a principle inherent to causality is a philosophical question treated for example in [170, 263].

3.2.1 Faithfulness

The global Markov property, the basic principle of correspondence between graphs and probability distributions (see Definition 3), can be phrased as follows: What is independent in the graph must also be independent in the data. In the context of causal structure learning, we can think of this as "we cannot remove edges in the graph unless the data suggest it". Note that this is more important for being able to reproduce the data than the inverse: If links are removed, there are dependences which we cannot express with the graph; however if there are too many links, we can still set their strength to zero to obtain an independence.

According to this argumentation, we could always just learn complete graphs because they are Markovian to every distribution. But this would not give us any information about the relationships of the variables. This is why we would like to claim the converse of the Markov assumption as well, which is usually referred to as *faithfulness* (see Definition 5).

Faithfulness is a stronger assumption than the causal Markov assumption and one can easily construct examples where it fails because two causal paths exactly cancel each other (an analytical example is given in Section 2.3.2 on page 20).

A practical example about the (in)dependence of the contraceptive pill and the risk for thrombosis [132] is often given in the literature to illustrate how two causal paths can cancel each other (see Figure 3.1).

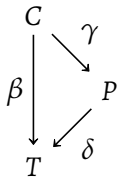


Figure 3.1: Example for how causal paths might cancel, leading to unfaithfulness.

On the one hand, the pill increases the risk for thrombosis directly. On the other hand it decreases the chance of becoming pregnant, which also would increase the risk for thromboses. It thus also *decreases* the risk. If the direct effects β, γ and δ have precisely the right strength, namely $\beta = \gamma \cdot \delta$, this situation could lead to the variables *contraceptive pill* and *thrombosis* being marginally independent although there exist two disjoint causal paths between them.

However, for each graph, there exist faithful distributions [185]. Moreover, the probability for cases of unfaithfulness to randomly occur is zero, or more formally, unfaithful distributions are of measure zero, with respect to the Lebesgue measure [263, 185]. However, Meek remarks that this does not necessarily mean that unfaithfulness is not an issue in causal structure learning. This issue is further discussed in the paragraph on errors du to unfaithfulness in Section 5.2.

3.2.2 Causal Sufficiency

In this thesis, we assume that the available data cover all the relevant variables, that is to say that there are no unobservable variables or latent confounders. This means that every common cause of several of the variables that is not constant or irrelevant in the population from which the sample is drawn is included in the data.

Formally, this is defined as follows: We require for causal sufficiency of a set of variables $\mathcal{V}$ for a population that if a variable Z is not in $\mathcal{V}$ and is a common cause of two or more variables in $\mathcal{V}$ that the joint probability of all variables in $\mathcal{V}$ be the same for each value of Z that occurs in the population [263].

This assumption entails that the data can be seen as independent samples from the distribution. It is usually stated that the data are independent and identically distributed, abbreviated as *i.i.d.*). This is a standard assumption in statistics, but it is nevertheless not self-evident in practice.

Frot, Nandy, and Maathuis [102] remark that "causal sufficiency is unrealistic in most applications", meaning that often latent confounders, also termed unobserved or hidden variables, do exist. Since they are not regarded, they are effectively marginalised out in the observed distribution – except if they act as a selection variable. Such variables determine whether an instance is included in the sample and thus are not marginalised out, but conditioned on in the data [53].

The most typical example for an inference fallacy caused by selection bias is described by *Berkson's paradox* [18]. Berkson noticed that two diseases (for example, diabetes and cholecystitis – an inflammation of the gallbladder) that are uncorrelated in the general population can appear correlated if only hospital patients are considered. This can be easily explained by what we know from Figure 2.1. Both diseases cause hospitalisation (although not necessarily), so that the three variables form a v-structure. Albeit independent unconditionally, the diseases thus become dependent when conditioning on the collider, hospitalisation. Intuitively, the situation can be explained as follows: People do not need to be taken to the hospital for either disease, but the few that suffer from both are likely to need inpatient treatment.

To model hidden variables, the underlying structure must be generalised from a DAG to a maximal ancestral graph (MAG) and the graphical criterion for independence from d- to m-separation [222]. Furthermore, the class of Markov equivalent MAGs can in this case be represented by a partial ancestral graph (PAG) (replacing a CPDAG in the causally sufficient case) [316]. Methods for learning such structures are briefly discussed in the paragraph on more general settings in the end of Section 3.5.1.

3.2.3 Acyclicity

Another assumption about the structure that we make here is acyclicity, meaning that we assume the graph of the underlying SCM to be a DAG. This as well is a debated assumption, as discussed in Section 5.1.

Although it might seem unintuitive to allow cyclic causal dependences at first glance, it is easy to find real-world-examples in which this is the case, with a typical economical example being price level and demand [22]. Feedback loops are also common in biology, as discussed further in Section 5.1. On top of that, even systems with an acyclic behaviour at each point in time can exhibit cycles when approximated over time or regarded in their equilibrium state [22].

For this reason, the properties of cyclic models, that are also known as non-recursive, have been studied [205], in particular for SCMs [202, 260, 96, 23, 22]. However, this is a highly non-trivial endeavour. Solutions of such SCMs then describe the equilibrium state of the system, which does not necessarily exist. Therefore, such models no longer entail a unique distribution. (Note that the process for sampling from the entailed distribution considered in Section 2.5 took advantage of the topological ordering of the nodes, which in the cyclic case does not exist.) Instead, there may be no or several consistent probability distributions [120]. Also, marginalisation is not possible in general but tied to further assumptions in cyclic models [22].

Although d-separation in principle is still useful in non-recursive models – as first shown by [260] in the linear case, and then in general in [259, 159] – the different Markov principles do not hold in general when cycles are allowed, even in linear models [260, 259]. In particular, the graph and distribution entailed by a SCM are not always Markov to one another.

In the meantime, Sprites' results have been generalised and the notion of σ-separation has been put forward [96].

Basic terminology of SCMs with cycles can be found in [22, Appendix A.1 and A.2], where also their properties are studied extensively.

3.2.4 Gaussianity

It is a very frequent assumption in machine learning and data science that data have a *Gaussian* (or *normal*) distribution. This assumption is often motivated by the central limit theorem, which states that the sum of a set of independent and identical distributed variables has a Gaussian distribution. But despite the universality of this assumption, it is often made for practical reasons rather than because of an analysis of the data.

As indicated above, continuous data, which can be seen as discrete data with an arbitrarily many categories, are not tractable as such. One possibility to solve this problem is to discretise the data, for example by binning it, but this always involves some amount of discretisation error. Another option is to assume that the distribution has a certain parameterised functional form, for which Gaussianity is a natural choice. But there are also learning-theoretical reasons to restrict distributions to certain classes: If the properties of the distribution that is to be learned from data were unrestricted, the best fit would

for the observed distribution would always be this distribution itself. In other words extreme overfitting would be the result. By restricting the class of distributions that the target distribution shall belong to, overfitting is avoided and instead a good fit is likely to be due to successful learning. For a more thorough discussion of this *bias-variance trade-off*, see [158, Chapter 16]. In SCMs, randomness is introduced via the noise terms such that Gaussianity of the data is a result of Gaussian noise terms.

The assumption that the data are Gaussian is seldom discussed or justified in the literature. It is often taken for granted that continuous variables have a Gaussian distribution (see, e. g. [195]).

In its predominant version, the PC algorithm is implemented with a test for vanishing partial correlation and is thus based on the assumption that the data have a Gaussian distribution. However, it has been adapted for the broader class of Gaussian copulas [122, 62, 63]. This is further discussed in the corresponding paragraph in Section **??**.

There is an approach to structure learning that *takes advantage* of non-Gaussianity, using *linear non-Gaussian acyclic models* (LiNGAMs, for short) [151, 245, 136]. Pearl [201, p. 64] gives an eidetic description of the principle of such methods as follows: "The idea is that in a linear model $X \to Y$ with non-Gaussian noise, variable Y is a linear combination of two independent noise terms. As a consequence, $P(Y)$ is a convolution of two non-Gaussian distributions and would be, figuratively speaking, 'more Gaussian' than $P(X)$. The relation of 'more Gaussian than' can be given precise numerical measure and used to infer directionality of certain arrows."

More formally, if X_2 is generated as $X_2 = \beta X_1 + \varepsilon_2$, where the noise ε_2 is independent of X_1, then a so-called backward model $X_1 = \gamma X_2 + \varepsilon_1$, with independent noise $(X_2 \perp\!\!\!\perp \varepsilon_1)$ exists if and only if ε_2 and X_1 are Gaussian. Otherwise the noise in the backward model will not be independent of X_2 and thus it can be distinguished from the original model. This renders also the structures $X_1 \to X_2$ (corresponding to $X_2 \leftarrow \beta X_1 + \varepsilon_2$) and $X_2 \to X_1$ (corresponding to $X_1 = \gamma X_2 + \varepsilon_1$) distinguishable (see Peters, Janzing, and Schölkopf [214], who also give a simple graphical example).

The LiNGAM method for to causal structure learning is very popular and has also been generalised in different ways, including the case where latent confounders might exist [137] and the non-linear setting [135].

3.2.5 Linearity

While the Gaussianity assumption is avoided in the concept of LiNGAMs (however substituted by the inverse assumption of non-Gaussiantiy), linearity is fundamental basis. This is also true for SCMs, which are often regarded in their linear form, that is with linear functions relating a variable to its parents. This yields and SCM of the form (2.9) (usually with Gaussian error terms).

Although this assumption might often be justified, there are plenty of examples for exponential relationships between variables in nature. Furthermore, thresholding might play a role when processes initiate other processes.

In the causal structure learning literature, relaxations of the linearity assumptions are often combined with other generalisations, for example in [259, 188, 187, 95].

3.2.6 Sparsity

Contrary to all the above assumptions, the assumption that the true graph is sparse is a bit vague and not used as a constraint in this thesis. It is, however, necessary to show that the PC algorithm is asymptotically correct. Kalisch and Bühlmann [148] prove uniform consistency of the PC algorithm assuming that the neighbourhoods in the true DAG are of lower order than the sample size. However, in their empirical evaluation they use very low values for the average neighbourhood size, namely 2 and 5, and refer to the latter as the dense case. They find what we will see in many of the empirical analyses of this theses: The results of the PC algorithm are considerably better when the true graph has an expected neighbourhood size of 2. No general assumptions about the sparsity of causal structures are made in this thesis, but rather sparse models are considered in the empirical study. Furthermore we should keep in mind that the Kalisch and Bühlmann's characterisation of the consistency of the PC algorithm depends on the sparseness of the model. The same holds for the algorithm's computational tractability, as explained in the description of the PC algorithm in Section 3.5.1.

Steck [267] remarks that in domains which require dense graphs, Bayesian networks might not be the adequate model for two reasons. Firstly, a very dense network is usually not very informative. On top of that, the learning procedure is problematic and also causal inference can be very inefficient or intractable in dense networks.

3.2.7 Correctness of Conditional Independence Tests

A common approach for causal discovery is to base the process solely on conditional independence information retrieved from the data (see Section 3.5.1). To prove the soundness, that is to say the correctness of such methods, it is often assumed that the conditional independence information in the data is perfect: No sampling errors occur and one can completely trust results of the conditional independence test. The test can then be seen as an oracle that provides conditional independence information. The setting is thus referred to as the oracle model. If it is further assumed that the distribution is Markovian and faithful to some known true DAG, the independence oracle can be implemented through d-separation in the this DAG.

Being given a conditional independence oracle is the only assumption that we usually do not make in this thesis; however, when we consider the soundness of an algorithm, the oracle model is usually implied.

Methods with Very Weak Assumptions

If one presupposes neither causal sufficiency nor acyclicity, many useful properties of SCMs are lost. Bongers et al. [22] have considered the properties of such general SCMs. They generalise notions like d-separation and the Markov and faithfulness condition.

There are also some relatively recent algorithms for learning cyclic structures in the presence of latent confounders, including other relaxations of the above assumptions, e. g., [95, 269]. See also the paragraph on more general settings in the end of Section 3.5.1.

Setting in This Thesis

We make all the above assumptions – except the oracle model – in this thesis. The validity of the assumptions will be discussed in Section 5.2. Especially faithfulness is a very strong assumption, but it is necessary to ensure that the algorithm cannot be mislead by spurious independences that are not reflected in the graph structure. It thus renders the Markov equivalence class of the true graph identifiable.

However, as mentioned before, we cannot learn a single *true* graph from observational data alone since it is indistinguishable from all other DAGs in its Markov equivalence class. This is summarised in the following theorem, which can also be found in [214, Lemma 7.2].

Theorem 3.1.

Let P be a distribution that is Markovian and faithful to a DAG $\mathcal{G}^*$. Then for every graph $\mathcal{G}$ that is in the Markov equivalence class of $\mathcal{G}^*$ there exists an SCM that entails P. Moreover, there is no graph $\mathcal{G}'$ that is not Markov equivalent to $\mathcal{G}^*$ such that P is Markovian and faithful to $\mathcal{G}'$.

3.3 Learning the Parameters

Having defined the problem and assumptions, we are now ready to actually learn the model. Essentially, we want to learn an SCM as defined in Definition 11. In practice, it makes sense to first learn the graph that describes the parent-relations in the SCM (like in Figure 2.3).Then, what is missing are the actual functions f_i in Equation (2.8). However, a certain functional form of the SCM that is to be learned is usually assumed, and thus only the parameters of this function remain to be determined. In linear (Gaussian) models, for example, values for the parameters β_{ij} (see Equation (2.9)) need to be estimated. These are usually seen as the edge weights of the graph structure and referred to as such throughout this thesis.

Although we will be mostly concerned with learning the structure of the causal model in this thesis, it is also necessary to learn its parameters for the estimation of effects, but also for the interpretation of the model itself.

Learning of the parameters is a model fitting problem, where the model class is given by the structure. Usually, the fitting is done using maximum likelihood estimation. In the following, I will give a very basic introduction to model fitting and maximum likelihood estimation that the experienced reader might wish to skip. On the other hand, a detailed introduction is given in [158, Section 17.1-17.2].

3.3.1 An Introduction to Maximum Likelihood Estimation and Bayesian Reasoning

Many models are general and have a number of parameters with which they can be calibrated for a particular set of data. This process is referred to as *fitting* the model to the data.

For example, let us assume we have a data set of lengths and durations of somebody's jogging tours. Intuitively, one would assume that the duration increases with tour length, and this relationship can be assumed to be linear, if we disregard the effects of, for example, fatigue, terrain and traffic. This assumption – that the relationship is linear – constitutes a simple model.

We can interpret this as an SCM with two variables X_1 and X_2, where X_1 is the length of the tour and X_2 is the duration of the run. Then we can use the assignment $X_2 \leftarrow v \cdot X_1 + \varepsilon_2$ for some noise function ε_2.

This model has one parameter v (apart from the noise): the slope of the linear function that describes the duration in terms of tour length, i. e. the mean running speed. (As opposed to other linear models, there is no offset as a second parameter in our model since a route of length zero will correspond to a duration of zero.)

If we are given data for different athletes, the model would likely hold equally well for all of them. But because of people's different athleticism, the slope would probably be different for each person: We would need information about this particular person's running speed in order to obtain a complete, parameterised, model. It suggests itself to gather this information in terms of a data set of this person's latest running tours along with their durations. We could then estimate the missing value in the model, the parameter v, from these data. That is what we do when we fit a model.

An important concept in model fitting is the maximum likelihood. It is a way to *approximate* the so-called posterior, the probability that a chosen (parameterised sub-) model is true given the data. Intuitively, it makes sense to maximise the posterior probability. We would like to choose values of the parameters (a parameterisation) such that the probability that this is the parameterisation that produced the data is maximised given the data that we obtained. If we could compute this probability for all parameterisations of the model, we could simply choose the most probable parameters for the model.

Unfortunately, the posterior probability is not generally computable. What we can compute, however, is the probability that the given data are generated from a model with a particular parameterisation. For example, let us fix a certain route in our running example and record the time that an athlete needs for this route. Now we could assume that this time is normally distributed: The athlete usually runs at a certain speed and might be slightly faster or slower, depending on daily form and other circumstances, but large deviations are increasingly improbable. This would correspond to a Gaussian model in which the time is described by a normal distribution whose mean depends on the athlete's personal speed and that has a standard deviation describing how severe the variations in speed are for this particular person. Both parameters, the mean and the standard deviation, would then need to be determined (fitted) for each athlete that we have data for to complete a model for that person. As described above, we would now like to compute the probability that the true model has a certain mean μ, say thirty minutes, and standard deviation σ, say two minutes, if we are given some specific data. It is unclear how we could compute this probability. If we, on the other hand, choose a μ and a σ, we can use the distribution's probability density function to calculate the probability to obtain the given sample, the above mentioned likelihood, which we want to use as an approximation.

To explain why this works, we need the equation known as Bayes' theorem, which stats that, for two random variables M and D

$$P(M \mid D) = \frac{P(D \mid M) \cdot P(M)}{P(D)}.$$

This amounts to swapping the event whose probability is to be computed with the condition, with an appropriate correction based on the marginal probabilities of the event and the condition.

Now we can think of the event D as obtaining the given data and the event M as the model (with certain fixed parameters) being true. Then we see that the posterior, that we *want* to compute, is the term on the left, and what we *can* compute, the likelihood, is one of the terms on the right. To be precise, the probability of obtaining the data given the model is equal to the probability of the model being true given the data, multiplied by the so-called prior probability of the model and divided by the (prior) probability of the data. Since the latter term is constant for all parameterisations that we might consider, it just constitutes a scaling factor and can be disregarded when the aim is to find the *maximum* of the term. Now, if also the prior probability of the *model* is uniform, that is to say equal for all models, then the model that maximises $P(D \mid M)$ will also maximise $P(M \mid D)$.

Under the assumption of a uniform prior, it is sensible to choose the model with the highest likelihood $P(D \mid M)$. This procedure is called maximum likelihood fitting, and is a special case of Bayesian statistics.

In summary, our task is now to compute the likelihood of a parameterised model and, moreover, to find the maximum of this likelihood over all possible values of the parameters. These possible values are given in terms of the parameter space, which is typically infinite. This maximisation is called maximum likelihood estimation, and is highly dependent on the type of model, that is to say the *model family*.

Given a model family, one needs to obtain the pertaining *likelihood function*. The likelihood function can be derived from the probability density function by interchanging parameters and argument: The probability density function describes the probability of an event given certain parameter values. The likelihood function $\mathcal{L}$, on the other hand, describes the "likelihood" of certain parameter values given the events (or data). Both are computed in the same way: The likelihood of a set of parameter values Θ can be derived from a set of data D as the probability for these data under the parameter values [158, Page 721]:

$$\mathcal{L}(\Theta \mid D) = P(\Theta \mid D).$$

In a maximum likelihood estimation, we want to find the value for which the likelihood function is maximal. Since the logarithm function is monotonically increasing, taking the (natural) logarithm does not affect the maximum (but simplifies the computation).

Thus the values for the parameters that we want to estimate, $\hat{\Theta}$, are chosen as:

$$\hat{\Theta} = \arg\max_{\Theta} \mathcal{L}(\Theta) = \arg\max_{\Theta} \log(\mathcal{L}(\Theta)) = \arg\max_{\Theta} \mathcal{LL}(\Theta),$$

where $\mathcal{LL}$ denotes the log-likelihood function.

3.3.2 Maximum Likelihood Estimation of the Parameters of an SCM

As mentioned above, given the causal structure it remains to determine the parameters of the functional relations of the variables dependent on their parents. For linear models, these parameters can be thought of as the edge weights of the structure.

For Bayesian networks and under the assumption of causal sufficiency, the likelihood (and thus the log likelihood) conveniently *decomposes* [256]. This means that it can be expressed as a product of separate terms for the individual variables and further, since the samples are assumed to independently drawn, into an individual term for each sample:

$$\mathcal{L}(\Theta) = \prod_{i=1}^{n} \prod_{j=1}^{s} P_\Theta(x_i^j \mid Pa_i = pa_i^j),$$

where P_Θ denotes the probability (density) function instantiated with the parameters Θ, x_i^j is the value of the variable X_i in the j-th sample, and pa_i^j are the values of the parents of X_i in this sample. Note that, as we saw above, the likelihood of the parameters is computed as the probability of the observed data that the model yields when it is parameterised with these parameters.

Owing to the decomposability the problem reduces to the computation of the parameters – in our case, weights – of the ingoing edges for each variable X_i, for which we only need to regard X_i and its parents Pa_i.

The form of the function P obviously depends on the assumptions made in the model. For discrete data, multinomial models are usually used, but the decomposition holds also for discrete density functions as, for example in the Gaussian case. The analytical derivation of the maximum of the likelihood function varies depending on the type of model, but closed forms exist for the mentioned model types and others [31].

Often, maximum likelihood estimation is a sufficient tool for the estimation of the parameters. It is, however, an inherent property of maximum likelihood estimation that the a priori probabilities of all parameter values are assumed to be equal. On top of that, varying confidences in the estimated values, depending on the size of the sample, cannot be represented. Small sample sizes can also cause problems when probabilities are undefined (compare Section 2.1.2) or imprecise, which can lead to overfitting. There are more sophisticated, often Bayesian, methods for parameter estimation that overcome these and other limitations [32, 255]. In particular, the well-known expectation maximisation algorithm for settings with incomplete data should be mentioned [76]. A good general introduction to the topic is given by [31] and [126].

3.4 Complexity of the Problem

Let us now turn to the core of the problem of learning a causal model, namely learning its structure. As indicated above, this problems is more difficult than the parameters learning problem. The nature of this complexity is regarded in this section.

One can get an impression of the difficulty of the problem of learning a DAG by analysing the size of the search space [267, 195]. Some bounds on the number of DAGs with n nodes,

$f_{\text{DAGs}}(n)$), can be easily obtained. Given n nodes, $2^{\binom{n}{2}}$ different DAGs can be constructed by considering all possible combinations of edges and directing them according to an arbitrary topological ordering of the nodes afterwards to obtain a DAG. This gives us a lower bound for f_{DAGs}. This bound is not tight as there may be several ways of directing the edges without creating directed cycles. On the other hand, the number of proper DAGs cannot exceed the number of directed graphs $3^{\binom{n}{2}}$ because each pair of nodes can be connected by an edge in one or the other direction, or not at all. Since not all directed graphs are acyclic, also this bound is not tight. The exact number of DAGs with n nodes can be recursively computed as follows [225, 224]:

$$
\begin{aligned}
f_{\text{DAGs}}(n) &= \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} \cdot f_{\text{DAGs}}(n-1) \quad \text{for } n \geq 2 \\
f_{\text{DAGs}}(0) &= 1 \\
f_{\text{DAGs}}(1) &= 1.
\end{aligned}
$$

This function increases super-exponentially. There are 4175098976430598143 different DAGs for $n = 10$. Furthermore, $f_{\text{DAGs}}(40)$ is of the order 10^{276} [251], far more than the estimated 10^{80} atoms in the universe [84].

This, in addition to the "involved" constraints and objectives that causal structure learning involves, renders it a "difficult" problem. To properly characterise the computational complexity of the problem, I will first briefly introduce some complexity theory basics (see also the "classical" book by Garey and Johnson [106]).

In complexity theory, problems are classified according to their tractability. An important class is P, which contains all problems that can be solved in polynomial time (on a deterministic machine), i. e., for which there exists an algorithm with a running time that is asymptotically bounded by a polynomial. The time complexity is then said to be in $O(n^k)$ for a constant k. An important question – in fact, one of the millennium problems endowed with one million dollars by the Clay Mathematics Institute – is the question whether the amendment "on a deterministic machine" really makes a difference. Otherwise, the class P would coincide with the more generally defined class NP, which allows non-determinism. One can think of non-determinism as guessing the right answer. For a problem to be in NP, however, this answer must be verifiable in polynomial time.

Typical graph problems in P are: finding a shortest path between two nodes[6] or finding a minimal spanning tree, that is, a connected undirected acyclic subgraph[7] with minimal edge weight which covers all the nodes. Such problems are usually thought of as feasible.

A relation between problems is given by reduction [290, 216]. If a problem A can be reduced to a problem B, this means that A can be solved with a subroutine that solves B. Then A is not "harder" to solve than B. Certain types of reductions are used to define the hierarchy of complexity classes. In this sense, NP hard problems are those that are at least as hard to solve then all other problems in NP. The so-called NP-complete problems are the problems that are NP-hard and also *contained* in NP, and thus solvable by a hypothetical nondeterministic machine in polynomial time. NP-complete problems are in practice

[6] Actually, complexity classes contain *decision* problems like "is there a shortest path of length l". We will ignore this technical detail here and refer to the corresponding search problems instead.

[7] Naturally, in this case the subgraph does not need to be induced, but consist of an arbitrary subset of the edges.

hard to solve (also characterised as *infeasible*), but it is often possible to solve small instances or find approximate or heuristic solutions. One can also regard sub-problems that are easier to solve or instances with certain properties. A prototypical NP-complete problem is SAT, determining a satisfying assignment for a Boolean formula. There are also many famous NP-complete graph problems, including the travelling salesman problem that is about finding a cycle in a graph which passes all nodes and has minimum weight; or the feedback arc set problem, in which the aim is to determine the minimal number of edges (a so-called feedback arc set) that must be removed from a directed graph so that it becomes acyclic [153].

It had been known that certain sub-problems for learning Bayesian Networks in specific settings or by certain strategies are NP-hard [133, 26, 41, 66, 183] even before Chickering, Heckerman, and Meek [46] proved this for a more general setting by a reduction from (a degree bounded version) of the feedback arc set problem. The reduction can be summarised as follows: For an input to the feedback arc set problem, a Bayesian network is constructed by replacing each directed edge in the graph with a small discrete Bayesian network. We will refer to these as the subnetworks in the following. This is done in such a way that an algorithm that learns a minimal Bayesian network would at the same time determine a minimal feedback arc set, thus solving this problem as well.

The proof also covers the case that the degree is bounded by a constant larger than 2 and in the oracle setting. (Apart from an independence oracle, the authors consider constrained inference and information oracles, see [46, Section 3.3].) But this proof is not completely general. First of all, it crucially depends on the minimality of the learned Bayesian network. This means that the network has a minimal number of parameters necessary to describe its distribution. For discrete networks, this number is given by

$$\sum_{X_i} (S_i \cdot \prod_{X_j \in Pa_i} S_j),$$

where S_i refers to the number of states that a variable X_i has. The reduction uses a hidden variable so that the structure learning algorithm cannot learn the correct network. Instead there are two possible structures for each subnetwork: (a) one that, containing the subgraph $\rightarrow\leftarrow$ blocks any directed cycle and (b) one that forms a directed path in the same direction as the edge that it replaces, but is smaller with respect to the number of parameters. An algorithm that learns a minimal Bayesian network would prefer structure (a) whenever it is not necessary to choose structure (b) to avoid a directed cycle. The edges corresponding to the substructures that have been learned as structure (b) then form a feedback arc set in the original graph.

Minimality usually plays a role in score-based causal structure learning algorithms where the number of parameters is used as a regulariser, that is to say a penalty term to prevent overfitting, but not in constraint-based algorithms like the PC algorithm.

Moreover, Claassen, Mooij, and Heskes [51] have pointed out that the result does not apply to algorithms that are able to learn hidden variables. Such algorithms can learn the correct substructure, which has even fewer parameters, so that the reduction would fail.

However, the hardness result in [46] excluded algorithms that assume faithfulness, which Claassen, Mooij, and Heskes [51] do, too. After all, in the construction described above, there is no faithful graph with only the observed variables. In this context, Chickering, Heckerman, and Meek [46] mention the PC algorithm, that in the faithful setting

solves the problem in polynomial time if the degree of the graph is bounded by a constant a priori. This will be further discussed in Section 4.1.2.

For now, we can summarise that the problem of structure learning by score-based methods from data is NP-complete for inconsistent distributions [158], that is when the distribution is not Markov and faithful to a DAG without hidden variables. For constraint-based methods, there are polynomial algorithms for the consistent case when the degree of the true and learned graphs are bounded. Next, we will describe the different approaches to causal structure learning in more detail.

3.5 Causal Structure Learning Algorithms

In Section 3.2.4, we have already considered the LiNGAM method for causal structure learning or causal discovery. This "SEM-based approach" [265] is special as it does not belong to one of the two main groups of causal structures learning algorithms, which are introduced here and then described in more detail in separate sections.

The first general approach is to view causal structure learning as a constraint satisfaction problem, where the goal is to find a graph that fulfils a number of conditional independence relations imposed by a probability distribution (given through the data). This approach is therefore sometimes called the independence-based approach. Canonical algorithms in this group include the inductive causation or IC algorithm [297, 208], the SGS algorithm [261] and the PC algorithm [263].

The other strategy is to measure the accordance of the graph and the data by a scoring function and search the space of graphs (or classes of graphs) for the one that optimises the function. For this, the BIC score, Bayesian scoring functions and information-theoretic scores have been used, among others (see Section 3.6.1).

As discussed in the previous section, causal structure learning is a hard problem. In the case of the PC algorithm this is remedies by applying a certain procedure by which the "combinatorial explosion" can be avoided in all cases where the maximal degree of the output graph is limited (see 4.1.2). The idea of limited node degree has also been applied in score-based methods to limit the search space [100, 99]. In addition to the restriction of the search space, efficient score-based algorithms search greedily, ruling out provable general consistency, but usually yielding good results in practice.

Many of the methods described here are implemented in the R packages `pcalg` [149] and `bnlearn` [239, 240, 192]. An extensive introduction to the topic is given by Neapolitan [195]. On top of that there are several recent surveys [61] [129] and evaluation studies [241, 54].

3.5.1 Constraint-based Causal Structure Learning Algorithms

Instead of using the distribution itself, constraint-based algorithms only work with the independences that are estimated to hold in it and try incorporate them all into the graph structure as introduced in Section 2.1.2. Because of this approach, they are sometimes also called independence-based methods (for example in [214]).

In 1983, Wermuth and Lauritzen [303, 305] proposed an algorithm in which a topological ordering $X_1, X_2, \ldots$ of the variables in the true graph is already known, for example through their temporal ordering. Clearly, this renders the problem significantly easier. Consequently this case was excluded by Chickering, Heckerman, and Meek [46] in their hardness result (see Section 3.4). In this case, two variables X_i and X_j are independent if and only if they are independent given all other nodes with a smaller index than j. Thus, to recover the causal DAG, it remains to iterate over the pairs of variables that may have an edge $X_i - X_j$ between them (that is, with $j > i$) and check the independence given all of the nodes with a smaller index than j.

The strong assumption of a known variable ordering was overcome by the IC (inductive causation) algorithm [297] and SGS algorithm [261, 263]. Both algorithms first learn a skeleton, then turn it into a pattern, and then try to direct the remaining edges in order to obtain a CPDAG. This basic design is identical to that of the PC algorithm. However, the algorithms differ in the implementation of the individual phases as we will see in the following.

The first phase of the IC algorithm is based on a finding that was independently obtained by [261, Principle 1], [258] and [297, Lemma 1]:

THEOREM 3.2 ([258, 297]).
There is an edge between X and Y in $\mathcal{G}$ if and only if for each set $S \subseteq \mathcal{V} \setminus \{X, Y\}$, X and Y are not d-separated given S.

Recall from Section 2.2.3 that we refer to a separating set for the nodes X and Y as $S_{X,Y}$. Theorem 3.2 implies a brute-force algorithm for the computation of the skeleton, which is used in the IC algorithm:

```
Start with an empty graph 𝒢 = (𝒱, ℰ), ℰ = ∅
for every pair (X, Y) ∈ 𝒱² do
    if X ⫫ Y | S for every subset S ⊆ 𝒱 \ {X, Y} then
        add the edge X − Y
    end
end
```

The corresponding procedure in the SGS algorithm is in some sense inverse, starting with a complete graph and removing edges $X - Y$ when a separating set S_{XY} is found.

Furthermore, the implementation of the second phase differs a bit, although in both algorithms it is based on the same principle [297, Lemma 2], [263, Theorem 3.4 (ii)]:

THEOREM 3.3 ([297], [263]).
The chain $X - Y - Z$, where X and Z are non-adjacent, has to be directed as a v-structure if and only if X and Z are not d-separated by any set containing Y.

In the SGS algorithm, it is checked that all subsets that contain Y do not separate X and Z. This is solved a bit more elegantly and efficiently in the IC algorithm. Since X and Z are not adjacent, in phase 1 a separating set S_{XZ} must have been found. If the regarded structure was a v-structures, the d-path $X - Y - Z$ would be opened by conditioning on Y. So a separating set for X and Z, in particular S_{XZ}, would never contain Y, regardless of the rest of the separating set that might be needed to block other d-paths. It is thus sufficient to check whether $Y \in S_{XZ}$, and direct $X - Y - Z$ to $X \to Y \leftarrow Z$ otherwise. (Of course, all this only applies in the consistent case, that is, when the distribution is Markovian

and faithful to a DAG and all independences are estimated correctly. The so-called sample case, where this is not assumed, is discussed in Section 4.1.1.)

The last phase of the SGS algorithm and IC algorithm is meant to convert the pattern resulting from phase 2 into a CPDAG. But not only was the terminology different when the algorithms were proposed (and the term *pattern* used to refer to a CPDAG), CPDAGs were also not that well characterised. In particular, the rules used in both algorithms to direct the pattern are neither optimal with respect to computation effort needed to apply them, nor complete, meaning that they do not necessarily identify all the edges that can be directed (and should, in order for the result to be a proper CPDAG). This issue was later handled by Andersson, Madigan, and Perlman [11] and Meek [181]. The former obtain a complete characterisation that entails an algorithm to direct a pattern into a CPDAG, the latter provides a provably complete set of rules for this last phase of the above algorithms based on a superset of rules proposed by [295], as described in detail in Section 2.4.2.

The first phase of these algorithms is very costly. They generally have exponential running time because for each of the $\binom{n}{2}$ pairs of nodes 2^{n-2} potential separating sets have to be considered. To reduce the computational complexity in practice, Spirtes et al. [261] propose to try and limit the search space, for example by including prior knowledge.

Verma and Pearl [297] propose an algorithmic modification which relies on the fact that separating sets S_{XY} need only be searched for among the parents of X or Y. However, Verma and Pearl's proposal involved finding cliques (an NP-hard problem) in a probably quite dense graph and considering all subsets of nodes in this clique, and is probably not very suitable in practice. Yet, the idea that only the parents need to be considered [297, Lemma 1] turns out to be worthwhile and leads to the PC algorithm [263].

Theorem 3.4 ([297]).
There is no edge between X and Y in $\mathcal{G}$ iff there exists a set

$$S_{XY} \subseteq pa(X) \text{ or } S_{XY} \subseteq pa(Y)$$

such that X and Y are d-separated given S_{XY}.

Like the SGS algorithm, the PC algorithm starts with a complete graph. Thus considering only the neighbours when searching for a separating set does not improve the performance in the beginning. However, as soon as some edges have been removed, the sets of nodes that have to be considered to find separating sets decreases, for sparse graphs significantly. However, this procedure would be useless if finding the first edge that can be removed from the initial graph is infeasible. Here, another trick is used in the PC algorithm. To reduce the combinatorial effort in the beginning, marginal independence is considered first, and the size of the conditioning set is increased step by step. In the stage where the cardinality k is considered and the largest neighbourhood of a node is of the size N, $\binom{N}{k}$ separating sets have to be considered. By starting with $k = 0$, the maximal combinatorial complexity attained for large N and k is avoided: N has high values in the beginning and decreases gradually, while k increases. In practice, this means (for sufficiently sparse graphs) that large k are often not considered at all, because when k reaches high values, the neighbourhoods have already become too small to actually consider separating sets of size k. This is also a statistical advantage because conditional independence tests with a large conditioning set usually have low power, as briefly motivated in Section 2.1.2.

For the PC algorithm, the more efficient phase 2 from the IC algorithm is adopted and Meek's complete set of rules used. It can thus be seen as the improvement of the IC and the SGS algorithm. We will extensively analyse it in the following chapters, and discuss published and new extension.

The successive incrementation of the order of the tested conditional independences also has the advantage that it is easy to stop at a certain value of k. This is useful, for example, when the size of the neighbourhoods is known in advance. It is very beneficial for the performance, but also helps to avoid statistical errors if the power of high-order tests becomes too small. Some adaptations of the PC algorithm have been proposed that use only very low order conditional independence queries [33, 254, 310].

A related idea is to first determine for each node the set of potential parents by finding its so-called Markow blanket [204]. For some node X, the Markov blanket is defined as the set of nodes Y conditional on which X is independent of all other nodes. Due to the special purport of v-structures in the definition of d-separation, this includes not only the parents and children but also the spouses of X. From the Markov blanket, the parents and children can be selected by fewer and lower-order conditional independence tests than from the set of all other variables. As the first phase of the PC algorithm, this phase would then be followed by procedures to establish edge directions. The use of Markov blankets for causal structure learning was first proposed by Margaritis and Thrun [179] in terms of the so-called grow-shrink (GS) algorithm (see also [178]), which is theoretically sound and polynomial when the size of the Markov blankets is bounded. A similar approach with a more dynamic heuristic was suggested in [284]. [7] propose a general framework, called *generalised local learning* (GLL) for building causal structure learning algorithms based on Markov blankets and provide an extensive evaluation (albeit focused on variable selection, not causal structure learning) as well as analysis and extensions in a second paper [8]. The GLL framework includes a variety of algorithms that are sound under the faithfulness and causal consistency assumptions in the oracle model. Among them are three established algorithms: The Max-Min Parents and Children Algorithm MMPC [283], HITON-PC, where "Hiton" is greek for cloak [9], and Min-Max-Hill-climbing MMHC [287], a hybrid algorithm that uses MMPC as a subroutine, and will be introduced in Section 3.5.3.

Algorithms for More General Settings

The IC algorithm was originally proposed for the case where hidden variables may exist, but it was not completely correct. This was remedied in [298]. In the meantime, Spirtes, Glymour and Scheines had published the first edition of their book [263] and, for setting with latent confounders, proposed the FCI (fast causal inference) algorithm– an adaptation of the PC algorithm. Some time later, the FCI algorithm was in turn modified [264] in order to incorporate selection bias (see Section 3.2.2). After the characterisation of ancestral graph models by Richardson and Spirtes [222], Zhang [316] showed that the output of the FCI algorithm can be interpreted as a PAG and provided complete orientation rules [317]. The name *fast causal inference* is very euphemistic; the FCI algorithm and its adaptations are significantly slower than algorithms like the PC algorithm [266]. However, recent work aimed at improving the performance and worst case running time [53, 51].

In addition to these time considerations, the extension to hidden variables comes at a cost, as is to be expected with regard to the trade-off between assumptions and under-

determination discussed in Section 3.2. Methods that allow arbitrary hidden structures often produce far too many bidirected edges [266]. That is, they are not as powerful in detecting causal effects, as Frot, Nandy, and Maathuis [102] remark. Therefore, the authors propose a method for a more restricted case in which only *some* latent confounders may exist.

Another extension of our setting is to learn cyclic models. Richardson [223] first studies this matter and gives an asymptotically correct algorithm called CCD that is polynomial for graphs with bounded degree. Mooij and Heskes [187] propose a method that does not assume linearity and can make use of a mixture of observational and interventional data.

Recently, some very general methods have been proposed that can learn models with cycles and latent confounders, for example a PC-style algorithm by Strobl [269] called CCI, which can also handle selection bias. More methods for general settings are discussed in the end of Section 3.5.3.

3.5.2 Score-based Causal Structure Leaning Algorithms

The contrary approach to what was discussed in the previous section is usually referred to as score-based. The underlying idea is to describe the compatibility of a graph and the given distribution by a scoring function and then search the space of graphs for one that maximises this function. Formally given the data D over the variables $\mathcal{V}$ and a scoring function S, we want to find the DAG $\mathcal{G}^*$:

$$\mathcal{G}^* = \arg\max_{\text{DAG } \mathcal{G} \text{ over } \mathcal{V}} S(\mathcal{G}, D).$$

As was discussed in Section 3.4, this problem is NP-complete and thus no polynomial time algorithms can exist, unless P = NP.

Different possibilities have been pursued to make the process feasible anyway; an overview is given in [7]:

- Restrict the types of distributions that are considered.
- Include domain knowledge to restrict the search space.
- Learn only a part of the network (specifically or unspecifically).
- Learn only the skeleton of the model.
- Include known causal relationships to restrict the search space.

However, in addition to considering only distributions that are faithful to a DAG over the observed variables, two other approaches are more predominant in the literature, as we will see in the following:

- Use heuristic approaches.
- Restrict the analysis to relatively small settings.

In the context of score-based learning, the characteristics of the scoring function play a role. In particular *decomposability* of the score is crucial for the feasibility of the search. (This property is similar to the decomposability of the likelihood function considered in 3.3.2.) A scoring function S, is decomposable if the score of the complete structure

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be computed as the sum over the nodes, where the individual terms depend only on a node and its respective parents:

$$S(\mathcal{G}) = \sum_{X_i \in \mathcal{V}} S_{\text{local}}(X_i \mid Pa_i). \tag{3.1}$$

This can be seen as a Markov property for scores, analogous to the factorisation of distributions detailed in Section 3.4. It entails that the score can be computed locally, individually for each node. The main advantage is that it is not necessary to compute the whole score anew every time something in the structure is changed during the search. In most useful search algorithms, the search often proceeds in relatively small steps like adding, deleting or flipping an edge. Such an operation only changes the parents of one, in the latter case of two variables. This means that only two terms of the sum change. Since usually the relative score is of interest to determine whether the algorithm shall proceed in the considered direction, the score of the rest of the graph is irrelevant anyway. Having to compare only the scores on the nodes that are involved in the operation of the current step significantly reduces the cost of the evaluation of the scoring function – the most expensive part of the search.

As a consequence, most of the scores used in score-based methods today are decomposable, including the ones briefly described below.

Another important property of a scoring function is the so-called score equivalence. This amounts to Markov equivalent graphs, in our case DAGs, being scored equally: Given an equivalent score S and a DAG $\mathcal{G}$, for every DAG $\mathcal{G}' \in \text{MEC}(G)$ it holds that:

$$S(\mathcal{G}) = S(\mathcal{G}').$$

This condition is reasonable when we want to learn a graph that might have generated a set of data. From Theorem 3.1, we know that we cannot discern such graphs on the basis of observational data. It thus makes sense not express preferences among these models that are fundamentally indistinguishable. This assumption also has a practical implication that is described below.

In this section, we first focus on the algorithmic aspects of the methods and name scoring functions only if an algorithm uses a specific one. Important scoring functions are described in more detail later, in Section 3.6.1.

Cooper and Herskovits [55] were the first to use a Bayesian score to learn Bayesian networks, resulting in an algorithm called K2. It is named after a previous, entropy-based method called Kutató (Hungarian for "explorer" or "investigator") [131]. Both algorithms use a greedy search strategy that assumes a given ordering of the variables. The possible parents of a node X_i are then the nodes $X_1, X_2, \ldots X_{i-1}$ (as in the Wermuth-Lauritzen algorithm described in the Section 3.5.1). Then, the algorithms start with an empty graph and iterate over the nodes. For each node, they determine which of the possible parents maximises the score and add an edge from this parent. This procedure is pursued until no single parent addition increases the score anymore. In K2, the score measures the posterior under a uniform prior. On top of that, the inclusion of hidden variables is discussed which is particularly difficult in score-based methods as it renders the search space infinite – at least when an arbitrary number of latent variables is allowed for.

Given the scoring function, the search problem is a very general one. The only crucial feature of the search is that it needs to be ensured that every graph that is considered is

acyclic. Therefore, many general optimisation techniques are applicable and have been applied to the problem.

Larranaga et al. [162] apply a genetic algorithm. They use the scoring function from the K2 algorithm but limit the space to DAGs with maximum in-degree four. This entails the necessity to check the offspring generated in each cycle and remove some of the ingoing edges of nodes with too many parents. This is either done at random or via a local optimiser that choses the remaining parents so that the score is maximised. Otherwise, the parent set can be randomly selected. On top of that, two different selection mechanisms are evaluated: Either the offspring completely replaces the parent generation in each round, or they are merged and the fittest half of the elements, a so-called elite, is carried over to the new generation. The authors find experimentally, on the basis of real-world data, that the use of both the local optimiser and the elite selection are beneficial. Another biology-inspired approach, to apply ant colony optimisation, is described in [74] and evaluated using the same scoring function.

Heckerman, Geiger, and Chickering [128] focus on a new scoring method, which they evaluate using different search techniques. This includes the one used by the K2 algorithm, both with the correct variable ordering and the reverse, worst possible, ordering to evaluate robustness. They compare with local search (that is, hill climbing) and iterated local search. In the latter case, at each restart the current structure is modified by 100 random edge modifications. Finally, they try an algorithm that is called simulated annealing, which models a physical cooling process in which particles gradually anneal and their motion becomes less stochastic: In the beginning, relatively large steps in the search space are allowed, potentially even decreasing the score. Afterwards the algorithm – hopefully – has escaped local maxima and is close to the global maximum. This is then approached via relatively small steps with high probability of increasing the score. The evaluation shows that the K2 algorithm, indeed, is strongly dependent upon the variable ordering. Given the correct ordering, it is perceptibly better than the other algorithms, while with the wrong ordering, it is far worse. On top of that, the authors conclude that the results produced by local search are not significantly worse than those of the other algorithms, despite a significantly shorter running time.

A very promising method was proposed by Bouckaert in his PhD thesis [24]: using the TABU search algorithm [113] to search the space of DAGs. The idea of this search method is to actively avoid local maxima that have already been found. Initially, the scoring function is optimised via hill climbing; when a local optimum is reached, a certain amount of steps are done regardless of the scoring function, but avoiding regions of the search space that have already been visited – thus the name TABU search. The purpose of the TABU search is to escape a local maximum and reach the vicinity of another, hopefully the global optimum. To increase the chances of finding the global maximum, modern implementations repeat the procedure several times and randomly perturb the resulting graph in each iteration to obtain the initial graph for the next iteration. This method has proved to yield very good results in recent studies, e.g. [54].

While the usual operations to move through the search space are the addition, removal or reversal of a directed edge [128], Moore and Wong [190] propose a different operation that facilitates larger steps in the search space, called optimal reinsertion (OR). It consists in choosing a so-called target node and removing it from the graph, along with all its adjacent edges. Then, the globally optimal in- and outgoing edges for the reinser-

tion of the target node are computed, subject to certain constraints. This technique improves the running time by one or two orders of magnitude as compared to conventional multi-restart hill climbing.

In addition to greedy search, new theoretical results like increasingly tight bounds on optimal scores and techniques for pruning of the search space [72, 232, 275] have made exact methods feasible for small or medium sized inputs, with up to a few dozen nodes. These methods are often based on linear programming (see Section 3.5.3), dynamic programming[157, 249, 156] and similar techniques[248].

Score-based Search over Other Structures than dags

Instead of searching through the space of DAGs, other search spaces are imaginable. It has been discussed before that under the assumptions made in this thesis, the causal structure is identifiable only up to Markov equivalence. Based on this it might seem unreasonable to search in a space where two neighbouring DAGs often are from the same equivalence class and thus not distinguishable anyway. On top of that, if we assume score equivalence, i. e. that the score yields the same result for these two DAGs, it is sensible to search in the space of equivalence classes instead. We also know from Section 2.4.2 that the average size of an equivalence class (at least for graphs with up to $n = 10$ nodes) is about 3.7 (see page 26), so this space would be smaller by a factor of about 4.

The idea has been put into practice by Acid and de Campos [1] and Chickering [42, 43]. Chickering's approach is to represent the states of the search by CPDAGs and define operations on these CPDAGs, which are special forms of the following: Insertion of a directed or undirected edge, removal of a directed or undirected edge, formation of a v-structure, reversal of an edge; all of them under the assumption that no directed cycle is formed. It is not hard to see that these operations – even without edge reversal – are complete: Any CPDAG can be transformed to every other CPDAG by first removing all the existing edges and then inserting the new ones. All regions of the search space are thus in principle reachable from every initial element. (This also holds for the specific versions of the rules that GES uses.) However, it is not that obvious how it can be ensured that each operation results in a valid CPDAG– in general, the result is a PDAG.

Chickering solves the problem by transforming the PDAG back to the space of CPDAGs as follows: A consistent extension of the PDAG is computed using the algorithm by Dor and Tarsi (see Section 5.4.1); then the resulting DAG is turned into a CPDAG on the basis of the characterisation 3 in Theorem 2.2. Since this procedure makes the steps of the search more complex, the search in the space of equivalence classes is considerably less efficient, despite its smaller size. In the journal version of his paper, Chickering thus proposed a local way to score the neighbouring graphs that involves recomputing only up to four of the terms in Equation (3.1). This avoids the transformation of the PDAG to an arbitrary extension as well computing all terms of the score for this DAG. (All extensions of the PDAG are Markov equivalent and thus, under the assumption of score equivalence, yield the same score.) However, note that the above transformation is still necessary for *obtaining* the new best scoring CPDAG in each step.

In his PhD thesis, Meek [184] proposed a very simple search algorithm in the space of equivalence classes that he conjectured to be correct. However, the proof was only given a few years later by Chickering [43], based on his transformational characterisation of Markov equivalence for DAGs (3 in Theorem 2.2). The algorithm is called *greedy equivalence*

search (GES). It searches the same space (of equivalence classes) as Chickering's earlier method, but consists of two phases with different notions of connectivity. In the first phase, the *forward* phase, a state $\mathcal{E}'$ in the search space can be reached from a state $\mathcal{E}$ if there is a DAG in $\mathcal{E}'$ that is obtained from a DAG in $\mathcal{E}$ by a single edge *addition*. Conversely, in the second phase, the *backward* phase, a state $\mathcal{E}'$ in the search space can be reached from a state $\mathcal{E}$ if there is a DAG in $\mathcal{E}'$ that is obtained from a DAG in $\mathcal{E}$ by a single edge *deletion*. Because of the construction of the operations, this algorithm is statistically consistent. In the first phase an equivalence class of DAGs is found that are Markovian to the distribution. (Since we have seen in Section 3.2.1 that the complete graph is Markovian to every distribution, it would be possible, but impractical, to start the second phase with a complete graph instead.) In the second phase, the algorithm moves to a class of DAGs that are faithful to the underlying distribution, while preserving the Markov property. The algorithm, albeit greedy, is therefore consistent in the large sample limit as long as the distribution is Markovian and faithful to a DAG over the observed variables, and for scoring functions that are score equivalent, consistent and decomposable (which many of the scoring functions used in practice are). This is a somewhat surprising result, but does not contradict the NP-completeness of the search problem. On the one hand, despite the simple search strategy, the time complexity of GES is not polynomially bounded because a state in the search space can have an exponential number of neighbours[8] [48]. On the other hand, the hardness result does not apply for methods that assume faithfulness.

Several improvements of the GES algorithm have been proposed. While, in the large sample limit, the GES algorithm finds the globally minimal model that is able to explain the data – the global maximum of the score – it does not (necessarily) in practice, even with large data sets. Chickering and Meek [47] therefore propose an extension called UGES, which allows forward and backward steps in both phases. They can show a local form of optimality for this algorithm under relaxed assumptions – for example, the distribution does not need to be faithful to a DAG anymore and hidden variables and selection bias can be taken into account. Nielsen, Kočka, and Peña [196] propose a version where the choice of a neighbouring state is not completely greedy. Instead of moving to the best scoring neighbouring state in each step, they also consider randomly moving to one of the neighbours that increase the score. Without loosing asymptotic optimality, the greedyness versus randomness trade-off can be controlled via a hyperparameter k, which is reflected in the algorithm's name. Several restarts of an instantiation with randomness can help finding a *global* optimum. Hauser and Bühlmann [123] adapt the algorithm to make use of data from different, possibly interventional data settings. Chickering [45] shows that the accuracy can be improved by an adaptation called SE-GES, which uses statistically efficient operators.

Other extensions aim at increasing the efficiency: Chickering and Meek [48] provide an algorithm named *selective greedy equivalence search* (S-GES), that is polynomial in the number of nodes (but exponential in other measures, such a the maximum in-degree). Nandy, Hauser, and Maathuis [193] restrict the set of edge additions in the forward phase without loosing the consistency, while Ramsey et al. [218] propose the *fast greedy equiva-*

[8] This is because the operators are not defined for pairs of nodes X and Y only, but additionally depend on a set of nodes of arbitrary size. For the INSERT operator, for example, this set is an arbitrary subset of the neighbours of Y that are not adjacent to X

lence search (FGES) and show its applicability to a million variables. Nine other modifications, which in part retain optimality, are given in [10].

Yet other search spaces have been considered, for example the space of variable orderings, that is to say permutations. This leads to many useful algorithms because it amounts to a search in a smaller space with a lower branching factor and elegantly ensures acylicity. In addition, for a given node ordering, the best scoring DAG can be efficiently constructed when the in-degree is bounded [55]. This is used, for example, in ordering-based search (OBS), [276]. The latter algorithm has been improved on by [232], leading to acyclic selection ordering-based search (ASOBS). A related approach using interventional data to determine the causal ordering is proposed by Shojaie et al. [246]. Recently, Solus, Wang, and Uhler [253] proposed a consistent algorithm applicable to hundreds of variables, based on the sparsest permutation (SP) algorithm by [220]. A related, albeit not permutation based approach is pressured by Aragam and Zhou [13]. The resulting algorithm does not assume faithfulness and is very efficient in the high-dimensional setting.

3.5.3
Other Causal Structure Leaning Algorithms

In the previous sections, we have already encountered a few algorithms that blend the two design principles for causal structure learning algorithms – based on independences or based on a whole distribution. In this section, more algorithms are presented that are neither constraint-based in the style of the PC algorithm, nor search a space of graphs or variable orderings. Some of them are hybrid and combine the two strategies, others use a completely different approach, some could even be assigned to one of the two above-mentioned categories, but are very different from related methods or make very distinctive assumptions.

A very early approach to graphical model learning that predates the first *causal* graphical models was limited to structures that have the from of a tree, that is to say a DAG with an acyclic skeleton [49]. This approach has been generalised, for example to graphs that are similar to trees [152, 38, 88]. The similarity is usually measured in terms of the graph theoretic notion of *treewidth* – the lower the tree width of a graph, the more similar to a tree the graph is.

From a complexity theoretic perspective, an interesting approach to solving NP-complete problems is fixed parameter tractability [80]. The idea is to construct exponential time algorithms that, however, are exponential only in some parameters of the input, not the size of the input as a whole. This makes sense for parameters that are small in practice, for example the maximum degree of a graph or its treewidth. Both parameters are regarded in [197] to characterise the problem's parameterised complexity. Alongside a number of negative results, the authors find that the problem is solvable in linear time when the so-called super-structure has both bounded degree and treewidth.

Typical hybrid methods can be characterised as follows [48]: First, the search space is pruned, eliminating a subspace of elements that can be ruled out on the basis of independence constraints. Then, the search is conducted in the remaining subspace, which hopefully has become much smaller. One way to restrict the search space significantly is to fix the skeleton by learning it in a constrained-based fashion before the edge directions

are determined in a score-based fashion, as for example in [267]. In the first hybrid algorithm, the CB algorithm [250], the first two phases of the SGS algorithm are interweaved with the K2 algorithm and executed iteratively for increasing order of the conditional independence tests.

Another possibility to indirectly learn the skeleton in a constraint-based fashion is considering parents sets or Markov blankets for all nodes. Friedman, Nachman, and Pe'er [100] suggest the *sparse candidate* algorithm. The idea is to first restrict the parents of each node to a *candidate* set of fixed size and then determine the best scoring subset of parents for each node, such that the resulting DAG is valid.

The well-known min-max hill climbing (MMHC) algorithm [287] can be seen as an instantiation of the sparse candidate algorithm, but employs a sound method to generate the restricted sets of possible parents and children. It further does not limit the size of these sets globally. This first phase is implemented via an improved version of the max-min parents and children algorithm [283], and works as follows.

It is based on a measure of the association of variables, for example the negative p-value of a conditional independence test. This measure is used in the so-called max-min heuristic which, for a node X, determines the node Y which maximises the minimum association given a subset of a set C_X, that is

$$\text{max-min-heuristic}(X, C_X) = \max_{Y \in \mathcal{V} \setminus (\{X\} \cup C_X)} \min_{Z \subseteq C_X} (\text{association}(X, Y \mid Z)).$$

The Markov blanket of each node X is computed by iteratively augmenting the set of candidate neighbours C_X by $\text{max-min-heuristic}(X, C_X)$. Subsequently, the spouses are removed from the C_X. To identify whether a node $Y \in C_X$ is a spouse, one can check it there is a node Z that separates X and Y, because as we know from Theorem 3.4, only neighbours cannot be separated by a subset of the neighbours of X_i.

The second phase of MMHC is the score-based search, for which a hill climbing approach is used. The available operations are adding, deleting and reversing edges. However, the insertion of edges is restricted to the skeleton implicitly determined in phase one: The edge $X \to Y$ can only be inserted if Y is in the set C_X of candidate parents and children of X. The authors include an extensive evaluation of structure learning algorithms, showing the practical applicability of MMHC.

A very different hybrid approach is taken for the essential graph search (EGS) algorithm proposed by Dash and Drużdżel [67]. The idea is to search the space of parameters of the PC algorithm. For each parameter setting, the result of the PC algorithm is extended to a DAG, and the best scoring DAG is searched for randomly. Since the original PC algorithm is order-dependent (as described below), the parameters consist of the ordering of the variables as well as the level of significance for the conditional independence tests.

Algorithms for More General Settings

There are also approaches that are constraint-based in that they describe the problem by constraints of some kind, but differ from the methods in Section 3.5.1 because they are not (solely) based on the constraints imposed by conditional independence and are sometimes combined with some form of optimisation.

For example, the constraints can be expressed in a more general form, like propositional logic [83, 281, 142, 280] or *(integer) linear programs*, (I)LPs for short, [146, 64, 16]. This

is possible because the ILP and SAT problems have a high expressiveness. Moreover, these problems are very general, and decades of optimisation have yielded very efficient algorithms and solvers for these more general constraint satisfaction problems, including the CPLEX LP-solver and the MINISAT solver [86]. Work on linear programming based solutions has been conducted by Cussens [64] and Bartlett and Cussens [16] (using CPLEX). Triantafillou, Tsamardinos, and Tollis [281] and Hyttinen et al. [142] propose methods directly based on SAT-solving (using MINISAT).

Hyttinen, Eberhardt, and Järvisalo [140] suggest a new SAT-based using *answer set programming (ASP)* that allows to handle conflicting information due to sampling errors and thus improves the accuracy in the presence of such errors. The idea is to weight the constraints and compute the solution that minimises the total weight of the constraints that are not satisfied. The weighting can be done with the aim of obtaining a model that is as simple as possible, in a Bayesian fashion [140], or according to p-values of statistical tests [280, 174].

Because such methods do not rest upon constraints like "there is a directed edge between X and Y", but instead on more abstract statements like "X is independent on Y conditional on Z", it is easier to express complex requirements and fewer assumptions are needed: It is easy to extend SAT-based methods to learn cyclic models and allow for latent confounders [139]. It has also been shown that ASP is robust and can resolve conflicts in the input. [140] By extending this approach, Forré and Mooij [95] provide a way to also deal with non-linear relationships between the variables and make use of interventional data, possibly from different data sets (see also the corresponding paragraph in the next Section).

The flexibility of the approach also concerns the output. If only certain features of the model are of interest, it is not necessary to compute the whole causal structure. This fact is exploited for example in [141] and [174]. All in all, the SAT-based methods is versatile and accurate. The problem is that it does not scale well. Hyttinen, Eberhardt, and Järvisalo [140] show the applicability to models over six variables, although in specific, very restricted domains about 70 variables can be handled [143].

There are other strategies for learning causal models with relative mild assumptions (see also the constraint-based methods in the last paragraph of Section 3.5.1). Using independent components analysis (ICA), Lacerda et al. [160] generalize the LiNGAMs to the cyclic case. Based on this, Mooij et al. [188] consider cyclic SCMs with possibly non-linear assignments with additive noise, but only in the bivariate case. Mooij and Heskes [187] propose a method to learn cyclic models from a mixture of observational and interventional data without assuming linearity, and show its practical usefulness on an "classical" immunological data set by Sachs et al. [231].

3.5.4 Special Settings

The following settings concern more practical issues regarding causal structure learning. These include cases where additional information is available, for example in terms of the variable ordering or interventional data. The following methods are concerned with using additional information for the largest benefit, or coping with missing data.

Another type of additional information is usually referred to as background knowledge. This means that some edges of the model are known in advance, or are known to be absent. This is treated specifically in [181, 128], but many of the above-mentioned algorithms, in particular the PC algorithm, can incorporate such information.

Time Series Data

We have seen that the problem of causal discovery is very difficult, not least because of the huge size of the search space. However, we have also seen that the problem becomes much easier when the ordering of the variables is known [276, 247], [214, Section 7.2.4]. If there are no hidden variables, it is even possible to relax the assumption of faithfulness and still retain identifiability [214]. In fact, many early causal discovery algorithms like the Wermuth-Lauritzen [303, 305] (see Section 3.5.1). and the K2 algorithm [55, see Section 3.5.1] assumed a given topological ordering of the variables. Although this assumption is often invalid in practice, it can be justified if the variables have a clear *temporal* ordering. For *time-series* (or *longitudinal*) data, such an ordering is given, making the problem in general more tractable. However, the causal ordering might not be a strict total order: There might be variables whose time order cannot be distinguished, for example because the measurement was carried out (nearly) simultaneously. Moreover, measurements can sometimes have time scales that are not in accordance with the time scale of the underlying causal processes [143]. On top of that, we cannot always assume causal sufficiency. Another problem stems from the fact that often only one sample of the time series is available.

Causal modelling with time series data is a very early branch of causality [307, 115]. Many causal structure learning methods have been devised particularly for time series data [143]. On the other hand, some practical approaches are based on the PC algorithm (for example [21]). When latent confounding is involved, FCI-based methods are common [50, 89, 176]. A method by Brodersen et al. [29] is efficient enough to be used on data with 500 time points and 10 variables. So called *dynamic Bayesian networks* are a hybridisation of Bayesian networks and Markov processes specifically devised for time series data [191].

Eichler [87] gives a good introduction to causal concepts in the context of time series data. The topic is also discussed in detail in [130, Chapter19-22].

Heterogeneous Data

If multiple data sets are available, they can often not simply be merged. Nonetheless, since a large sample size is often a very important prerequisite for accurate causal inference, it is very desirable to incorporate all the data into the structure learning process simultaneously. The main issues that have been considered in the literature are combining data sets with different interventional settings (possibly combined with observational data) or data sets over different (but overlapping) sets of variables. This can be particularly useful when only small samples of interventional data can be obtained, but observational data are relatively easy to acquire [246].

Like causal orderings, interventions can provide worthy additional information about the causal structure. As mentioned above, they can help to obtain a more fine-grained structure of Markov equivalence classes and thus to learn the underlying model more accurately.

Generally speaking, there are two main approaches to working with data from multiple data sets [189]: The first option is to derive constraints from the individual data sets separately and then try to find a model that is compatible with all or as many as possible of the constraints. This approach is taken in [278, 139, 140, 280, 95] Many such methods use a SAT-based approach.

The second approach is to merge the data sets and construct a causal model from the pooled data, as in [56, 231, 123, 187, 213, 189].

Some other methods [83] solve the problem in a specific way. For example, Shojaie et al. [246] use interventional data to infer a causal ordering of the variables and then compute a consensus network on the basis of observational data.

Concerning the second problem, namely learning from multiple overlapping data sets, Triantafillou, Tsamardinos, and Tollis [281] argue that the lack of methods for this problem is due to the "prevalence of association (correlation) as the conceptual cornerstone of data analysis", while *causal* methods using graphical models offer an elegant option for such analyses. The first asymptotically correct method for learning causal structure from a collection of data sets with overlapping variable sets called ION was presented by Tillman, Danks, and Glymour [277]. An improvement that accounts for conflicting information was given by Tillman and Spirtes [278]. Triantafillou, Tsamardinos, and Tollis [281] proposed a faster but less general method based on SAT solving that is able to compute a network also when the set difference between individual variable sets is larger than three. They extend their algorithm to predict correlations between variables that never occur in the same data set [288], and to handle conflicts, resulting in a method called COmbINE [280]. Note that SAT-based methods do not only offer a way to combine constraints expressed through multiple data sets, this approach also allows to combine data sets with overlapping variables; see also [142].

Often, data is also heterogenous with respect to the types of the variables, that is a data set might comprise continuous and discrete variables, An ad-hoc solution to work with such data is to discretise the continuous data, as done, for example, in [282, Section 5.2], but sophisticated methods also exist, including [62].

Missing Data

We have already considered causally insufficient settings, where not all the variables are measured. One specific case where a hidden variable can distort causal reasoning is if this variable determines whether a sample is included in the data – it is then called a selection variable. A similar situation arises when a hidden variable determines whether individual data *entries* are missing. If there is such a hidden cause of missing values, just deleting the concerned cases – a practice usually referred to as listwise deletion – would bias the conclusions, just like conditioning on a selection variable.

Rubin and Little [230, 167] propose a systematics of missing data according to the processes that caused their "missingness":

- MCAR stands for *missing completely at random*. If the missing values occur completely at random, the (unobserved) processes which determine whether a value is missing or not are independent of one another.
- MAR stands for *missing at random*. This describes a setting where the missingness of variables is determined by observable variables.

▷ MNAR stands for *missing not at random* (also abbreviated as NMAR). In this case, the causes for missing values are unobserved but not independent.

For a graphical depiction of the settings see [238, Figure 2]. If in the first two cases there are any non-random patterns of missingness, these can be found using the given data, which makes them more tractable ([63]). In these cases it is thus valid to discard missing data. This is not possible if the data are not missing at random. On top of that, deleting complete observations only because one value is missing, is very sample inefficient. It is just desirable to keep such samples and work with incomplete data.

There are two general strategies for coping with missing data. The first one is to guess and then gradually improve the values of the missing data. The most famous algorithm of this kind is the structural EM algorithm [98], see also [195, p. 9.1.3], which is based on the canonic expectation-maximisation algorithm [76] and works is the MAR setting. It consists of an expectation step, where completed data are estimated, and a maximisation step, where the optimal network is computed for the current completed data. In the next round, the missing data are estimated more precisely given this network and so on. Note that complete data are given in the maximisation step, which is why any general structure learning algorithm can be used for this.

A recent method of this type is given in [2]. In this method, the circumstances that cause the missingness are modelled explicitly as additional (auxiliary) nodes in the causal network, yielding the same complexity of the algorithm as for complete data. On top of that, the method is shown to work well, even when data are not missing at random. Foraita et al. [94] propose a new possibility to reconstruct missing values and compare it to the EM-based approach.

The second strategy is to devise algorithms or scores that work directly with incomplete data. For example, Beal and Ghahramani [17] use an adapted version of a likelihood, which they prove to be a lower bound to the true likelihood. Another approach of this kind is given in [90].

Cui, Groot, and Heskes [63] consider the accuracy of an extension of the PC algorithm, when data are missing completely at random (MCAR), and propose an extension for the MAR setting. Recently, Tu et al. [289] extended a method by Strobl, Visweswaran, and Spirtes [272], that is devised particularly for constraint-based algorithms. Instead of discarding complete samples with missing data, it involves deleting samples with missing data in the individual conditional independence tests – but only for variables that are regarded in the current test. In this way, known as *test-wise deletion*, the effective sample size can be significantly improved.

An evaluation of general causal structure learning algorithms on data with missing values (5% or 10%) and other types of noise has been conducted by Constantinou et al. [54]. A profound introduction to the topic can be found in [158, Chapter 19].

3.6
Evaluation of Causal Structure Learning Algorithms

3.6.1
Measures for Comparing Models and Data

Score-based causal structure learning algorithms rely on a measure of the fit between a candidate causal structure and given data, the *scoring function*, *score metric*, or simply *score*. Although score-based learning is not a main topic in this thesis, in this section some of the most common scoring functions are briefly introduced. These can be useful also for other purposes than score-based learning, for instance parameter tuning. This will be further discussed in Section 5.3.

We have already encountered two characteristics of most of the scoring functions used today: *decomposability*, which allows efficient dynamic computation, and *score equivalence*, which ensures that indistinguishable DAGs are assigned the same score (see, for example, [128]).

Otherwise, the score

1. should describe how well the graph can describe the data, as well as
2. reward simplicity, in order to prevent overfitting.

The latter claim follows Occam's Razor principle, which suggests that the simplest model that can explain the data is generally the most appropriate.

If this was not required, the score could easily be maximised by choosing the complete graph, which can best reproduce arbitrary distributions. These two objectives are subsumed under asymptotic consistency: A scoring function is consistent if the true underlying structure has the maximal score, while all structures that do not encode the same independences as the given distribution have a lower score. Chickering [44, Definition 6] defines the notion of local consistency, which more directly reflects the two requirements: The score should increase when an edge is added which remedies an independence that does not hold in the data but was previously expressed in the graph. If an edge is added that does not remedy such an independence, the score should decrease. A scoring function with these properties is then said to be locally consistent.

The scoring functions presented in the end of this section are consistent, decomposable and score equivalent. For a more comprehensive introduction to scoring functions for Bayesian networks see, for example, [267, 158, 35, 168]. Empirical studies of different scoring functions can be found in [60, 73, 35].

Most of the scores used in the literature belong to one of three groups. The first group consists of measures that regard a Bayesian network from a statistical perspective, as a parameterised class of probability measures. The class is mainly represented by two scores, the Bayesian information criterium (BIC), which is introduced in detail below, and Akaike's information criterion (AIC), which traces back to Akaike [4] and is first considered in the context of Bayesian networks in Bozdogan [27]. It differs from the BIC only in that it has a smaller penalty term and thus favours denser structures (see below).

The second group, information-theoretic scores, are based on measures like entropy or mutual information [49, 131, 25, 161, 73]. This group is generally underrepresented in

the recent literature, except for the MDL score, which, in the case of Bayesian networks is an information-theoretic equivalent of the very common BIC score.

The third group is constituted by Bayesian scores. These maximise the posterior probability of the causal structure given the data. As explained in detail below, this amounts to maximising the product of a prior and a likelihood, or the sum of a logarithmic prior and logarithmic likelihood. Bayesian methods differ above all in the choice of the prior.

Before we consider this group in detail let us first think about why the maximum likelihood approach that we took for the parameters does not work directly for structure learning. First of all, the maximum likelihood approach seems reasonable: Why not search for a model that is likely to produce the given data? On top of that, this approach has two information-theoretic properties that are very much in accordance with our goals [158, Section 18.3.1.2]: On the one hand, it is larger for variables with a higher dependence on its parents. In this case, the dependence is measured by the information-theoretic notion of *mutual information*. Another interpretation is that the maximum likelihood measures the extent to which the structure satisfies the local Markov property (see Definition 2), since it decreases with increasing mutual information of each variable and its non-descendents, given the variable's parents.

However, the crucial drawback of the maximum likelihood score is that it never decreases when an edge is added to the network, and thus lacks the second if the properties defined in the beginning of this section. In terms of the mutual information of a variable this can be explained as follows: Adding an edge, and thus another parent to a variable can only increase the mutual information of this variable and its parents. We never lose information by adding a parent. The result is overfitting. A structure selected by maximum likelihood would always be fully connected (unless in the rare case that two variables are completely uncorrelated in the data). Maximum likelihood can thus only be applied fruitfully when there are additional constraints that limit the complexity of the model, for example when the node degree is fixed.

Bayesian Scores

The obvious Bayesian approach to remedy this insufficiency of the maximum likelihood is to maximise a posterior instead. As we have seen above, the posterior can be computed from the likelihood by multiplying with a prior.

This has two advantages. First of all, by choosing a lower prior for more complex networks, we can counteract overfitting. Nevertheless, the prior has only a small effect in practice, so this is not the whole secret.

There is also a difference in the role of the likelihood. In *maximum* likelihood estimation, the graph with the highest likelihood given its *optimal* parameters is preferred. Bayesian scores, on the other hand, depend on the *marginal* likelihood, which can be seen as the expected likelihood over all choices of the parameters. [267, 158]

Bayesian scores have two convenient properties that the maximum likelihood score lacks: They prefer simpler structures, but take the sample size into consideration and can learn more complex networks when indicated by a large sample. They thus provide a good trade-off between complexity and fit to the data. This behaviour is approximated by the BIC score (see the next Section).

Buntine [32] first proposed a Bayesian score for discrete Bayesian networks, using a Dirichlet prior. This is a natural choice because the Dirichlet distribution is a *conjugate*

prior for the multinomial distribution, implying that the resulting posterior is multinomially distributed if the likelihood is. Buntine's results were generalised by Spiegelhalter et al. [255]. [128] showed that under certain reasonable assumptions like score equivalence, a certain form of Dirichlet prior is actually the only possible choice for multinomial distributions, and dub the resulting score the *BDe* score. Similarly, [107] establish the *BGe* score for Gaussian distributions using a Normal-Wishart prior. More recently, [236] revisited the assumptions made to obtain the BDe score. Under slightly different assumptions, he derived the *BDs* score-based on a marginal uniform (MU) prior. However, the BDe score is still most common in score-based algorithms when the data are discrete. [282]

The BIC-Score

In 1976, Schwarz [235] defined the Bayesian information criterion for estimating the dimension if a model, that is, how many parameters a model should have to avoid overfitting the data. The idea was to approximate the value that a maximum a priori approach would yield, and indeed, for large samples, the BIC score converges to the logarithm of the BDe score [282].

The BIC score constitutes a penalised log-likelihood and is defined as follows:

$$\text{BIC}(\mathcal{G}, D) = \mathcal{LL}(\mathcal{G}, D) - 1/2 \cdot \log(s) \cdot |\mathcal{E}|,$$

where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the DAG, D denotes the data, which consists of s samples and $\mathcal{LL}$ refers to the log-likelihood function. For the computation of the log-likelihood, the optimal parameters of the model are used.

As indicated above, the BIC score is an adaptation of Akaike's information criterion. AIC penalises the likelihood of a model only with the number of its parameters, in our case $|\mathcal{E}|$. In contrast, the penalty term $\frac{1}{2} \cdot \log(s) \cdot |\mathcal{E}|$ is larger for $s \gtrapprox 7.4$, that is to say for practically all realistic sample sizes, and therefore more strictly prevents overfitting.

As all common scores for continuous data, the properties of the BIC score depend on the multivariate normality of the data. However, they often yield good although this assumption is rather unrealistic in practice [282].

The BIC score is usually the first choice in causal structure learning with continuous data and yields good results [168, 241]. It also has the advantage of being efficiently evaluable. Modifications of the BIC score have been used to learn Bayesian networks, including the *risk inflation criterion* [117] and *extended Bayesian information criteria* with a hyperparameter γ, where $\gamma = 0$ yields the ordinary BIC [39, 97].

3.6.2 Measures for Comparing Causal Structures

Especially for the evaluation of learned causal structures, it is often necessary to measure the degree of conformance, or the closeness, of two structures. For synthetic data that have been generated according to a predefined or randomly generated DAG, one can measure the accuracy of the learning algorithm in terms of the similarity of the output and the true causal structure.

The most common measure for this similarity is the *structural Hamming distance* (SHD). This measure is first formally defined by [287], but is based on an idea by Acid and de

Campos [1]. The SHD is defined as the number of operations required to convert one graph into the other, where available operations are edge insertions, deletions and reversals. Perrier, Imoto, and Miyano [211] propose to penalise the latter only half, because they constitute less grave errors.

In this thesis, the SHD is defined also with respect to conflict edges, which will be formally defined in Section 5.2.1.

Definition 12 *(SHD).*
The structural Hamming distance (SHD) for the partially directed graphs $\mathcal{G}$ and $\mathcal{G}'$ over the same set of nodes $\mathcal{V}$ is a measure for the number of the edges that are not common in both graphs. It is defined as follows as a sum over all pairs of nodes:

$$\mathrm{SHD}(\mathcal{G}, \mathcal{G}') = \sum_{(X_i, X_j) \in \mathcal{V}^2} \mathrm{SHD}_{\mathrm{ij}}(\mathcal{G}, \mathcal{G}'), \tag{3.2}$$

where

$$\mathrm{SHD}_{\mathrm{ij}}(\mathcal{G}, \mathcal{G}') = \begin{cases} 1 & \text{if } X_i \text{ and } X_j \text{ are adjacent in } \mathcal{G} \text{ but not in } \mathcal{G}' \text{ or vice versa,} \\ 1 & \text{if } X_i - X_j \text{ in } \mathcal{G} \text{ but } X_i \to X_j \text{ or } X_i \leftarrow X_j \text{ in } \mathcal{G}' \text{ or vice versa,} \\ 1 & \text{if } X_i \to X_j \text{ in } \mathcal{G} \text{ but } X_i \leftarrow X_j \text{ in } \mathcal{G} \text{ or vice versa,} \\ 1 & \text{if } X_i \leftrightsquigarrow X_j \text{ in } \mathcal{G} \text{ or in } \mathcal{G}' \text{ (or in both),} \\ 0 & \text{if } X_i \to X_j \text{ in } \mathcal{G} \text{ and } X_i \to X_j \text{ in } \mathcal{G}' \text{ or} \\ & \quad X_i \leftarrow X_j \text{ in } \mathcal{G} \text{ and } X_i \leftarrow X_j \text{ in } \mathcal{G}'. \end{cases}$$

De Jongh and Drużdżel [75] remark that the SHD can either be applied to DAGs or CPDAGs and point out that the latter is to be preferred. Given two DAGs, the corresponding CPDAGs can be uniquely identified, as we have seen in Section 2.4.2. Comparing the corresponding CPDAGs has the advantage that edges whose direction is indistinguishable given the data will not be penalised by the SHD. (This is in accordance with the measure by Acid and de Campos [1]). Along these lines, we will also compare to the pattern of the true DAG when the result of an algorithm is a pattern in Section 4.3.3. Constantinou et al. [54] object that comparing CPDAGs instead of DAGs overestimates the accuracy of the learning algorithm. They define an adaptation of the SHD that can be applied to compare DAGs and MAGs (Table 5).

De Jongh and Drużdżel [75] recommend not to rely on only one measure that aggregates information. Accordingly, we will regard other structural measures than the SHD as well.

Kalisch and Bühlmann [148] complement the SHD with measures that completely disregard the edge directions and treat the discovery of the edges as a classification problem (see also [..., 13, 105]). For this, it is a assumed that one of the graphs is the true graph and the other one is considered to be the estimated graph. Then a very general approach is taken that can be found in the machine learning literature, e. g. [279, 144]. A *confusion matrix* is computed, which contains the following quantities:

TP – the number of true positives, that is, edges that are in both graphs;
FP – the number of false positives, that is, edges that are in the estimated but not in the true graph;
FN – the number of false negatives, that is, edges that are missing in the estimated graph but exist in the true graph; and

TN – the number of true negatives, that is, pairs of nodes that are not adjacent in either graph.

These values can be combined to obtain the number of edges in the true graph

$$\text{EDGES} = \text{TP} + \text{FN}$$

and the number of "gaps" in the true graph

$$\text{GAPS} = \text{TN} + \text{FP}\,.$$

To denote the number of gaps and edges in the estimated graph, we use the terms positives and negatives:

$$\text{POS} = \text{TP} + \text{FP}$$

and

$$\text{NEG} = \text{TN} + \text{FN}\,.$$

Note that all these quantities depend on which of the graphs is the true graph and thus, also the following derived measures are not symmetric. All such measures assume values between 0 and 1. Although many other measures are regarded in other applications, we will only use the ones used in [148]. The false positive rate FPR, sometimes called *fall-out* or *false alarm ratio*, is the quotient of misplaced edges and true gaps:

$$\text{FPR} = \frac{\text{FP}}{\text{GAPS}} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \tag{3.3}$$

The true positive rate TPR, also known as *sensitivity* or *recall*, is defined as the ratio of the number of correctly identified edges to the number of edges in the true graph:

$$\text{TPR} = \frac{\text{TP}}{\text{EDGES}} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3.4}$$

The inverse measure, $1 - \text{TPR}$, is commonly referred to as the false negative rate. We will also regard the true detection rate, *positive predictive value* or *precision*, the number of correctly identified edges relative to the number of edges in the estimated graph:

$$\text{TDR} = \frac{\text{TP}}{\text{POS}} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{3.5}$$

Constantinou et al. [54] consider two other measures based on the confusion matrix:

$$\text{F1} = 2 \cdot \frac{\text{TPR} \cdot \text{TDR}}{\text{TPR} + \text{TDR}} \tag{3.6}$$

and the balanced scoring function

$$\text{BSF} = \frac{1}{2} \cdot \left(\frac{\text{TP}}{\text{EDGES}} + \frac{\text{TN}}{\text{GAPS}} - \frac{\text{FP}}{\text{GAPS}} - \frac{\text{FN}}{\text{EDGES}}\right).$$

They find that these two quantities often are largely in agreement. This is why we only consider F1 in the following. There are other, related measures used in the literature (but

not in the remainder of this thesis). For example, Gu and Zhou [117] report the *Jaccard index*, which is defined as

$$\text{JI} = \frac{\text{R}}{\text{EDGES} + \text{POS} - \text{R}},$$

where R is the number of edges that the CPDAGs corresponding to the two DAGs have in common (*considering* edge direction).

Peters and Bühlmann [212] propose a measure called structural intervention distance SID. As the name suggests, it compares the graphs on the basis of their ability to predict interventions.

In early papers, it was common to use different measures. Instead of the graphs themselves, their induced distributions were often compared, for example via so-called cross entropy [127, 128] or Kullback-Leibler divergence [1]. Other measures stem from decision theory, above all *expected utility* [127]. An introduction to utilities and decisions can be found in [158, Chapter 22].

3.6.3 Gold Standard Structures and Random DAGs

For the evaluation of causal structure learning algorithms it is useful to know the true DAG. By and large, there are two possibilities for this: either one uses data for which the underlying causal structure is known, at least in terms of a consensus network, or one generates an arbitrary true graph and then samples data from it.

The first approach has the advantage that it is closer to the practical conditions, a bit like an *in vivo* study. The main concern with artificially generated graphs is that they might lack characteristics typical for real world networks. Some interesting properties of this kind have been identified over the last decades [302, 15, 147, 6, 3], but there might be more that we have not been able to capture yet. However, only few consensus networks are available and they might have other peculiar properties, for example regarding the type of data. What is more, it is not generally possible to adjust the parameters of the network and data to perform a more thorough evaluation.

The second approach to evaluating learned causal structures against a true graph is to start with the graph instead of the data. In principle, it is possible to use an arbitrary DAG as the gold standard, but to avoid bias, a collection of graphs with equal parameters is often randomly generated and later averaged over. This procedure is also applied in this thesis, with varying numbers of replications for different DAGs. This number denoted by REP.

To generate random DAGs, we proceed like Kalisch and Bühlmann [148] and Colombo and Maathuis [52]: For a DAG we can order the nodes topologically, such that we have $i < j$ in every edge $X_i \to X_j$. This in turn entails that the adjacency matrix A has only zeros on the main diagonal and below it. An elegant way to generate a random DAG is thus to fill only the upper triangle of A. To obtain a DAG with expected degree d, we can draw each entry of this triangle independently from a Bernoulli distribution with the parameter $d/n-1$. That is, an edge $X_i \to X_j$ is established between two nodes X_i and X_j with $i < j$ with probability $d/n-1$.

Note that in this way, isolated nodes might occur. Although in practical applications such situations could be ruled out by removing nodes that are not correlated to any of the

other nodes, there are methods that expect not only sparseness du to a low overall degree, but also due to (nearly) isolated nodes or components [117]. Aragam and Zhou [13] state that "many realistic DAG models tend to be sparse in one of these two senses".

To parameterise the linear model, the edges are now weighted, yielding the weighted adjacency matrix W. The entries w_{ij} that correspond to non-existing edges are set to zero. The other edge weights w_{ij} are then drawn uniformly from the interval $[b, 1]$. Theoretically, the lower bound $b = 0$ could be chosen, but for the dependences and independences to be distinguishable statistically and to avoid cases of near-unfaithfulness we always use $b = 0.1$ in this thesis. The whole process is implemented in the function `randomDAG` in the R package `pcalg`.

Given a DAG, the data are generated via ancestral sampling, as described in Section 2.5. For this, we use a linear Gaussian SCM with assignments $X_s = \varepsilon_i$ for all sources X_s and for the other nodes

$$X_i \leftarrow \sum_{j=1}^{i-1} w_{ij} \cdot X_j + \varepsilon_i.$$

The Gaussian error terms ε_i are drawn independently from a normal distribution $\mathcal{N}(0, 1)$ in the sampling process. This sampling procedure is repeated independently s times to generate the data on with which the causal structure learning algorithm learns the *estimated* graph. This graph can then be compared with the true graph using measures introduced in the previous section.

The advantage of this approach is that it facilitates comprehensive parameter studies: The number of nodes in the graph, the type of graph, its density, the type of noise, the functions of the SCM and more can be varied to see how this affects the outcome. Such analyses using the PC algorithm are presented in the next chapter.

4

Theoretical and Empirical Analysis of the pc Algorithm

After an introduction to the foundations of causal modelling and an overview of methods and circumstances for causal structure learning, we will now focus on the pc algorithm. In this chapter, the pc algorithm is presented in detail and its properties are analyses – both theoretically and empirically.

4.1 Description of the pc Algorithm

The pc algorithm is a "prototypical" constraint based structure learning algorithm [287]. For a discussion of its historical and theoretical background see Section 3.5.1, in particular Theorems 3.2 and 3.4.

A description of the pc algorithm in pseudocode is given in Algorithm 1. It consists of three principal phases: It first estimates the so-called skeleton, the undirected graph that describes direct correlations in the data (lines 1 - 13). The skeleton is computed by starting with a complete graph – a graph where every pair of nodes is connected by an edge. Then, whenever an dependence between a pair of nodes (X, Y) can be explained by a set S_{XY} of other nodes that mediate the effect, the direct edge between X and Y is removed. By proceeding in this way for increasing cardinalities of the set S_{XY}, this procedure is empirically fast, although it theoretically has an exponential running time.

To estimate whether an effect is mediated by other variables, the pc algorithm makes use of conditional independence tests. Conditional independence is assumed (and an edge removed) when the test does not reject the hypothesis that the two variables are conditionally independent given some subset S of the other variables at a significance level of α. That means that the edge is kept when the hypothesis is rejected. The parameter α must be predefined and controls the sparsity of the output graph. The tuning of this parameter is crucial to the method and described in Section 5.3.

Phases 2 and 3 of the pc algorithm consist in directing as many as possible of the edges in the skeleton. The emergence of any directed edges in the graph relies on the observation that so-called v-structures can be distinguished from other subgraphs on the basis of the pattern of conditional independences that they produce, as described in Section 2.2.2. The identification of the v-structures constitutes the second phase of the algorithm (lines 15 - 19).

Data: The Algorithm depends on the results of the independence queries $X \perp\!\!\!\perp Y \mid S$ in line 6

1 Start with a complete undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
2 $k = 0$
3 **while** *there exists a node* $X \in \mathcal{V}$ *with* $k < |\mathcal{N}(X)|$ **do**
4 **for** *every adjacent pair of nodes* $(X, Y) \in \mathcal{V}^2$ **do**
5 **for** *every subset* $S \subseteq \mathcal{N}(X) \setminus \{Y\}$ *or* $S \subseteq \mathcal{N}(Y) \setminus \{X\}$ *with* $|S| = k$ **do**
6 **if** $X \perp\!\!\!\perp Y \mid S$ **then**
7 remove the edge $X - Y$ from $\mathcal{E}$
8 $S_{X,Y} = S$
9 **end**
10 **end**
11 **end**
12 $k = k + 1$
13 **end**
14 $\mathcal{G}' = \mathcal{G}$ ▷ *$\mathcal{G}'$ is the estimated skeleton*
15 **for** *every induced subgraph* $X - Y - Z$ *in* $\mathcal{G}'$ **do**
 ▷ *no edge between X and Z!*
16 **if** $Y \notin S_{XZ}$ **then**
17 orient the edge between X and Y and the edge between Y and Z towards Y in $\mathcal{G}$, potentially creating a bidirected edge
18 **end**
19 **end**
20 Mark every bidirected edge as a conflict edge
21 **while** *possible* **do**
22 Apply the following rules in $\mathcal{G}$ (if the orientation of an edge is ambiguous, mark it as a conflict edge
  1. If X and Z are not adjacent in $X \to Y - Z$ then orient as $Y \to Z$
  2. If $X \to Y \to Z$ and $X - Z$ then orient as $X \to Z$
  3. If X and Z are not adjacent, but $X - W - Z$ and $X \to Y \leftarrow Z$ and $W - Y$ then orient as $W \to Y$
23 **end**
24 **return** $\mathcal{G}$

Algorithm 1: The PC algorithm. The input of the algorithm are results of conditional independence queries (see line 6). In the oracle version of the algorithm they correspond to the (in)dependences in the true graph (determined by d-separation), in the sample version they are assessed from data by a conditional independence test.

After all edges that are part of v-structures have been directed, it is usually possible to direct other edges based on the established directed edges (lines 21 - 23). This is accomplished using a set of rules that have become known as Meek's rules [182], and are described in Section 2.4.2 (see in particular Figure 2.2).

Since a unique DAG is generally not identifiable from data, the result of the PC algorithm represents a *set* of Markov equivalent DAGs. If the provided conditional independence information is perfect, this set is a Markov equivalence class (see Definition 7) that can be represented by a CPDAG. This succinct graphical representation of the set of

DAGs that are consistent with the given independences can be used to make the following analysis more efficient than it would be to analyse all the alternative DAGs separately.

Terminology

As is formally defined in Section 2.2.3, a separating set S_{XY} is a set that renders two nodes X and Y independent when conditioned on, that is to say it separates them.

In the first phase of the pc algorithm, the *skeleton phase* (lines 1 - 13), sets of increasing cardinality k are considered for all pairs of adjacent nodes X and Y in order to find a separating set S_{XY}. We will refer such a stage as round k of the algorithm in the following. The first round, in which marginal independences are assessed, is thus round 0.

The output of the pc algorithm is denoted by $\hat{\mathcal{G}}$ and referred to as the (pc)-estimated graph. If the true graph from which the data were sampled is known, it is denoted by $\mathcal{G}^*$.

4.1.1 The Sample Version

In practical applications, conditional independences (line 6 of Algorithm 1) have to be assessed from data using conditional independence tests (see Section 2.1.4). Since the data represent a sample of an abstract population, the resulting algorithm is called the sample version of the pc algorithm. Conversely, the oracle version, in which all conditional independence test yield the correct result, is sometimes referred to as the population version (e.g., in [148]).

We consider independence to be the null hypothesis of the conditional independence test. If we can reject this hypothesis for nodes X and Y and a set S_{XY} at a certain level of significance α, that is to say when we have found significant evidence that X and Y are dependent given S_{XY}, only then do we keep the edge between them.

The level of significance α corresponds to the probability of committing a type I error, the probability of mistakenly rejecting the null hypothesis and keeping incorrect edge. It is a parameter of the pc algorithm that is chosen by the user. However, it is not necessarily beneficial to choose α as small as possible; it should thus rather be seen as a tuning parameter than an overall level of significance.

This is because, in this context, the inherently asymmetric statistical test is applied to judge a symmetric query: Is there an edge or not? In a typical application where a deviation or anomaly is to be found, the assignment of null and alternative hypothesis is obvious – the anomaly constitutes the alternative to the ordinary. In such a case, the result – that is, the detection of the divergence from the null hypothesis – is stronger if it is found at a lower level of significance α. On the other hand, the lower α, the higher are the chances to overlook a deviation. Such an error is referred to as the type II error and its probability is usually denoted by β. In the anomaly-detection example β is less important than α. If we can show that we find a very significant anomaly, it does not matter that we might have missed it had it been a bit less significant.

The situation changes completely in the case of the pc algorithm. The lower α is chosen, the lower the chance to judge independent nodes dependent and thus keep superfluous edges. However, at the same time we increase β, the chance of erroneously removing edges that should have been kept. The aim is thus to find a value for the level of significance that yields a balance between α and β. This issue is treated in detail in Section 5.3.3.

Theoretically, it would be possible to choose *dependence* as the null hypothesis. This would also be more in accordance with the initial condition of the algorithm, where all edges are present and are then gradually removed. But more importantly, choosing *in*dependence as the null hypothesis is in accordance with our objective to learn faithful graphs (see Definition 5). We want to keep edges only if necessary, that is if we have found significant evidence for it – and this is what a statistical test assesses for the alternative hypothesis. Conversely, if a statistical test results in assuming the null hypothesis, this is not necessarily a result of strong evidence for the null hypothesis. Due to the asymmetry of the test, it should rather be seen as a failure to reject the hypothesis than as accepting it. The *power* of the test is the probability of rejecting the null hypothesis if it is false, which amounts to $1 - \beta$ and is dependent upon the choice of α. The power of the test is also determined by the nature of the statistical question, in particular by the sample size and assumptions about the data. In the case of conditional independence testing it further decreases with the size of the conditioning set.

To summarise, an edge is removed from the graph if and only if a set S is found for which the null hypothesis that X and Y are conditionally independent given S is not rejected at a significance level of α.

Because of how the subsets S are chosen in line 5 of Algorithm 1, this means that an edge $X - Y$ will persist in round k of the PC algorithm if and only if the null hypothesis $X \perp\!\!\!\perp Y \mid S$ is rejected at a significance level of α for all subsets S of size at most k of the neighbours of X and Y, respectively. The quantification ("for all") thus intensifies the bias towards sparser graphs.

Since we only regard continuous data that we assume to have a Gaussian distribution, we can measure conditional correlation in terms of partial correlation and thus assess conditional independence employing Fisher's Z-test, as defined in Equation (2.6).

The PC algorithm is correct in the oracle model, that is when all independence queries in line 6 of Algorithm 1 are correctly answered. [263, 148]. Kalisch and Bühlmann [148, Theorem 1 and 2] show that the PC algorithm is consistent also in the sample model provided that the graph is sparse. Strictly speaking, they require the degrees of all nodes to be bounded by the sample size and allow the number of nodes to grow polynomially in s.

4.1.2 Time Complexity

While the complexity for phases two and three is polynomial in the number of variables, the running time of the PC algorithm is dominated by the cost of the first phase, the computation of the skeleton. In this, the PC algorithm is significantly more efficient than the SGS and IC algorithms in practice. However, albeit not directly exponential in n, its running time is still exponential in the degree of the output graph $\hat{d} = \mathrm{d}(\hat{\mathcal{G}})$, which in the worst case can be as large as $n - 1$.

The theoretical running time of the PC algorithm (in the oracle model) is given by Spirtes et al. [263]:

$$2\binom{n}{2}\sum_{k=0}^{d}\binom{n-1}{k} \leq \frac{n^2(n-1)^{d-1}}{(d-1)!}. \tag{4.1}$$

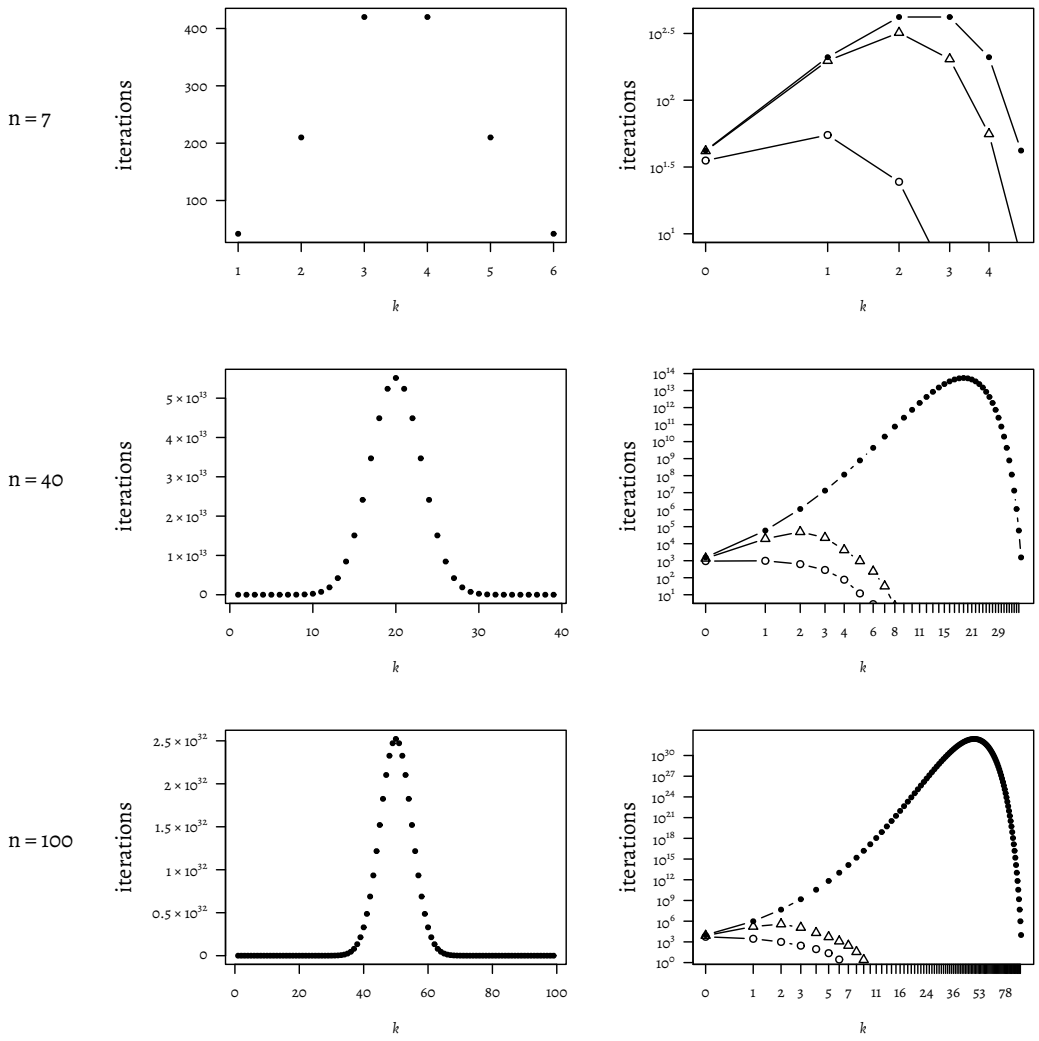


Figure 4.1: Theoretical upper bound and empirical number of iterations of the for-loop in line 5 of Algorithm 1 dependent on order k. *Left column*: Theoretical maximum of conditional independence tests of each order that would be executed by the pc algorithm when no edges would be removed. *Right column*: Comparison of the upper bound with the number of iterations actually executed, which are averages over 20 repetitions each, using $s = 10^6$ samples. Note that in the right column the scales of both axes is (pseudo)logarithmic. Filled dots mark theoretical upper bounds, triangles the empirical results for $d = 5$ (expected degree of the true DAG) and empty circles for $d = 2$. Although the data is of course discrete, the dots are connected by lines in the lower panels to improve the distinguishability of the three "curves".

Here, the term $2\binom{n}{2}$ corresponds to the for-loop beginning in line 4 of Algorithm 1 and $\sum_{k=0}^{d} \binom{n-1}{k}$ accounts for the different sets S considered in line 5 for all values of k in the outer

while-loop (line 3). The binomial coefficient $\binom{n-1}{k}$ arises because strictly, the neighbourhood of X (or Y) could comprise all the $n-1$ other nodes. But since one never needs to consider any of the nodes X and Y to find a separating set S_{XY}, we only need to consider $\binom{n-2}{k}$ possible sets S. Note that in the oracle model, the evaluation of a conditional independence has a constant cost. In practice, the complexity of the skeleton phase would need to be multiplied with the cost of the evaluation of a conditional independence test. That is where the sample size would typically enter into the evaluation.

From Equation (4.1) we can see that the running time is in $O(n^{d+1})$. That means that it is polynomial if d is fixed to a constant value. Note, however, that d is the degree of the *estimated* graph. In the sample case, where the PC algorithm does not necessarily output the true graph, it thus does not suffice for efficiency when the true graph is sparse. Also the significance level must be chosen sufficiently low in order for the PC algorithm to have a polynomial running time. Since it usually is unclear how low "sufficiently low" is, it is usually a better idea to algorithmically ensure the polynomial running time by bounding the maximal value of k that is considered. Such an algorithmic modification also has the advantage that high order tests that have relatively low power, especially for multinomial distributions, and thus a high risk of producing wrong results are avoided. Algorithms based on this idea are presented in [33, 254, 310]. In the implementation of the PC algorithm provided by the R package `pcalg`, which was used for all empirical analyses presented in this thesis, an upper bound for k can be passed to in the function `pc` as the argument `m.max`.

Spirtes et al. [263] remark that even the left-hand side of Equation (4.1) is only a loose upper bound. In particular, it is rare for the neighbourhoods of one of the nodes X and Y to have size $n-1$. Instead, let us look at the maximum degree locally and at individual stages of the execution of the algorithm. We are able to better grasp the actual number of iterations of the for-loop in line 5 of Algorithm 1 by considering the maximum degree of X or Y, excluding the edge $X - Y$, which we denote by d_{XY}. Formally, $d_{XY} = \max_{W \in \{X,Y\}} \mathrm{d}_{\mathcal{G}'}(W)$where $\mathrm{d}_{\mathcal{G}'}(W)$ denotes the degree of the node W in the graph $\mathcal{G}'$ in which the edge $X - Y$ has been removed. It holds that $d_{XY} \leq n-2$. Now in line 5, there are $\binom{d_{XY}}{k} \in O((d_{XY})^k)$ possible separating sets S to consider for the pair of nodes X and Y chosen in line 4.

The binomial coefficient $\binom{d_{XY}}{k}$ increases with d_{XY}. For a *fixed* d_{XY}, it is minimal for $k = 0$ or $k = d_{XY}$, and maximal for k close to $d_{XY}/2$. This can be seen in the left panels of Figure 4.1, where the upper bound $\binom{n-2}{k}$ is plotted for $n \in \{7, 40, 100\}$.

The fact that $\binom{d_{XY}}{k}$ is minimal for small values of k and reaches is maximum only as k approaches $d_{XY}/2$ means that in the beginning, when d_{XY} is still big, the small value of k ensures that there are only relatively few separating sets to check. When k increases, a significant amount of edges has typically already been removed, and thus d decreases. Thus, the trade-off between large d_{XY} and k close to $d_{XY}/2$ is elegantly optimised in the PC algorithm. Particularly, the theoretical bound $\binom{d_{XY}}{k}$ is avoided save for $k = 0$. This is shown in the right panels of Figure 4.1, where the upper bound (filled dots), the empirical number of iterations for a true graph with expected degree $\hat{d} = 5$ (empty triangles) and with expected degree $\hat{d} = 2$ (empty circles) are plotted for every value of k.

On top of the algorithmic aspects, there are more reasons why it is efficient to gradually increase k: On the one hand, if two variables are not dependent, they are separated by a rather small set of other nodes in practice. On the other hand, low-order conditional

independence tests not only have more statistical power, they are also more efficiently evaluable in practice.

4.1.3 Extensions and Applications

A particular feature of independence-based methods is that they usually presuppose the use of a generic conditional independence test, which makes them adaptive to different kinds of data, but also to new conditional independence tests [266]. For example, the fairly recently developed kernel-based conditional independence tests (see the corresponding paragraph on page 13) can easily be applied.

There are also modifications of the conditional independence testing procedure that were designed specifically for the pc algorithm. In particular, a series of papers has been concerned with conditional independence testing for the broader class of Gaussian copulas. The idea of using rank-based correlation instead of Pearson correlation was first proposed by Harris and Drton [122], resulting in a method called rpc. This was further generalised for mixed (continuous and discrete) data [62], possibly with missing values [63].

Other authors have made it possible to parallelise the pc algorithm. Madsen et al. [173] consider a parallelisation over s, which makes sense for large data sets. Yet, as we have seen in the previous section, the number of nodes is usually more critical. A parallelisation over the nodes is available in the R package `pcalg` [150], which is used here, but also in the package `bnlearn` [237]. This has further been discussed in [165, 315].

Since especially the skeleton estimation is relatively efficient and accurate in the pc algorithm (as was already emphasised by Kalisch and Bühlmann [148]), there are several hybrid algorithms based on the pc algorithm. Tsagris [282] proposed to combine the pc skeleton with the hill-climbing approach of mmhc (see also Section 3.5.3). This is much faster than the original method using the max-min parents and children algorithm.

Gu and Zhou [117] suggest a hybrid approach that applies a divide-and-conquer strategy: Instead of computing the complete causal structure at once, the nodes are clustered and only the subgraphs for the individual clusters are established initially. Then, these subgraphs are merged using a modified bic score. [138] apply this approach specifically to the pc algorithm, enhancing the efficiency of the skeleton-phase. They first cluster the nodes based on their mutual information. They then estimate the skeleton for each cluster separately. This is advantageous not only for the total computational complexity, but also permits parallelisation. "Inter-cluster" edges are estimated only afterwards and filtered in the last phase.

In practice, the pc algorithm is often applied within the framework of a method called ida [171], which uses the pc-estimated graph to compute bounds on causal effects between the variables [172]. Recently, this approach has been adapted in a Bayesian fashion [209, 37].

Recent applications of the pc algorithm include environmental and energy topics [299, 198] – the former based on rpc– as well as biology and medicine [233, 105, 94].

4.2
Order-Dependence of the pc Algorithm in the Oracle Setting

It has been known for a long time [262, 67] that the sample version of the pc algorithm is order-dependent. This means that, assuming the conditional independence tests can produce false information about the independences in the true underlying graph, the pc algorithm can produce different results with different orderings of the input. In other words, the order in which the variables are considered has an influence on the output graph.

Some adaptations have been proposed that order the variables according to certain criteria and thus render the pc algorithm order-independent. [282] suggest to use the heuristics proposed by [114] to determine in each iteration of the skeleton phase which node or set of nodes should be considered next. A similar approach is realised by Cano, Gómez-Olmedo, and Moral [34].

The most widely used order-independent implementation of the pc algorithm is due to Colombo and Maathuis [52]. The authors give examples showing that in fact *each* phase of the pc algorithm is order-dependent and propose a modification for every phase: a stable version of the computation of the skeleton, the use of a voting system for v-structures (available via the options `maj_rule` and `conservative` in the R package `pcalg` as described below) and the recording of conflicting edge directions (as in Algorithm 1). The resulting version of the pc algorithm, known as the *stable* version, is order independent. The basic and stable version of the pc algorithm are evaluated and compared later in this section (see, for example, Figures 4.18 and 4.19). First, however, let us take a more theoretical perspective and consider the order-dependence or independence of the pc algorithm in the oracle model.

In the oracle model, it is assumed that a faithful true DAG $\mathcal{G}^*$ exists, and that all conditional independences are estimated correctly according to $\mathcal{G}^*$. The conditional independence tests can thus be implemented using a d-separation oracle that, to assess the conditional independence $X \perp\!\!\!\perp Y \mid Z$, returns 1 if $X \perp\!\!\!\perp_G Y \mid Z$ and 0 otherwise.

Since the pc algorithm is consistent in the oracle setting, it always returns the correct CPDAG, independent of the ordering in which the variables are given in the input. It thus might seem unreasonable to consider its order-dependence.

Nonetheless, we will see that order-dependence *is* an issue in the skeleton phase of the pc algorithm even in the oracle model in the following sense: Although the skeleton established in the end is always the correct one, the intermediate results may vary with the variable ordering. To be precise, the preliminary skeleton after round k for $k \geq 2$ can be different depending on the variable ordering.

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a DAG and let $\pi_i(\mathcal{V}), i \in \{1, \dots, |\mathcal{V}|!\}$ be the possible permutations of its nodes. Let $G_k^{\pi_i}$ be the skeleton generated by the non-stable pc algorithm after regarding separating sets of size k – that is, after the k-th round – when the variables are regarded in the ordering π_i.

For $k = 0$ and $k = 1$, the pc algorithm is order-independent in the oracle setting.

Lemma 4.1.
In the oracle model, the graphs $G_0^{\pi_i}$ and $G_1^{\pi_i}$ are independent of the permutation π_i.

Proof. In round 0 of the pc algorithm, unconditional independence tests are conducted, which means that the independence of each pair of variables is tested independent of any other variables. Since the results of unconditional independence tests only depend on the data and do not change in the course of the algorithm, the result $G_0^{\pi_i}$ is independent of the ordering π_i of the variables.

It remains to show that $G_1^{\pi_i}$ is independent of π_i. We proof this by contradiction and assume that there is an edge, say $I - J$, with the following properties:

1. $I - J$ cannot be removed in round 0 of the pc algorithm
2. $I - J$ can be removed in round 1
3. $I - J$ cannot be removed in round 1 if one or more other edges are removed first

The latter must hold due to the fact that if one or more other edges are removed, all separation sets $S_{I,J} = \{L\}$ of cardinality $k = 1$ are no longer a subset of the neighbourhood of either I or J.

Now, there might be several separating sets $S_{I,J}$, but since we assume that all connections of such nodes L to I and J must be removable before removing the edge $I - J$, we can assume that all but one node that separates I and J have already been disconnected from both I and J. The only remaining separating set that is in the neighbourhood of I or J is $\{K\}$. We thus have:

4. There is no $L \neq K$ for which $I \perp\!\!\!\perp J \mid \{L\}$.

For the construction above, the following conditional independences must hold:

5. $I \not\perp\!\!\!\perp J \mid \emptyset$: The edge is not removed in round 0.
6. $I \perp\!\!\!\perp J \mid \{K\}$, that is, $\{K\}$ separates I and J.

It follows from 5. that there is at least one d-path connecting I and J in the true graph (see Figure 4.2). Since $\{K\}$ separates I and J (as stated in 6), K must lie on all d-paths between I and J (and not be a collider on any of them), see Figure 4.3. Consequently, there must also be a d-path between I and K and J and K. Since they are thus not marginally independent, the edges $I-K$ and $K - J$ are not removed in round 0. Therefore, in the beginning of round 1, $K \in \mathcal{N}(I)$ *and* $K \in \mathcal{N}(J)$. This can be expressed in the following conditional independences:

Figure 4.2: Part 1.

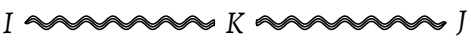


Figure 4.3: Part 2.

7. $I \not\perp\!\!\!\perp K \mid \emptyset$ and $J \not\perp\!\!\!\perp K \mid \emptyset$.

Analogously to the edge $I - J$, the edges $I - K$ and $J - K$ must survive round 0 but be removable in round 1, so two nodes M and N must exist such that

8. $I \perp\!\!\!\perp K \mid \{M\}$ and $J \perp\!\!\!\perp K \mid \{N\}$

In other words, M blocks all d-paths between I and K and N blocks all d-paths between J and K (confer Figure 4.4).

Figure 4.4: Part 3.

But this contradicts the fact that $K \in \mathcal{N}(I)$ and $K \in \mathcal{N}(J)$ as well as assumption 4 because M and N also separate the nodes I and J. Consequently, such a situation cannot arise. □

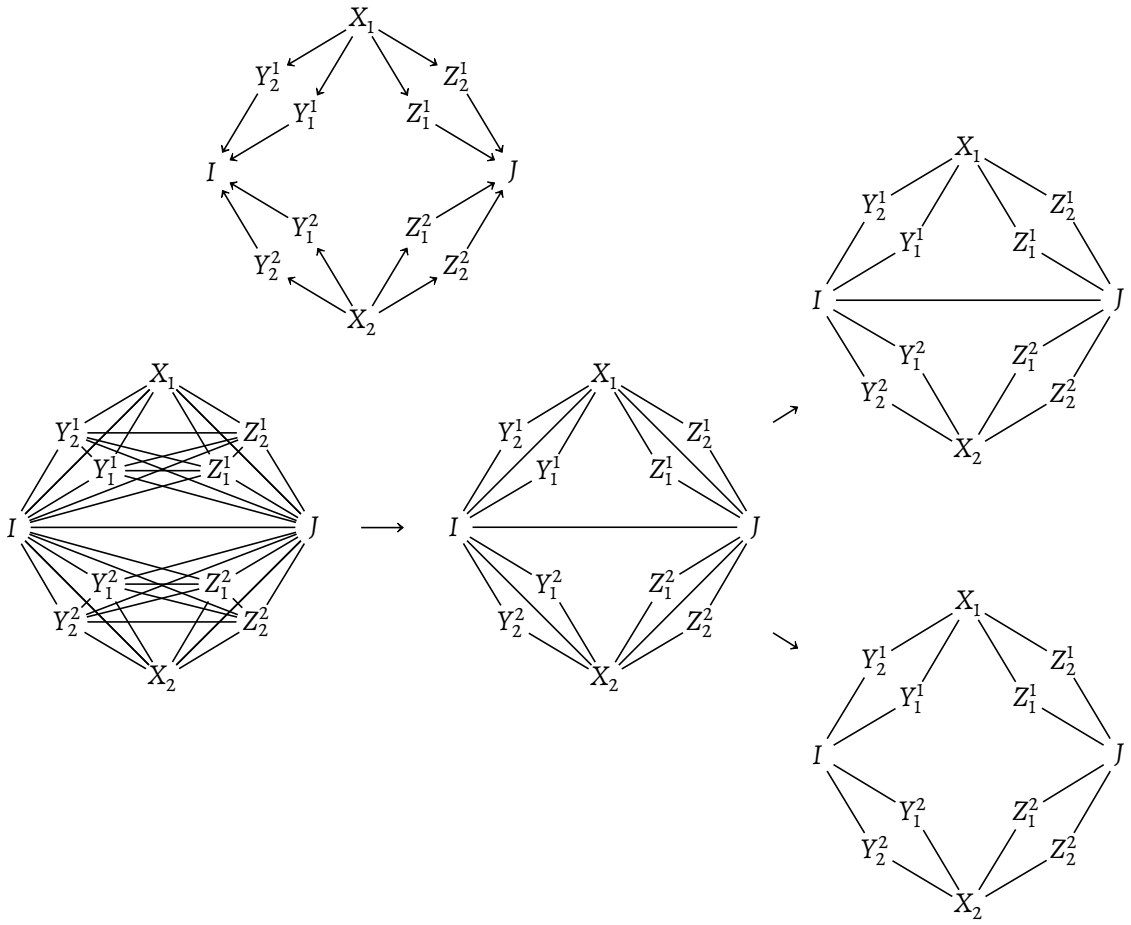


Figure 4.5: Example graph for the proof of Lemma 4.2. The skeleton G_2^π for the true graph shown at the top depends on the variable ordering π. Below, the skeletons G_0^π, G_1^π and the two possibilities for G_2^π are shown.

Lemma 4.2.
There exist instances where the every skeleton G_k^π for $k \geq 2$ depends on the permutation π.

Proof. For $k = 2$ consider the true DAG at the top of Figure 4.5. After round 0, the PC algorithm will have constructed the skeleton to the left in the second row. Note in particular that $X_1 \perp\!\!\!\perp X_2 \mid \emptyset$, because all paths between X_1 and X_2 contain I or J as a collider. After round 1 only the edges in the graph in the middle will be left.

Now, in any ordering in which X_1 and X_2 come before I or J, the conditional independences

$$X \perp\!\!\!\perp I \mid \{Y_1^1, Y_2^1\},$$
$$X \perp\!\!\!\perp J \mid \{Z_1^1, Z_2^1\},$$
$$Y \perp\!\!\!\perp I \mid \{Y_1^2, Y_2^2\} \text{ and}$$
$$Y \perp\!\!\!\perp J \mid \{Z_1^2, Z_2^2\}$$

are tested before $I \perp\!\!\!\perp J \mid \{X_1, X_2\}$, so the edges $I - X_2$, $J - X_1$, $I - X_1$ and $J - X_2$ are removed first. Then, $I \perp\!\!\!\perp J \mid \{X_1, X_2\}$ is not considered at all, because $\{X_1, X_2\}$ is not a subset of the neighbourhood of I or J any more. That is, the edge $I - J$ would not be removed in round 2 of the pc algorithm and the upper right skeleton in the second row would result.

On the other hand, if $I \perp\!\!\!\perp J \mid \{X, Y\}$ is considered first, the edge $I - J$ is removed. The edges $I - X$, $J - X$, $I - Y$ and $J - Y$ would then be removed later, leading to the lower right skeleton in Figure 4.5.

Remark: Of course, also in the former case, the edge $I - J$ would be removed later during the run of the pc algorithm, namely in round 4 when $I \perp\!\!\!\perp J \mid \{A, B, G, H\}$ or $I \perp\!\!\!\perp J \mid \{C, D, E, F\}$ is tested. After all, the pc algorithm is correct in the oracle version, due to Theorem 3.4 (see page 49). But this theorem does not yield *minimum separation* sets S_{XY} and the size of the sets found obviously determines the round in which an edge is removed.

However, for larger k, analogous subgraphs $\mathcal{G}_k$ can be constructed by replicating the nodes X_i, Y_j^i and Z_j^i for $i \in \{1, 2 \dots, k\}$ and $j \in \{1, 2 \dots, k\}$. As in the above example, the following edges exist for every i and j:

$$X_i \to Y_j^i$$
$$X_i \to Z_j^i$$
$$Y_j^i \to I$$
$$Z_j^i \to J$$

Now graphs with arbitrary maximum density d can be constructed that consist of the subgraphs $\mathcal{G}_k$ for $i \in \{1, 2 \dots, k\}$. For such graphs it holds that every skeleton G_k^π for $k \geq 2$ depends on the permutation π.

□

With regard to algorithms that are based on the pc algorithm but only use low-order conditional independence queries like [33, 254, 310], we can summarise as follows:

Theorem 4.3.
When the pc algorithm is terminated before the condition in line 3 of Algorithm 1 is met, the result is dependent on the ordering of the variables in the input even in the oracle setting.

Proof. This is a direct consequence from Lemma 4.2. □

4.3
Experimental Evaluation of the PC Algorithm

After the examination of the oracle version of the PC algorithm, we will now abandon the oracle setting that is mainly of theoretical importance and turn to the sample version of the PC algorithm for the rest of this thesis. However, for the evaluation of various aspects of the algorithm we consider the PC algorithm in a supervised setting, that is, the true graph is known, but not used as an independence oracle. Instead, it is used as a source for the generation of data, on the basis of which a conditional independence test, in our case Fisher's Z-test, assesses the (in)dependences. The resulting graph can than be compared to the true graph to evaluate the estimation performance of the PC algorithm.

4.3.1
Methodology

The evaluation is done using a supervised version of the PC algorithm which creates a random DAG $\mathcal{G}^*$ with a given expected degree, generates s samples of data from this DAG and performs the PC algorithm to obtain $\hat{\mathcal{G}}$. This subroutine is called *supervised PC*. Its basic structure follows the documentation of the package `pcalg` [150] and is given in Listing 4.6.

Listing 4.6: R code of the general procedure supervised PC for the evaluation of the PC algorithm. Code adapted from the documentation of the function `ida`, e. g., variable names have been changed to match the notation in the thesis.

```
## Simulate the true DAG
n <- 10
d <- 2
p_e <- d / (n-1)
alpha <- 0.01
s <- 10000

G_star <- randomDAG(n, prob = p_e) ## true DAG
covTrue <- trueCov(G_star) ## true covariance matrix

## Simulate Gaussian data from the true DAG
dat <- rmvDAG(n, G_star)

## Estimate CPDAG and PDAG
suffStat <- list(C = cor(dat), n = s)
pc.fit <- pc(suffStat, indepTest = gaussCItest, p = n, alpha = alpha)
G_hat <- pc.fit@graph
```

A random graph with n nodes and an expected degree of d is generated in the Erdös-Rényi model (see Section 3.6.3). This graph constitutes the true graph $\mathcal{G}^*$, from which s data instantiations are sampled using ancestral sampling. Given these data, the PC algo-

rithm is used to learn an estimated graph $\hat{\mathcal{G}}$, which then can be compared with the true graph (or analysed separately).

In this procedure randomness is involved in both the generation of the true graph and the data. The results in this thesis are averaged over r replications for each setting, where a new true DAG is used in each repetition.

For comparability, I did not only adopt the same methods as Kalisch and Bühlmann [148] but also the same basic parameters: graph sizes $n \in \{7, 40, 100\}$, expected densities of the true graph $d \in \{2, 5\}$, the value of the tuning parameter $\alpha = 0.01$ and a similar logarithmic scale for the sample sizes $s \in \{10^1, 10^{1.5}, \dots\}$. However, I extended the maximum regarded sample size from $10^{4.5}$ to 10^6. Moreover I replicated each setting $r = 40$ times.

Besides the density d and sample size s, I considered the effects of several other factors:

- How much does the use of Meek's rules for directing non-v-structure-edges influence the correctness of the estimated graph?
- How dependent upon the ordering of the nodes is the result?
- Which effect do the parameters `maj.rule` and `conservative`, that are necessary for the order-independence of the pc algorithm, have?
- Which effect does the detection of conflict edges have, which is also necessary for rendering the pc algorithm order-independent?

Conflict edges are detected as defined in Algorithm 1. To analyse the performance of the pc algorithm with conflict detection, I first defined how they are to be treated in measures like the structural Hamming distance (see Definition 12 in Section 3.6.2). When the correctness of the skeleton is measured, I always consider conflict edges as existing edges, although they can be seen as edges that should be removed (as described in Section 5.2.1).

All results have been produced using the R package `pcalg`, with the modifications indicated (e.g. to the function `shd`). The details of the methods are described below, before the results are presented in Sections 4.3.2 and 4.3.3.

Details of the Implementation Using the R-Package pcalg

All empirical results presented in this chapter are based on the supervised pc procedure (confer Listing 4.6). Random graphs are generated using `randomDAG`, data sampled according to a graph using `rmvDAG` and the pc algorithm is implemented in the function `pc`. The comparison of graph skeletons in terms of TPR, FPR and TDR is implemented in `compareGraphs`; the information whether a graph is extendable is retrieved via the function `pdag2dag`, whether it is a maximally oriented PDAG is determined using the function `isValidGraph` with the parameter `type = "pdag"`. The extendability is assessed using the function from the package `bnlearn` called `cextend` with the parameter `strict = TRUE`, because there should not bee any undirected edges left in the resulting graph. The function `pc` returns the estimated, incorrect (non-CPDAG) graph only with the option `u2pd = relaxed`. Otherwise, if the option `u2pd = "retry"` is used, up to `n.max` (default: `n.max = 100`) permutations of the v-structures are simulated in order to find a valid graph. If this is unsuccessful, tHe behaviour is equal to the one specified by `u2pd = "rand"`: All information about edge directions is discarded and a random DAG is generated on the obtained skeleton.

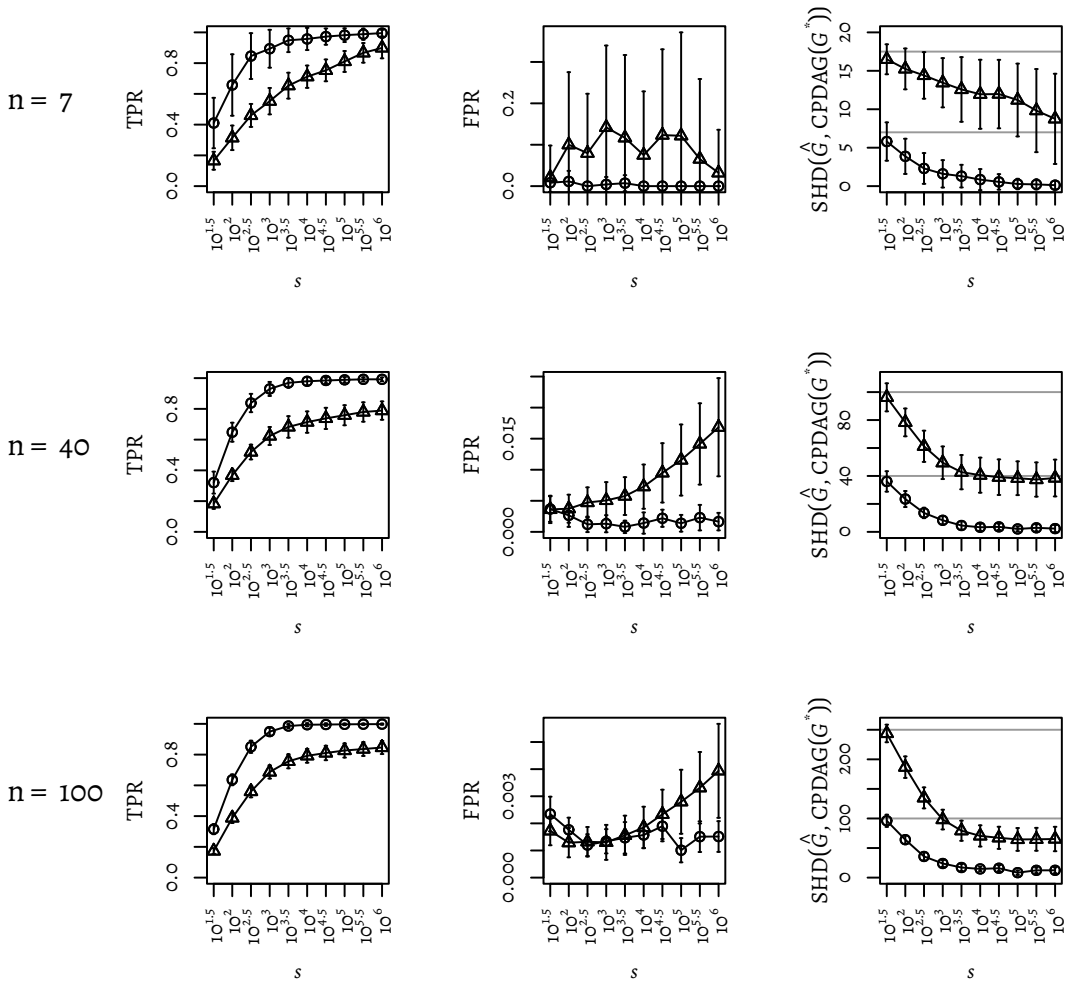


Figure 4.7: Reproduction of Figure 2 from [148]: Measures of the accuracy of the skeleton (false positive rate FPR and the true positive rate TPR of the edges) and structural Hamming distance (SHD) to the true CPDAG, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. The PC algorithm was conducted with Meek rules, without the options `maj.rule` and `conservative`, and without conflict detection. For comparison, faint background lines in the right plots show the mean total number of edges of the corresponding PC-estimated graphs.

Since causality is fundamentally about edge directions, it is obvious that the randomly directed edges are unsatisfactory. Therefore, the option `u2pd = relaxed` is used throughout this thesis.

Moreover, inconsistencies that occurred during the computation should not be resolved randomly. Instead, the option `solve.confl = TRUE` can be used. In this case,

conflicting information is represented explicitly in the graph in terms of a new type of edges that are called *conflict edges* in this thesis. Note that this option only makes the conflicts explicit by marking the edges accordingly, but does not *solve* the conflicts as the name suggests. The formation and processing of conflict edges is discussed in detail in the next chapter, in particular in Section 5.2.1 and Section **??**. However, conflict edges are already included in the empirical analysis in this chapter to estimate, for example, how often they occur.

In order to reproduce the results obtained by Kalisch and Bühlmann [148] the option `solve.confl = FALSE` is used. After that, that is first in Figure 4.18, the option was switched to `solve.confl = TRUE` for the stable version of the pc algorithm. For this, also the options `maj.rule = TRUE` or `conservative = TRUE` are used as indicated.

Reproduction of the Results by Kalisch and Bühlmann

In Figure 2 of [148], the convergence of the accuracy of the pc algorithm for increasing sample size s is shown in terms of true and false positive rate of the edges (TPR and FPR) and structural Hamming distance (SHD) which are defined in Section 3.6.2. The empirical analysis in this and the next section is based on this figure.

To first reproduce it, I assume that Kalisch and Bühlmann used Meek's rules, but neither the conservative or majority-rule-based identification of v-structures nor the stable version of the `skeleton` function since all these options were first published seven years later [52]. I further assume that also the option to explicitly handle conflict edges is a later invention.

The results can be found in Figure 4.7 and are in accordance with Kalisch and Bühlmann's in magnitude and asymptotic behaviour. Also the curve shapes are comparable. Deviations are visible only for the false positive rate and for $n = 7$. This is behause the FPR is very small in general and some fringe effects seem to occur for $n = 7$.

4.3.2 Empirical Analysis of the pc Algorithm

In addition to the measures taken into account by Kalisch and Bühlmann [148], I determined other measures including

1. the density of the estimated graph;
2. the number of conditional independence tests evaluated in the course of the pc algorithm and in each round;
3. the true detection rate (also known as the positive predictive value), see Section 3.6.2;
4. the fraction of graphs that were extendable, maximally oriented PDAGs, or proper CPDAGs.

In particular the latter analysis is novel. So far no systematic experimental analysis has been done, which shows the performance of the pc algorithm in terms of the extendability and related properties of the resulting graphs. The results are presented in Figure 4.8 (density end edge types), Figure 4.10 and Figure 4.11 (TDR and conditional independence tests overall) and 4.12 (type of the estimated graph) and are discussed in the next paragraphs.

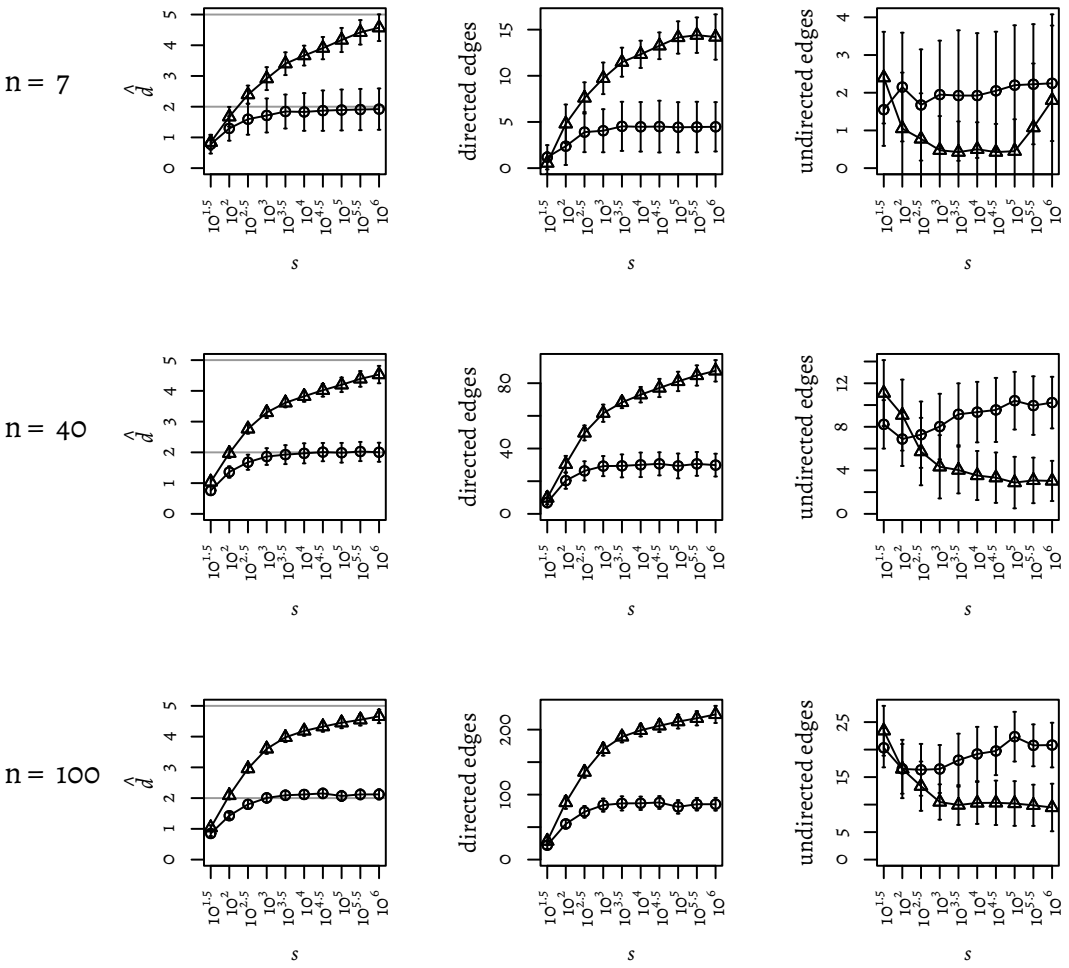


Figure 4.8: Density and number of directed and undirected edges in the graphs generated by the PC algorithm, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. For comparison, faint background lines in the left plots show the expected density d of the true graph.

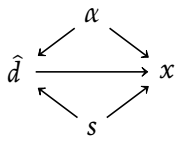


Figure 4.9: The causal diagram showing the dependence of the results of the PC algorithm upon the density $\hat{d}$ of the estimated graph and the parameters s and α. The variable x stands for a parameter that is influenced by the choice of s, d and α, such as the SHD, number of conditional independence tests performed or the extendability.

Density and Edge Types

The degrees of the randomly created true graphs deviate only marginally from the desired values: The mean degree over all replicates is 2.00 for $n = 7$, 2.02 for $n = 40$ and 1.99 for $n = 100$ when $d = 2$ was aimed at, and 4.94 for $n = 7$, 5.08 for $n = 40$ and 5.01 for $n = 100$ when $d = 5$ was desired.

However, the density of the estimated graphs slowly evolves towards the desired value. That means that for small values of s, the resulting graphs are significantly more sparse than expected, as shown in Figure 4.8. This issue is examined in more detail in Section 4.3.4, but it is worth noticing that the expected degree increases with s so that the graph density often might be the true cause of effects that arise from varying s. This is illustrated in the causal diagram given in Figure 4.9.

To cancel the effect of the graph density, it would be helpful to hold the degree of the graphs constant. This can be done by using α as a tuning parameter and adapting it in order to always obtain graphs with the same density, as is further investigated in Section 4.3.5.

It can further be seen in the Figure 4.8 that while the number of undirected edges is more or less constant, it is the number of directed edges that increases with s. This can be explained by the fact that in very sparse graphs nearly no v-structures occur, such that no directed edges can be identified.

The number of v-structures decreases again for very dense graphs, that is when the density approaches the upper limit $n - 1$. Then, most triples of nodes are shielded and thus cannot form a v-structure. A slight tendency for the number of directed edges to decrease for the largest values of s and thus the highest densities is evident for $n = 7$. However, even an average degree of 5 is not enough for this effect to be visible for $n = 40$ and $n = 100$.

True Detection Rate and Conditional Independence Tests

Complementing the measures shown in [148], the TDR is given in the left column of Figure 4.10. Recall from Section 3.6.2 that the TDR is defined as the number of correctly identified edges relative to the number of edges in the estimated graph. It can be seen that while the PC algorithm converges in terms of the TPR (left column in Figure 4.7) when s increases, this is not the case for the TDR. A preliminary explanation for this – that will be verified in Figure 4.22 – is that the number of edges in the estimated graph increases with s so that finally all true edges are recovered at the cost of identifying also some erroneous edges.

The number of conditional independence tests executed is shown in Figure 4.10. It is notable that most of them are of very low order, although the number of high-order tests increases with n and s. One might think that in the first round, i. e. for order zero, the number of conditional independence tests should always be equal, namely exactly the number of node pairs $\frac{n \cdot (n-1)}{2}$ (for instance, 21 for $n = 7$. That this is not the case is due to the details of the implementation. In the package `pcalg`, every node pair (line 4 in Algorithm 1) is actually regarded twice (once in each ordering), and each time the putative separating sets are chosen among the neighbours of the first node only. This way, the empty set is regarded twice in the first round for each edge that is *not* removed. The other edges need to be considered only once so that the theoretical upper bound of 42 for $n = 7$ – 1560 for $n = 40$ and 9900 for $n = 100$, is not reached for sparse graphs, in which some

edges can be removed in round 0. As can be seen in the figure, most graphs are so dense that the number of nodes removed in round 0 is not noticeable.

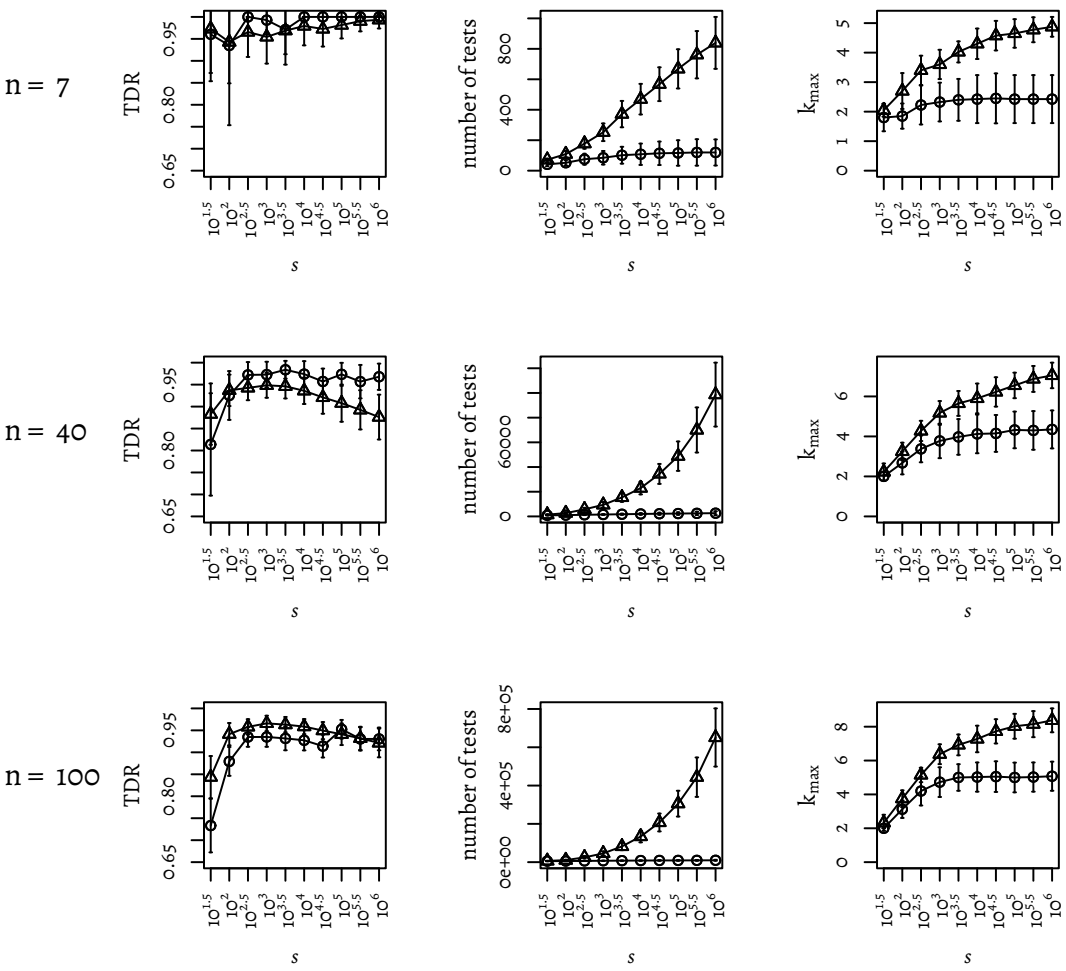


Figure 4.10: True detection rate (TDR) and number and maximal order k_{max} of conditional independence tests conducted by the PC algorithm, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals.

Figure 4.11: Number of conditional independence tests of certain order k executed by the pc algorithm, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. The range of the ordinate is equal in each row, to simplify the comparison of the numbers of tests of different orders. Horizontal lines mark theoretical upper bounds that would be reached if no edges had been removed at the current order k or a lower orders. If these lines are missing, they lie outside the range of the ordinate (confer Figure 4.1).

The number of conditional independence tests that the PC algorithm has to evaluate is in general exponential, but depends strongly on the degree of the graph in each round, as described in the analysis of the theoretical complexity of the PC algorithm in Section 4.1.2. A comparison of the maximal and actual numbers of conditional independence tests executed can be found there as well (see Figure 4.1).

Since the degree of the graph increases with s, it is not surprising also the number of tests does, in particular for a large degree of the true graph. Figure 4.22 shows that this increase is actually due to the density. On the other hand, it is interesting to note that the maximum order of a conditional independence test executed, albeit increasing with s, seems to converge towards a value below 10 although it could theoretically be as large as $n-2$. This explains the feasibility of the PC algorithm in sparse high-dimensional settings.

In Figure 4.11, one can see for which orders the most conditional independence tests are conducted. Interestingly, when $d = 5$ this is always the case for $k = 2$. For greater k that, whilst the theoretical upper bounds still increase, the combinatorial growth has been brought under control by the decreasing number of remaining edges and, with and beyond that, the decrease of the number of nodes with that high a degree.

Type of the Estimated Graph

In Figure 4.12, one can see the fractions of graphs estimated by the PC algorithm that are PDAGs, extendable PDAGs, or proper CPDAGs. As visualised in the Euler diagrams in Figure 5.1, the class of CPDAGs is a strict subset of the set of extendable graphs, which in turn is a strict subset of the class of PDAGs. In the setting regarded here, the only reason for a graph not to be a PDAG is that it contains a directed cycle. The details of the different graph classes are discussed in Section 5.1.

In the literature, the PC algorithm is usually said to output a PDAG [270] or a CPDAG ([148, 122, 52, 129]), the latter sometimes denoted by a different term [75, 123]. The first fact that we can conclude from Figure 4.12 is that due to sampling errors the PC algorithm nearly never outputs a proper CPDAG, and sometimes not even a PDAG. We therefore drop even the assumption of acyclicity. Allowing also conflict edges to be present, we refer to the estimated graph as a general partially directed graph (GPDG), as defined in Definition 13.

At least cyclic outputs are not very common: Cases with less than 50% proper PDAGs only occur for very small graphs ($n = 7$). However, the rates of PDAGs bound the rates of extendible graphs and CPDAGs. While the rates of PDAGs and extendable graphs are identical in this case, indicating that acyclicity is the only relevant cause of non-extendability, the fraction of CPDAGs in the output graphs rapidly drops to zero with increasing sample size, probably due to the higher density that large sample sizes entail (see also Section 4.3.4).

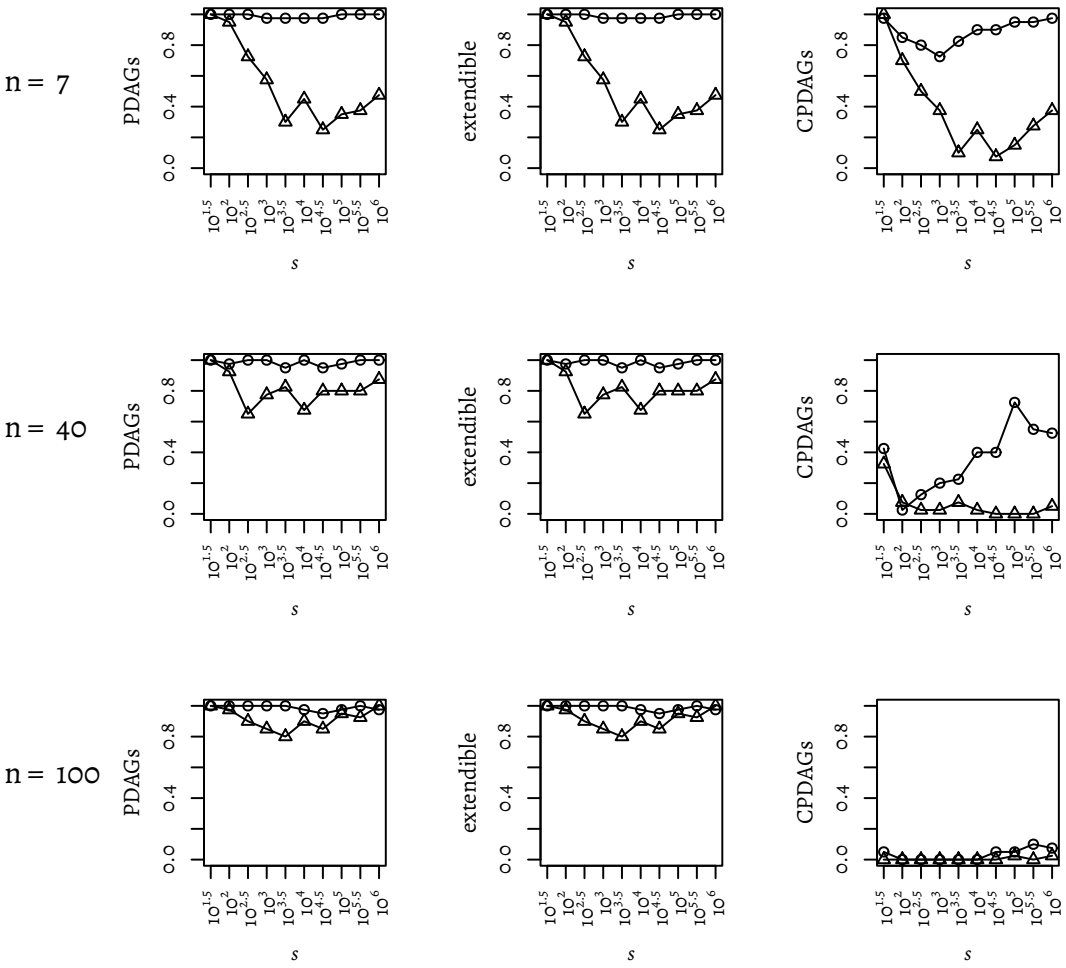


Figure 4.12: Fractions of graphs estimated by the pc algorithm that were proper pdags, extendable, and proper cpdags, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. Note that the regarded properties are gradually more restrictive as illustrated in Figure 5.1.

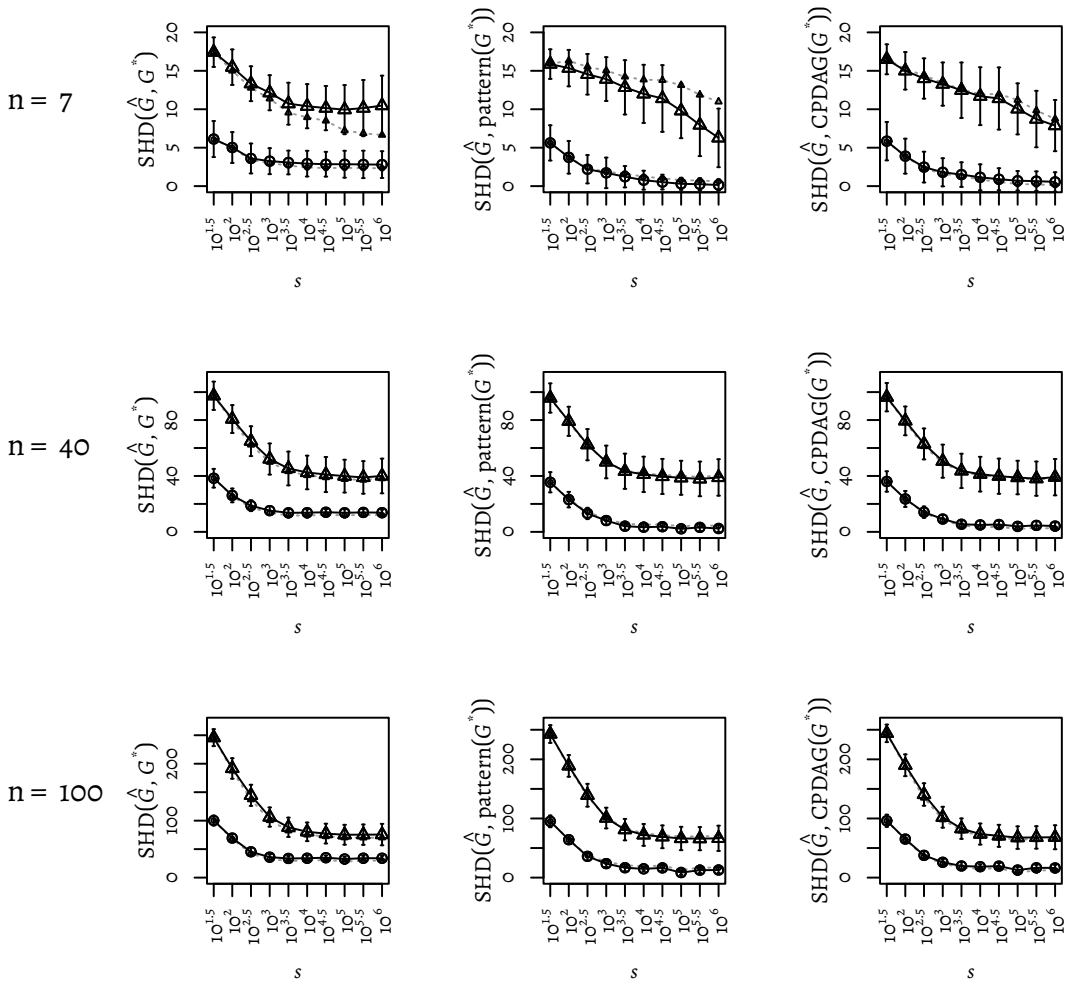


Figure 4.13: Structural Hamming distance (SHD) of the graphs generated by the PC algorithm *without applying Meek's rules* to the true DAG, its pattern and its CPDAG, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. Note that, because Meek's rules are not applied, the estimated graph would be the pattern of the true graph if no errors occurred. For comparison, faint background lines represent the corresponding results *with* Meek's rules.

There is a weak but perceptible tendency of the acyclicity rate to first decrease and then increase again (with s), which also can be explained by the graph density. For very low values of s, the resulting graphs are so sparse that edges hardly ever interfere, so each edge can be directed independently and every graph is extendable. As s and thus the graph density increase, extendability by chance becomes more and more infrequent and the fraction of non-extendable graphs reaches its maximum. Then, as s increases further, the

computed graph more and more resembles the CPDAG of the true DAG, which of course is extendable. However, the estimated graph hardly ever seems to be exactly the true CPDAG because nearly none of the graphs for very high values of s is a CPDAG, as can be seen in the rightmost column of Figure 4.12. Note, however, that one small error can turn even the true graph into a non-CPDAG, so that the fraction of proper CPDAGs might be an overly pessimistic measure of the accuracy of the PC algorithm, at least for large graphs.

4.3.3 Empirical Analysis of Variations of the pc Algorithm

Having reproduced and expanded upon Kalisch and Bühlmann's results, we will now investigate how they depend on the version of the PC algorithm by which they were obtained. Before looking into new extensions of the PC algorithm, let us first consider its last phase, where non-v-structure edges are directed based on v-structures. The basis for this is a set of rules referred to as *Meek's rules* (see Figure 2.2).

Meek's Rules

Meek's rules were devised to turn a pattern, which in the oracle model is the output of phase two of the algorithm, into a CPDAG. Because of this, the PC algorithm is commonly said to output a CPDAG. In fact, this is true only in the oracle model, that is when all conditional independences are correct. We have seen in the previous section that in the sample setting, the result of the PC algorithm is hardly ever a CPDAG, which indicates that Meek's rules do not have the desired effect when the initial graph does not have the expected properties. To get to the bottom of problems like the non-extandability of estimated graphs, we will thus abandon Meek's rules for the time being.

Since Meek's rules are not applied before the skeleton is established, discarding them does not have any effect on the measures that assess the correctness of the adjacencies, like TPR, FPR and TDR. However, it is interesting to investigate what happens to the structural Hamming distance (SHD) to the true DAG, its pattern and its CPDAG when Meek's rules are not applied; see Figure 4.13 for a comparison to the case with Meek's rules.

The results are consistent with what is to be expected. Remember that the pattern of a graph is a graph that retains skeleton and v-structures, but is otherwise undirected. Since Meek's rules are used to direct edges that are *not* part of v-structure, applying them will generally increase the SHD to the pattern of the true DAG. This can be seen in all regarded settings, although the effect is small for larger graphs ($n = 40, n = 100$).

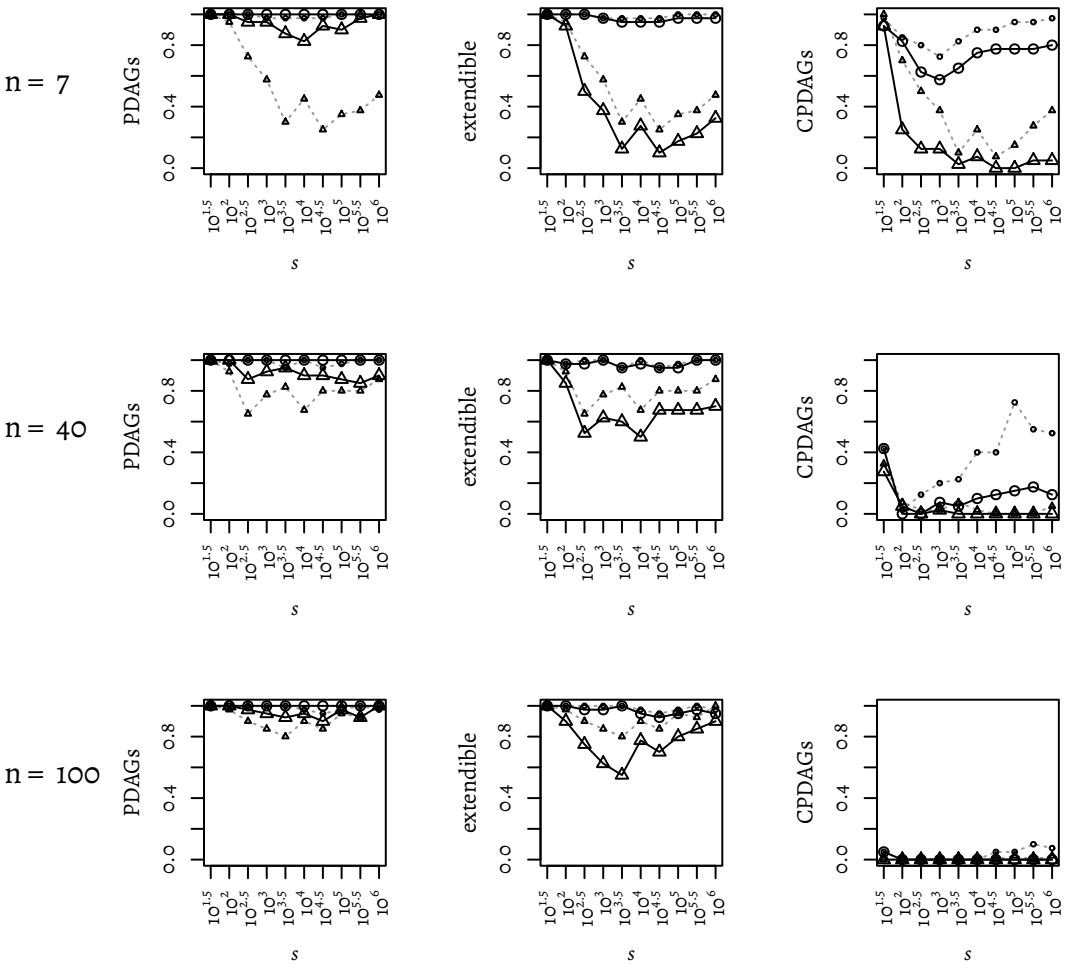


Figure 4.14: Fractions of graphs estimated by the PC algorithm *without applying Meek's rules* that were proper PDAGs, extendable, and proper CPDAGs, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. Note that, because Meek's rules are not applied, the estimated graph should be a pattern if no errors occurred. For comparison, faint background lines represent the corresponding results *with* Meek's rules.

On the other hand, Meek's rules decrease the SHD to the true graph and its CPDAG by directing also non-v-structure edges. Nonetheless, at least for $n = 7$ the effect of Meek's rules is smaller when comparing to the true CPDAG. This might indicate that in the sample setting not only edges that are directed in the CPDAG are directed by Meek's rules. Instead, more edges may be directed, some of them in the way they are directed in the true DAG. Some might also be directed in the opposite way as this would not influence

the SHD to the true DAG. If an undirected edge $X - Y$ is directed as $X \to Y$ by Meek's rules, although it is directed as $X \leftarrow Y$ in the true DAG, the SHD to the true DAG does not change since the edge constitutes a mismatch either way.

Figure 4.14 shows the fractions of graphs estimated by the PC algorithm the are PDAGs, extendible graphs or CPDAGs when Meek's rules are *not* applied (compared to the setting with Meek's rules that is shown in Figure 4.12). Again, we can see that the rates first decline and then improve with s, which can be explained by the graph density in the same way as the acyclicity in the case with Meek's rules.

It is not surprising that applying Meek's rules increases the probability of generating directed cycles, simply because more edges are directed. On the other hand, the extendability improves when Meek's rules are applied (or worsens if not). Since the true CPDAG is extendable by definition, the obvious explanation for the latter would be that Meek's rules simply render the result more "correct", that is to say lead to a higher agreement with the true CPDAG. After all, they are designed to generate the (true) CPDAG from the (true) pattern. However, the SHD to the true CPDAG improves only insignificantly and the fraction of output graphs that are proper CPDAGs is only slightly increased (if at all). This shows that another effect is of higher importance: Meek's rules lead undirected edges being forcedly directed. If an edge that can not be directed by the rules of graph extension is directed by Meek's rules, the graph might become extendable just by bringing the error forward. A small example for this is given in Figure 4.15. Note that the situation also occurs in each graph that contains the example graph as an induced subgraph.

$$W \to X - Y \leftarrow Z \qquad W \to X \to Y \leftarrow Z$$

Figure 4.15: Meek's rules can render a non-extendable graph extendable. The graph on the left is non-extendable because however the middle edge is directed, a v-structure is created. Meek's first rule, locally avoids v-structures by tuning an induced subgraph $X \to Y - Z$ into $X \to Y \to Z$. In this case, however, applying this rule would still produce a v-structure with the third edge. For example, if $A \to B - C$ is regarded first, the graph to the right is created. Since this graph is a DAG, it is extendable, although the original graph was not. The application of Meek's rule thus lead to an invalid extension of the graph on the left.

All in all, we can conclude that in the sample setting Meek's rules do not – as they are designed to – reliably direct non-v-structure edges in order to create a CPDAG. In fact, they might spoil the result by inducing directed cycles, although this is precluded in the oracle model. Furthermore, their application might disguise the accuracy of the previous phases by implicitly and without any notice solving conflicts in a random way. I therefore exclude them for the rest of the section and instead regard the pattern that the PC algorithm outputs when only its first two phases are performed: Learning the skeleton and directing edges that are part of v-structures, which can be identified from the conditional independence relationships determined during construction of the skeleton.

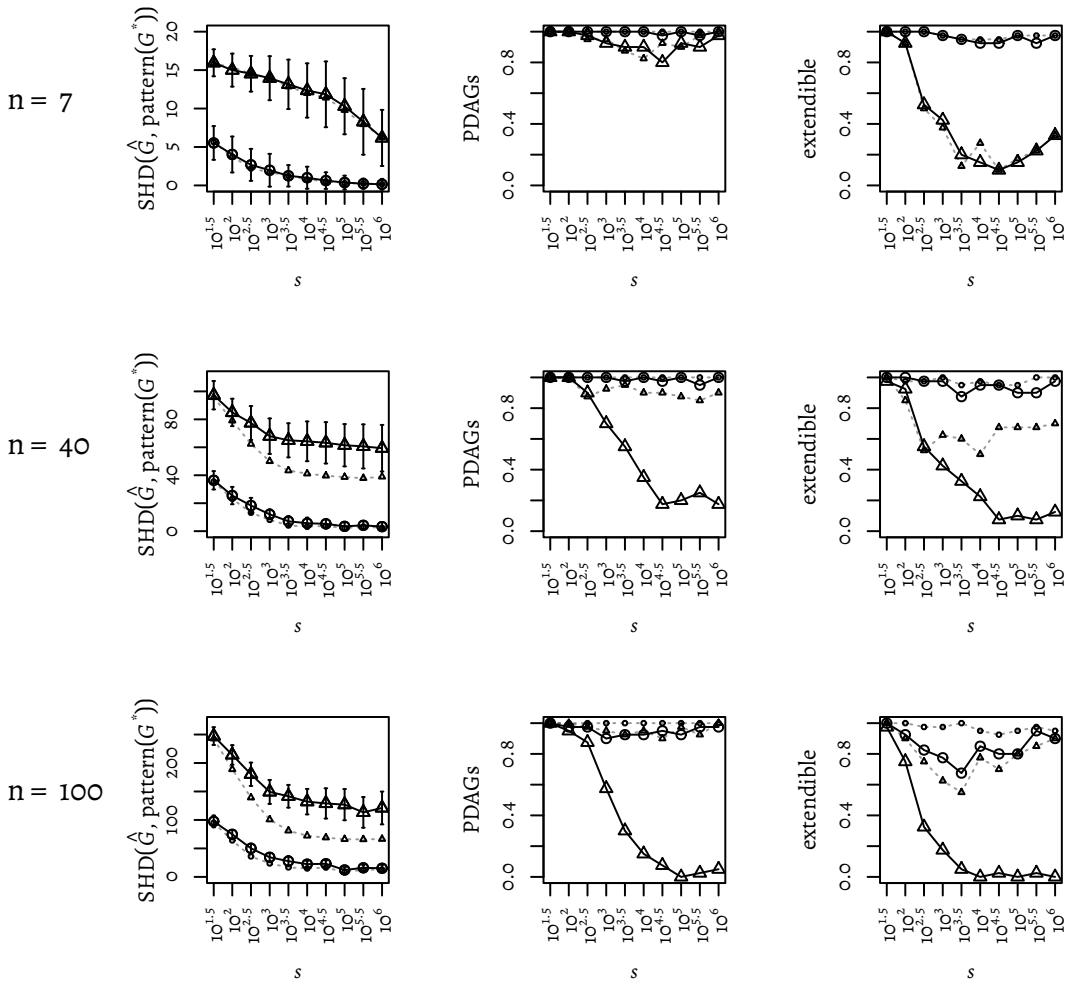


Figure 4.16: Structural Hamming distance to the pattern of the true DAG as well as fractions of graphs that were proper PDAGs or extendable for the graphs estimated by the order-dependent version of the PC algorithm, *when variables are permuted after data generation*. The measures are shown dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. For comparison, faint background lines represent the corresponding results *without permutation*, that is when the variables are given to the (order-dependent version of the) PC algorithm in a causal ordering. Meek's rules were not applied in both cases.

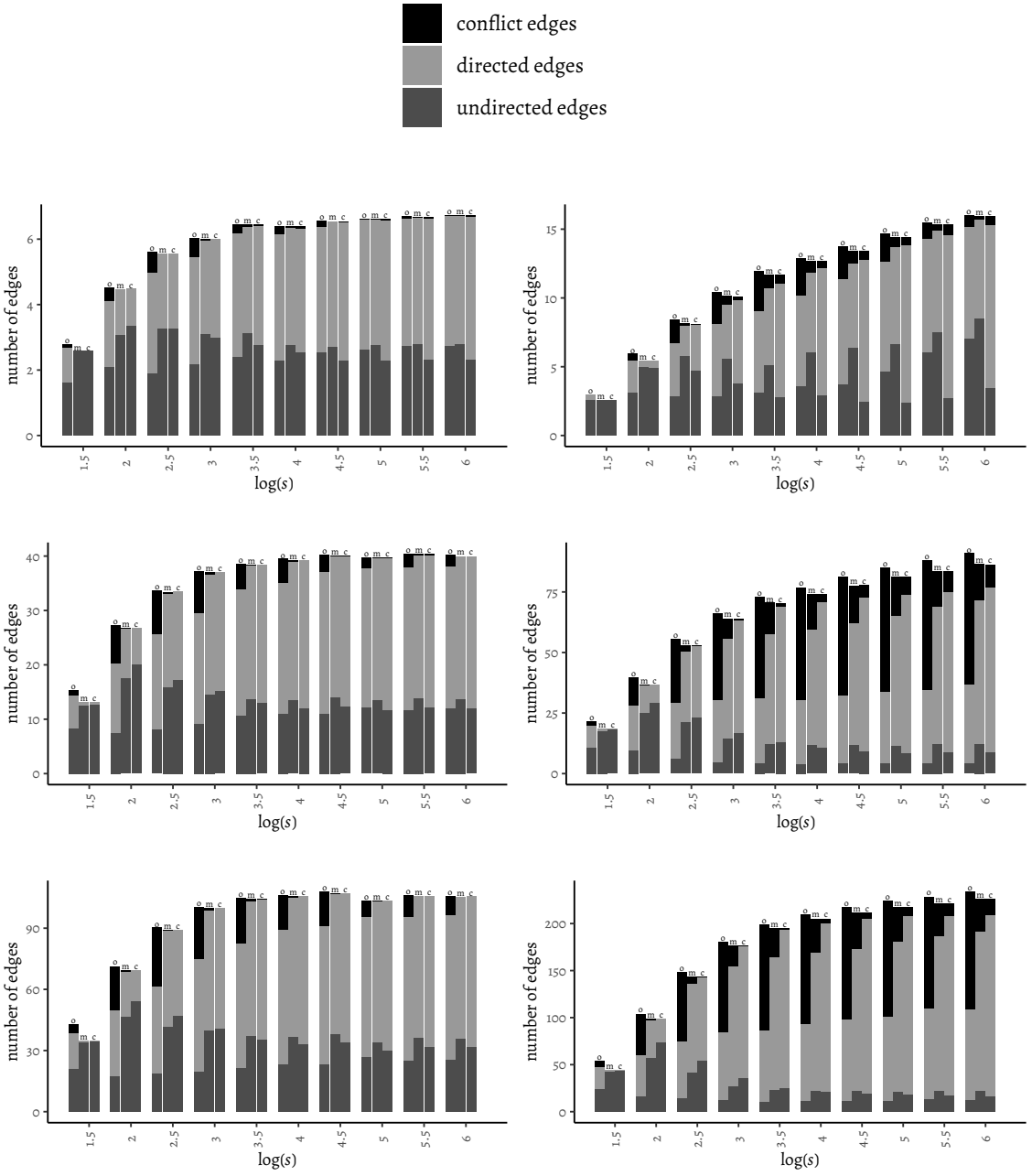


Figure 4.17: Number of edges of different types that are present in the output graph of the pc algorithm for different settings: original, non-stable pc with permuted variables (and conflict detection) (o); stable pc with majority rule (m); stable conservative pc (c). The measures are shown dependent on the sample size s; each stack consists of the number of conflict edges (bottom), directed edges (middle), and undirected edges (top). The rows correspond to numbers of variables $n \in \{7, 40, 100\}$ (from top to bottom), the two columns to $d = 2$ (left) and $d = 5$ (right). Meek's rules were not applied in all cases.

Order-Dependence

It remains to shed light on one more issue of the order-dependent version of the R function `pc`. When the basic procedure proposed in the documentation of the R package `pcalg` is used (similar to what is shown in Listing 4.6 and further described in Section 3.6.3), all edges in the true DAG are generated to be directed from a node with a lower index to a node with a higher index. If the variable ordering is not changed after the data generation, the PC algorithm receives the data with a variable numbering corresponding to a topological ordering of the nodes in the true graph. However, when the PC algorithm is applied in practice, such an ordering is usually unknown. Yet, *in principle* this ordering provides a lot of information about the true causal graph (confer the last paragraph of Section 3.5.2).

This information is of course not used explicitly in the PC algorithm, e. g., by deleting all edges from nodes with a higher index to nodes with a lower index. Nonetheless the PC algorithm would always regard the nodes in topological order. This influences the edge directions in the resulting graph in a profitable way, as can be seen when comparing with the results obtained when the columns of the data are randomly permuted after generating the data[9].

In Figure 4.16, this comparison is shown. After permuting the data, the measures of the skeleton, TPR and FPR, do not change considerably (not shown). However, the SHD of the produced graph to the true DAG, pattern and CPDAG increase significantly, as shown exemplarily for the SHD to the pattern in the figure. (It makes sense to consider the SHD to the pattern here, as we still consider the version of the PC algorithm where the last phase, the application of Meek's rules, is dropped.) If the skeleton remains the same, an improvement of the SHD must be due to edge directions. When the last phase of the algorithm is skipped, the only edges that are directed are putative v-structures. We can conclude that considering the nodes in their topological ordering facilitates the identification of v-structures. This might be because it is more probable to find "correct" separating sets in the skeleton phase when iterating over the nodes in a topological manner, which is important for the identification of v-structures. This hypothesis is in accordance with the finding that the number of conditional independence tests in the skeleton phase increases when the variables are permuted.

In Figure 4.16, one can see that the extendability is significantly enhanced owing to the gain in edge direction accuracy which results from using the topological ordering of the variables. One can see that nearly all estimated denser graphs ($d = 5$ and larger values of s) contain at least one cycle and thus are not valid PDAGs. The fraction of *extendable* PDAGs drops towards zero for even smaller sample sizes (that is, densities). The fraction of CPDAGs that is close to zero already when the topological ordering is used (see Figure 4.15) does not change noticeably (not shown).

[9] The fact that the column names still correspond to a topological ordering of the nodes is not important, since the PC algorithm never explicitly uses them. Instead, it always regards the nodes in the order of their appearance in the adjacency matrix, that is in the data.

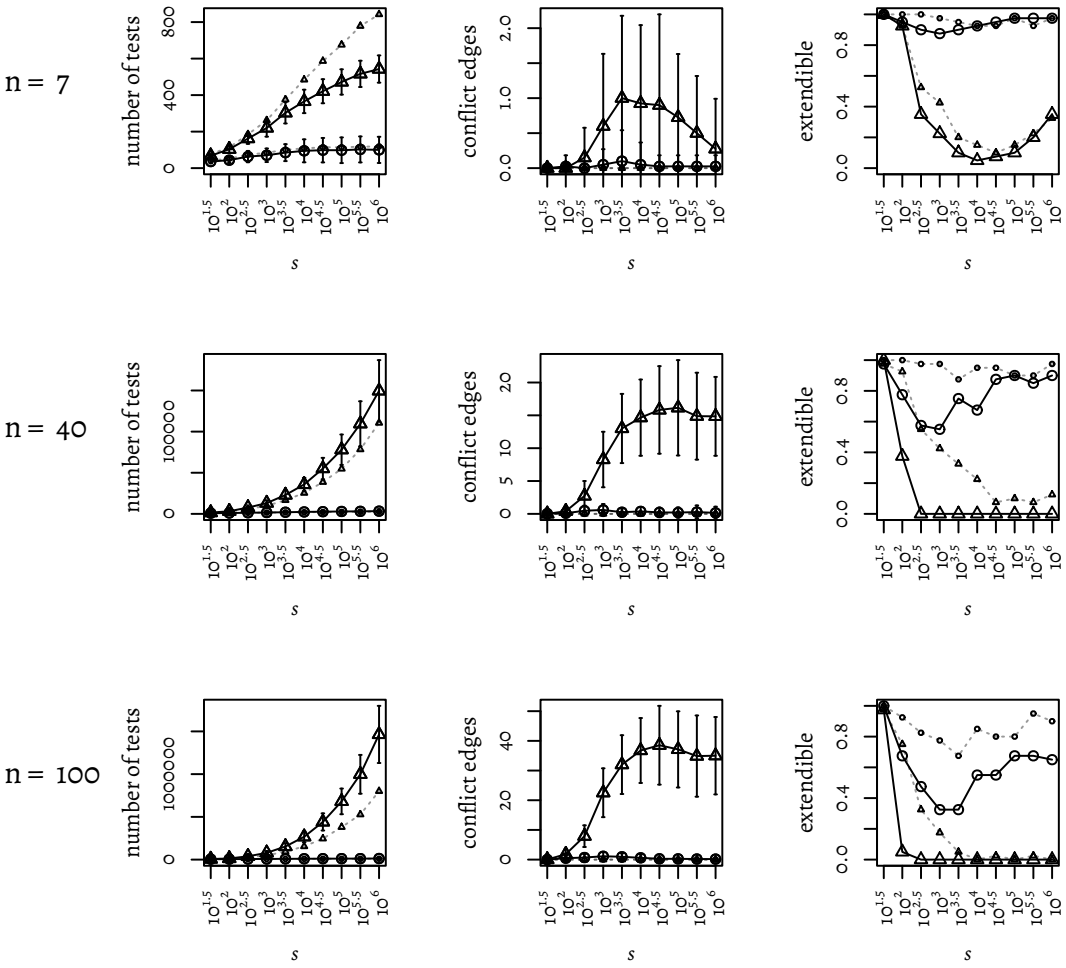


Figure 4.18: Number of conditional independence tests conducted, number of conflict edges in the estimated graph and fraction of extendable output graphs for the *stable version of the pc algorithm using the majority rule*. The measures are shown dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. For comparison, faint background lines represent the results of the *original, order-dependent method without conflict detection with randomly permuted variables*. Meek's rules were not applied in both cases.

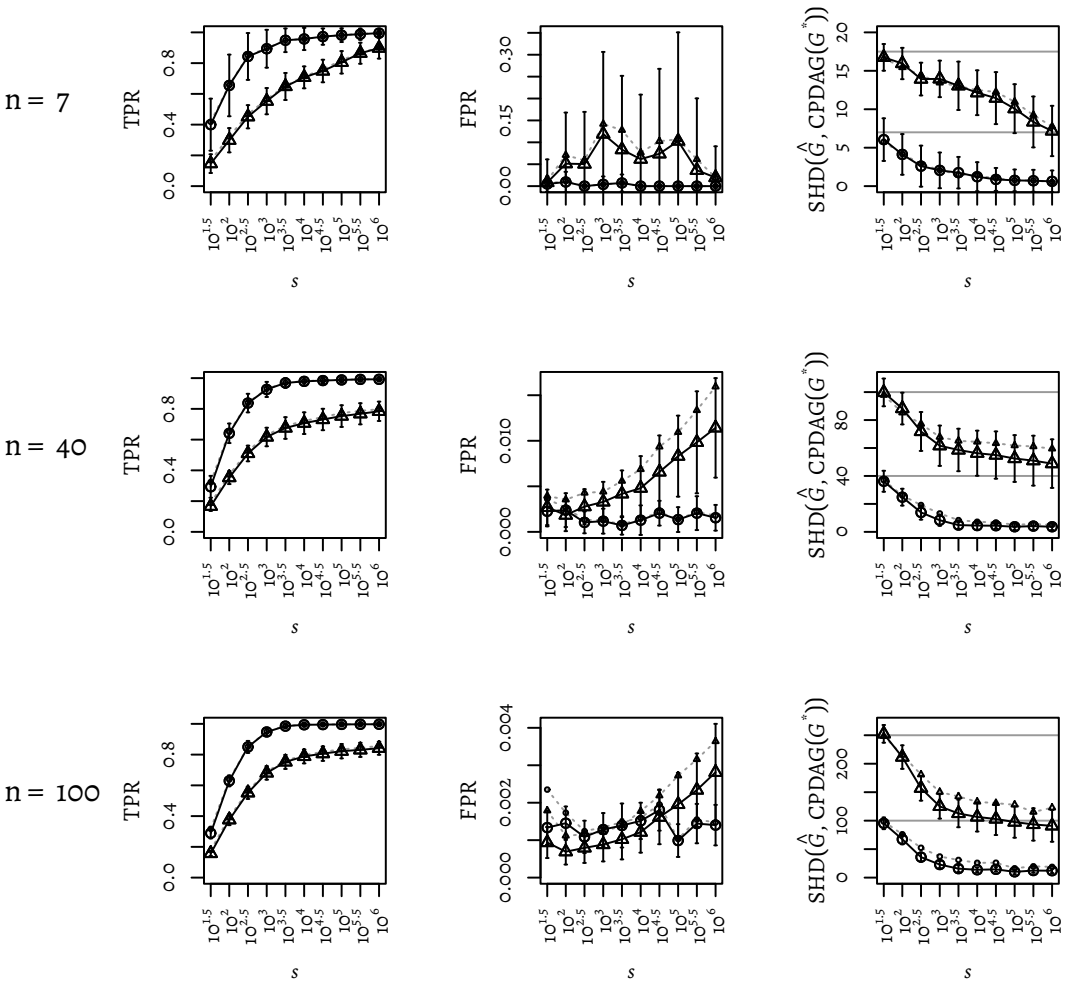


Figure 4.19: Reproduction of Figure 2 from [148] (see Figure 4.7) using the *stable version of the* PC *algorithm with the majority rule*: Measures of the accuracy of the skeleton (false positive rate FPR and the true positive rate TPR of the edges) and structural Hamming distance (SHD) to the true CPDAG, dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. The measures are shown dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. For comparison, faint background lines represent the results of the *original, order-dependent method without conflict detection with randomly permuted variables*. Meek's rules were not applied in both cases.

The Stable Version of the pc algorithm

We have seen that, in practice, the pc algorithm profits significantly from a topologically sorted input. The general fact that the output of the pc algorithm depends on the ordering of the variables has been known before and exploited for example by Dash and Drużdżel [67]. On top of that, a stable version of the pc algorithm by Colombo and Maathuis [52] resolves the order-dependence of the pc algorithm.

It consists of three modules that remedy the order-dependence of the three phases of the pc algorithm. Note that only if all three of them are performed in an order-independent way, the pc algorithm will be completely independent of the variable ordering.

To render the computation of the skeleton order-independent, edges are not deleted from the graph in the course of one round of the computation; instead, edges that are to be removed due to the results of the conditional independence tests are labelled as such and removed in the end of the round.

To resolve the order-dependence of the identification of v-structures, a voting procedure is introduced. Remember that in the original version of the pc algorithm, a chain $A - B - C$ is directed as a v-structure $A \to B \leftarrow C$ if $B \notin S_{AC}$, with S_{AC} being the separating set that caused the removal of the edge $A - C$ in the skeleton phase. In the oracle version, B will either be in all sets that separate A and C or in none. However, sampling errors can lead to a set without B being deemed separating although the structure is not a v-structure, or the other way around. It is therefore not only a possibility to attain order-independence, but makes the computation generally more robust to regard multiple separating sets instead of only the first one found.

In the majority rule version of this phase (available in the `pcalg` available via the parameter `maj.rule`), $A - B - C$ is directed as a v-structure if the majority of the separating sets made up by the neighbours of A and C does not contain B. Otherwise, the chain is not directed. If the conservative version of this phase is used (parameter `conservative` in the `pcalg` implementation), a v-structure is created only if B is in none of the separating sets. These two adaptations are gradually more strict with respect to creating v-structures, so they generally result in a higher number of undirected edges, as can be seen in Figure 4.17.

Finally, it matters to which subsets of variables Meek's rules are applied first, as we have seen in the example in Figure 4.15. There, the edge $B - C$ can be directed as $B \to C$ or $B \leftarrow C$, depending on whether $A \to B - C$ or $D \to C - B$ is considered first. It turns out that the case shown in this figure, where there is conflicting information about the direction of one edge, is all that needs to be considered in order to certain the order-independence of the last phase. It suffices to mark such edges with unresolved orientation as ambiguous, instead of directing them in one or the other direction. As Colombo and Maathuis [52] remark, such edges have no causal interpretation, but rather indicate conflicts in the conclusions of the pc algorithm and are therefore called conflict edges here. There is no obvious solution for this issue which is discussed in more detail in Section 5.2.1.

It can be seen in Figure 4.18 that the number of conflict edges is non-negligible. On the other hand, their number does not seem to increase unbounded with the graph density, i. e. increasing s. Rather, even for the denser true graphs ($d = 5$) it does not exceed half the number of nodes. Furthermore, when considering small graphs ($n = 7$) it can

be seen that the number of conflict edges decreases again for very large samples, which might indicate that the true graph is more accurately learned.

When comparing the stable majority rule version of the PC algorithm with the unstable version as shown in Figure 4.19, we can see that both the FPR and the TPR decrease slightly but consistently. Also Figure 4.17 confirms that in general, marginally fewer edges are found. This is not surprising because in the stable version of the computation of the skeleton, edges are removed at a later stage (namely at the end of the round). This implies that the neighbourhoods of the nodes remain larger for a longer time, so that the probability that a separating set is found within such a neighbourhood – and may it be by chance – is increased. It seems that most of additional separating sets that are found in this way are correct since the TPR decreases only slightly, while the effect is more clearly visible for the FPR.

However, this comes at a cost. In the stable version, more conditional independence tests are performed. The number of performed conditional independence tests is about twice as high for medium and large sized graphs (i. e. with $n \geq 40$), as can be seen in Figure 4.18.

Although the SHD does not revert to the values attained with the unstable version of the PC algorithm and causally ordered variables, the SHD decreases significantly when using the stable PC algorithm with majority rule compared to the unstable version with permuted variables. It thus seems that the more restrictive way to introduce v-structures truely reduces mistakes.

Theoretically, the conservative version of the PC algorithm differs from majority rule version in that a strictly more stringent criterion has to hold in order for a chain to be directed as a v-structure. The direct consequence of this is that fewer edges are directed. However, the most prominent difference between the results of the two versions is that restricting the formation of v-structures results in a significant reduction of conflict edges for sufficiently high values of s (see Figure 4.17). This effect seems to outweigh the immediate effect that fewer directed edges are created: For $s = 1000$ and greater, the conservative version created *more* directed edges, presumably because many of the directed edges created according to the majority rule turn into conflict edges. We can see that for around $s = 1000$, the number of conflict edges produced by the majority rule version increases strongly. In the same range, the SHD changes from being lower for the majority rule version to being lower for the conservative version (see Figure 4.20), most likely for the same reason.

By more systematically avoiding conflict edges, the conservative version is also able to keep the rate of PDAGs and extendable graphs in the output higher, for larger s. Nevertheless, for very large sample sizes, also with the conservative version both fractions drop to zero if the true graph is dense ($d = 5$). For very sparse graphs ($d = 2$), however, both fractions are comparatively high for both the conservative and majority rule version, hardly ever dropping below 50%.

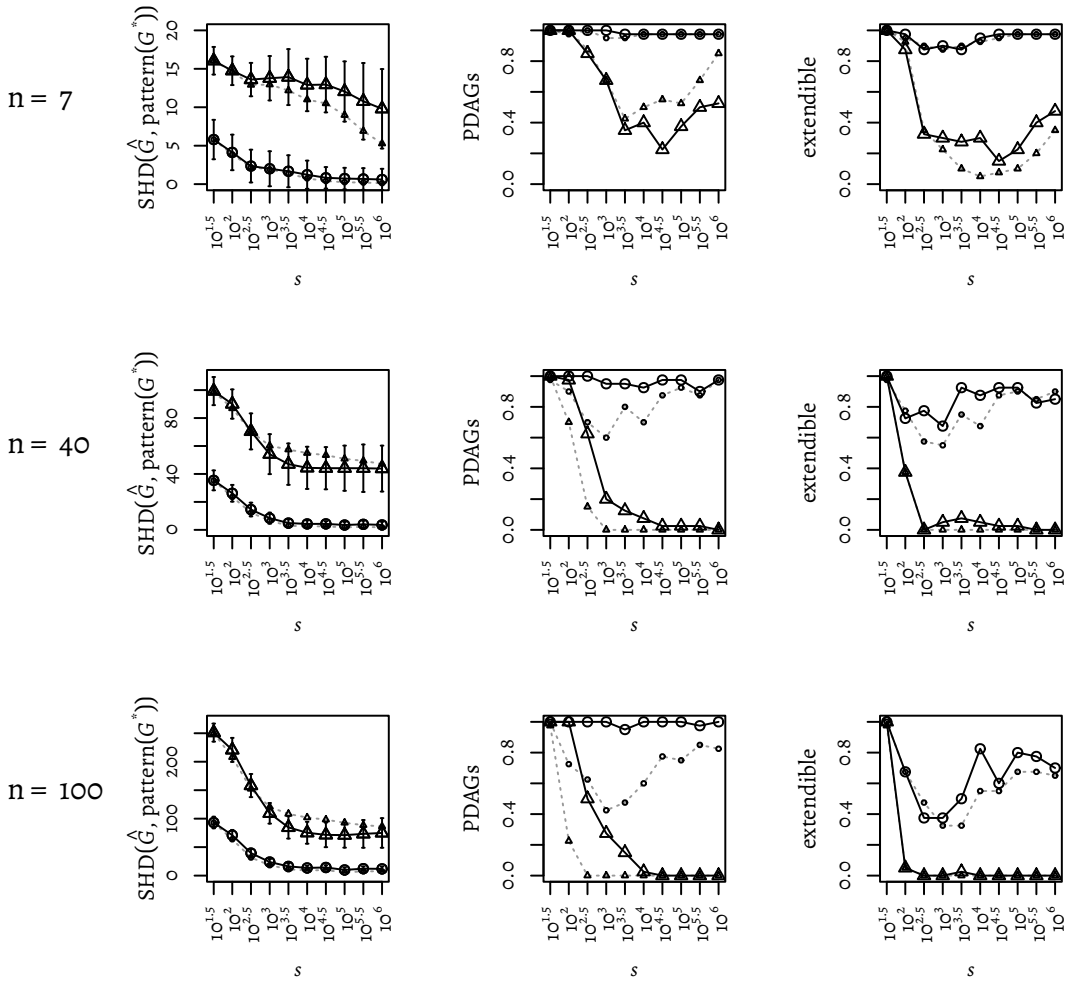


Figure 4.20: Structural Hamming distance to the pattern of the true DAG as well as fractions of graphs that were proper PDAGs or extendable for the graphs estimated by the *stable conservative* version of the pc algorithm. The measures are shown dependent on the sample size s for graphs with $n \in \{7, 40, 100\}$ nodes. Data points marked with circles correspond to an expected degree of the true graph of $d = 2$, triangles to $d = 5$. Lines are added to improve the distinguishability of these two settings. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals. For comparison, faint background lines represent the results for the stable pc algorithm with *majority rule*. Meek's rules were not applied in both cases.

4.3.4 A Model for the Density of the Estimated Graph

It has been noticed before [148] that the tuning parameter α should be chosen depending on the number of samples in the data. However, in contrast to this finding, they choose a value of α and use it over a vast range of settings. To be precise, they determine a value

of α by optimising the average structural Hamming distance (SHD) to the true CPDAG for s between 30 and 30000. They find that overall, $\alpha = 0.005$ or $\alpha = 0.01$ yields the lowest average SHD to the respective true CPDAG.

The reason for the dependence of α on s is that both parameters influence the density of the output of the PC algorithm, that is how many edges it finds: The higher α, the more probable it is that the null hypothesis that two variables are conditionally independent is rejected. Thus, the higher α, the more edges are kept.

On the other hand, an increase of s affects the test statistic that we use. As described in the paragraph on Fisher's Z-test in Section 2.1.4 and 4.1.1, for some nodes X and Y and a set of node indices Z, we consider the p-value computed as

$$-2(\Phi(\sqrt{s - |Z| - 3} \cdot |\mathcal{Z}(X, Y \mid Z)|) + 2.$$

If this value is smaller than α, the null hypothesis is rejected and the corresponding edge is kept. Thus if s increases, also the p-value becomes smaller so that the null hypothesis is more likely to be rejected and more edges are kept, likewise leading to denser graphs.

This is consistent with basic principles of learning. If we have little data available, we should prefer a simple model that is able to explain the data because otherwise, we would probably overfit. If there is a large amount of data available, which suggests that the model has some specific features, we should be more inclined to believe in those features. According to the law of large numbers, a certain feature in the data is more probable to be significant when observed in a large than in a small sample. In the latter case, it is more probable to be an artefact or characteristic of the sample rather than the population. This issue is also discussed in [158, e. g. Section 17.3.1]

To understand how the density $\hat{d}$ of the estimated graph depends one the other parameters, let us consider the relationship between $\hat{d}$, graph sizes n, the density $\hat{d}$ of the estimated graph, the sample size s, and the tuning parameter α.

For a range of values of α and s, I executed r repetitions of `supervised_pc` (see Listing 4.6) (using the original, unstable version of the PC algorithm with permuted variables) and recorded the average degree of the graphs obtained by the PC algorithm. As an example, the result for graphs with $n = 40$ nodes and an expected degree $d = 2.5$ of the randomly generated true graphs can be seen in Figure 4.21. Heatmaps similar to the one shown in Figure 4.21 can be obtained for varying n and d. A change in these parameters only shifts the values in the matrix but does not affect the overall pattern. It turns out that these heatmaps of values of $\hat{d}$ have an approximately linear colour shading when alpha grows doubly exponentially, that is when $\log(-\log(\alpha))$ increases linearly, and the scale of s is logarithmic, i. e. $\log(s)$ grows linearly.

On top of that, the matrix in Figure 4.21 grants us some interesting insights. If we assume that the estimated graph resembles the true graph more if their densities are more similar, then we could reason based on Figure 4.21 that α should be chosen smaller the higher s is. However, the measures described in described in Section 3.6.2, for example the structural Hamming distance more suitable to assess the estimation accuracy. A corresponding tuning procedure is described in Section 5.3.2.

To quantify the influence of the aforementioned parameters on the density of the output graph, I conducted a multiple linear regression of the density $\hat{d}$ of the estimated

graph on n, d, s and α. For the regression, I considered all combinations of the following parameter values for the supervised pc, and determined the average degree $\hat{d}$ of the estimated graph over $r = 100$ repetitions:

$$\begin{aligned} n &\in \{10, 20, \dots, 70\} \\ d &\in \{2, 2.5, 3, 3.5, 4, 5\} \\ s &\in 10^{e_s}, \text{ for } e_s \in \{1.5, 1.75, \dots, 4.5\} \\ \alpha &\in 10^{-10^{e_\alpha}} \text{ for } e_\alpha \in \{2.5, 2.4, \dots, -0.5\} \end{aligned} \tag{4.2}$$

Because of the finding that comparable values of $\hat{d}$ appear on a line when the values for α grow doubly exponentially and the values for s exponentially, I transform α and s accordingly and then use linear regression to fit the parameters in

$$\hat{d} = a_n \cdot n + a_d \cdot d + a_s \cdot \log_{10} s + a_\alpha \cdot \log_{10}(-\log_{10}(\alpha)) + b.$$

This yields the following best-fitting coefficients:

$$\begin{aligned} a_n &= 0.0045, \\ a_d &= 0.11, \\ a_s &= 0.85, \\ a_\alpha &= -1.22, \\ \text{and the intercept} & \\ b &= 0.11. \end{aligned} \tag{4.3}$$

The R^2-value of the fit is 0.82. Note that $\log_{10}(-\log_{10}(\alpha))$ decreases with increasing α, so despite the negative sign of a_α, $\hat{d}$ increases with all the regarded parameters.

To compare the strength of the regression coefficients, let us consider the overall increase in $\hat{d}$ that a change from the lowest to the highest value of this parameters would induce according to the regression. For example, if n is increased by 60, from $n = 10$ to $n = 70$ (keeping the values for all the other parameters constant), this would increase $\hat{d}$ by about $0.0045 \cdot 60 = 0.27$. Analogously, changing $d = 2$ to $d = 5$ yields an increase of $\hat{d}$ of about $0.11 \cdot 3 = 0.33$; increasing the sample size from $s = 32$ to $s = 31623$ adds about $0.85 \cdot (\log_{10}(31623) - \log_{10}(32)) \approx 2.55$ to $\hat{d}$, and when α is increased from its minimum, $6 \cdot 10^{-317}$ to its maximum, 0.48, $\hat{d}$ would likewise increase by $-1.22 \cdot (\log_{10}(-\log_{10}(0.48))) - \log_{10}(-\log_{10}(6 \cdot 10^{317})) \approx 3.66$.

In this sense, under the assumption that the ranges that were considered for all parameters to the same extent cover a range that is realistic in practice, s and α have the strongest effect on the density of the pc-estimated graph.

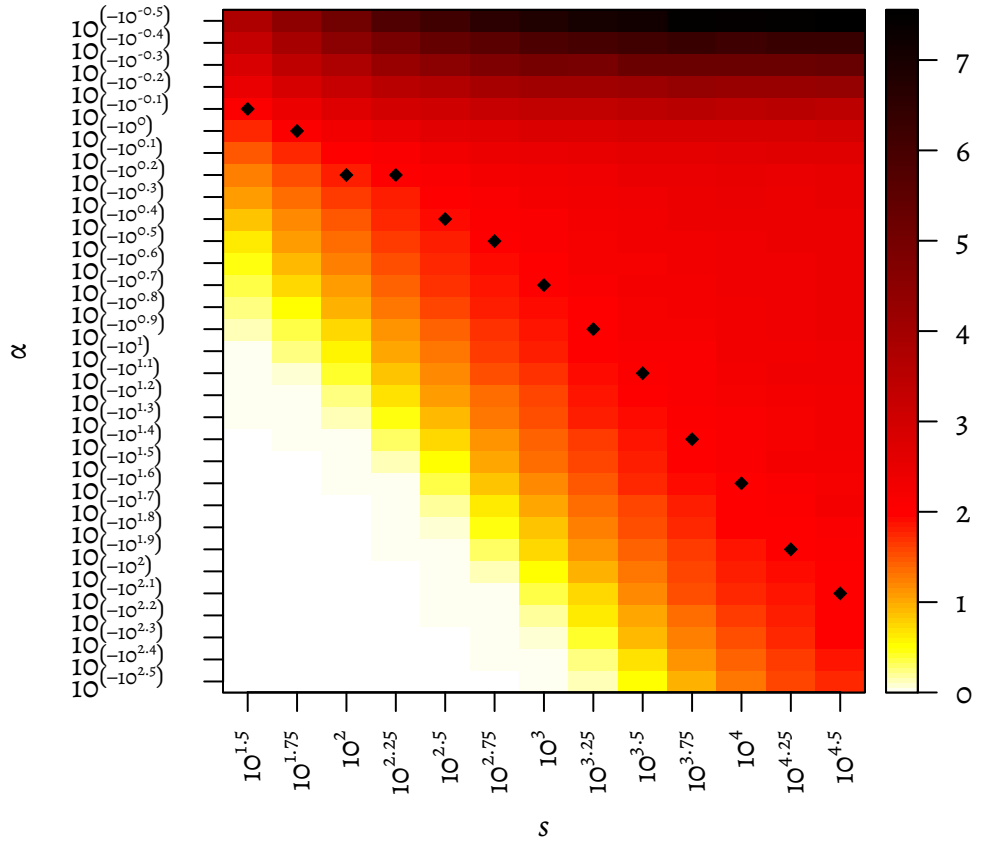


Figure 4.21: Density $\hat{d}$ of the graph computed by the PC algorithm depending on the tuning parameter α and the sample size s. The graphs from which the data were sampled have $n = 40$ nodes and an expected degree of $d = 2$. Accordingly, the value 2 corresponds to red, smaller values to lighter, higher values to darker colours. In each column the cell whose value is closest to the expected degree of the true graph ($d = 2$) is marked with a black diamond.

4.3.5 Experiments with Constant Density

Based on the data presented in Figure 4.21 it is possible to vary the sample size while holding the density $\hat{d}$ of the estimated graph approximately constant, and thus to verify that some effects for increasing s that we encountered earlier most probably are caused by the increasing density of the resulting graph.

To this end, the "optimal" value for α, referred to as $\hat{\alpha}$, is chosen for each combination of values of the other parameters as the one for which the target density d of the true graph is most accurately reproduced as the density $\hat{d}$ of the estimated graph. (These values $\hat{\alpha}$ are marked as black dots in Figure 4.21.)

The figure shows that for every s there is a range of α for which the desired value of d is well reproduced are many cells that reproduce the desired value of $\hat{d}$ very well – namely the red regions that fill most cells, especially for large values of s. Thus, to robustify the computation of the values $\hat{\alpha}$, they were chosen as the median of all α for which the resulting $\hat{d}$ deviates at most by some tolerance threshold τ from d. Here, this tolerance is chosen to be $\tau = 0.2$.

For the following analyses, I generated specific models for $n = 7$ and $n = 40$, us a linear regression of the "optimal" α. Since for larger densities d of the true graph the largest value of α that was regarded in the previous section is often optimal for a wide range of low values of s, I also extended the range of α and considered values $\alpha \in 10^{-10^{e_\alpha}}$ for $e_\alpha \in \{2.5, 2.3, \dots, -2.3, -2.5\}$, that is, $\alpha \in \{6 \cdot 10^{-317}, 3 \cdot 10^{-200}, \dots, 0.98, 0.99\}$.

Due to the high maximum values for α this analysis would take very long for $n = 40$ and is not feasible for $n = 100$, so it is only shown for $n = 7$ here. The regression yields the following "optimal" values of α:

$$\hat{\alpha} = 10^{-10^{a_s \cdot \log_{10} s + b}}, \quad \text{with} \quad \begin{cases} a_s = 0.37, b = -1 & \text{for } n = 7, d = 2, \\ a_s = 0.42, b = -2.6 & \text{for } n = 7, d = 5. \end{cases}$$

Results

It can be seen in the top left panel Figure 4.22 that the expected density of the true graph is quite accurately reproduced with the above model. While we have seen the accuracy of the pc algorithm decrease with s according to some measures in the previous section, this is not the case when the density is held constant. Firstly, the number of conflict edges in the estimated graph (top row, fourth panel) now decreases with s, while the number of graphs that do not contain any conflict edges increases (fifth panel). Except for a slight increase entailed by the decrease of conflict edges, the numbers of directed and undirected edges (second and third panel) are more or less constant with s.

Also the false positive rate (fpr, top row, first panel) and true detection rate (tdr, second panel) now clearly converge to the optimal values. Nevertheless, no significant change can be seen in the structural Hamming distance (third panel).

As hypothesised above, the increase of the number of conditional independence tests that we encountered in Figures 4.11 and 4.10 is actually due to the increasing density, because it does not occur here (fourth panel). Also the maximum order of a conditional test that is reached ($k_{\max}$, fifth panel) is now constant with s.

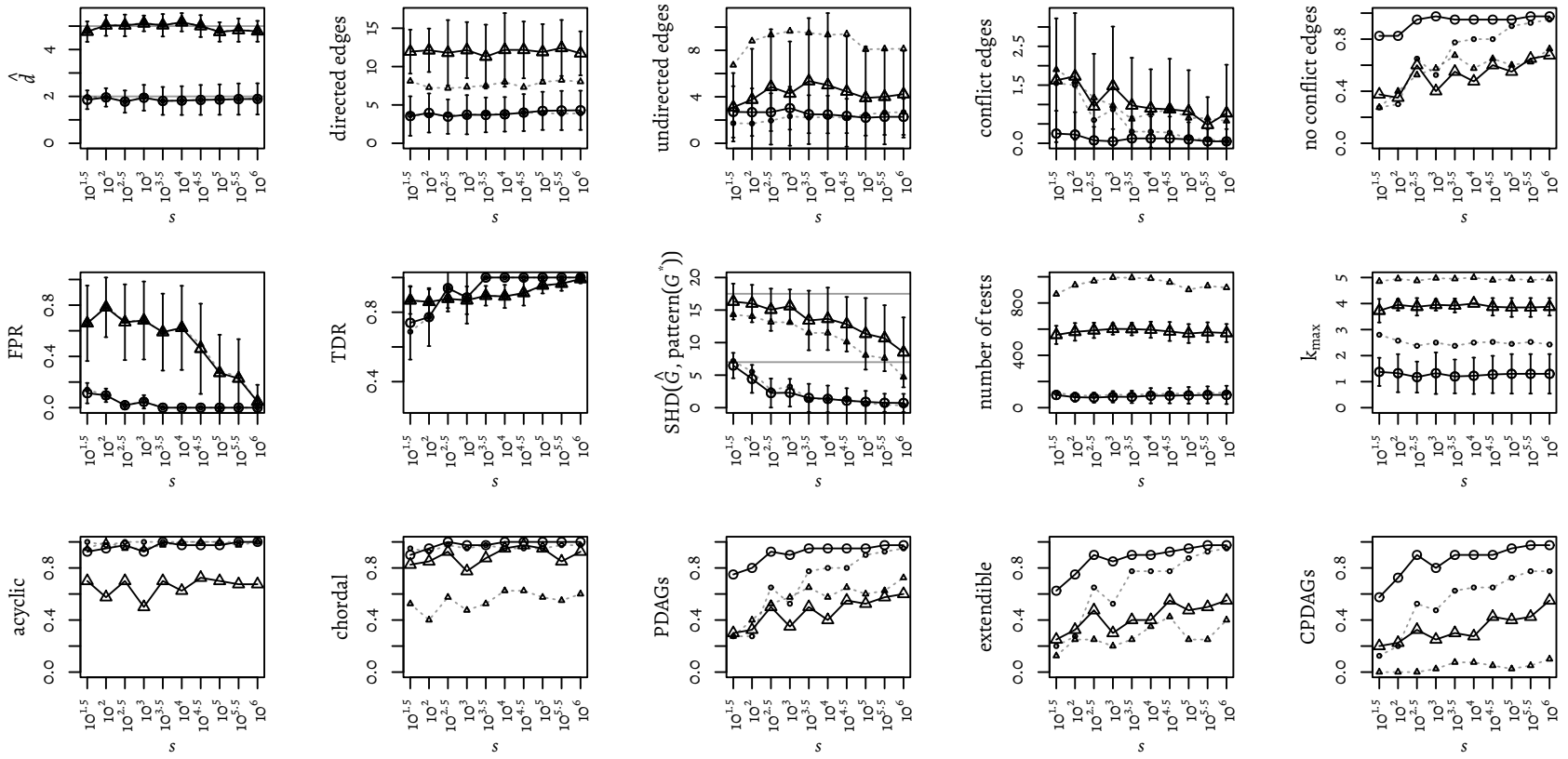


Figure 4.22: Selected measures for the accuracy of the PC algorithm depend on the sample size s when the density of the estimated graph is keep constant through the choice of α, so that the density of the estimated graph quite accurately corresponds to the expected degree of the true graph, $d = 2$ for data points marked with circles and $d = 5$ for data points marked with triangles (see the top left plot). Solid lines and larger data points correspond to the stub version of the PC algorithm using the majority rule, faint lines and smaller data points represent the results for the original version of the PC algorithm with conflict detection. Meek's rules were not applied in both cases. For all plots, the number of variables is $n = 7$. Data points are computed as the mean over $r = 40$ replications and the error bars represent 95% confidence intervals.

Furthermore, measures that describe the validity of the output graph as a causal structure now converge or remain constant. To be precise, independent of s about 60% of the denser graphs ($d = 5$) and nearly all of the sparser graphs ($d = 2$) are acyclic (bottom row, first panel), and, at least for the stable version of the pc algorithm, nearly all estimated graphs are chordal (second panel). The fractions of pdags, extendible pdags and cpdags all grow towards 1 with increasing sample size, although it can be seen that the graph classes are gradually more restricted. This will be analysed in more detail in the next chapter.

4.4 Conclusion

In Figure 2 in [148], the authors compare the pc-estimated graph with the true graph and report the result in terms of the true and false positive rate (tpr and fpr) of the edges as well as the structural Hamming distance (shd) for the cpdag. They find that all these measures converge, especially for sparse graphs.

Despite the positive conclusions that can be drawn from this, one can also see some limitations in their simulation results. For one thing, the shd is still quite high. We can see in Figure 4.7 that it is not uncommon that the shd is a third or quarter of the number of edges in the graph. In particular, this means that if an algorithm would always return the empty graph, it would still just be about three or four times as far from the truth as the pc algorithm is. Furthermore, the fpr is certainly small, but the denominator of this quantity, the number of non-existent edges in a graph, is usually huge. Therefore, this measure might not be very informative, especially for dense graphs.

In addition to the results by Kalisch and Bühlmann, a number of other measures of the estimated graph have been studied in this section, also in variations of the pc algorithm. We have seen that the fraction of the graphs estimated by the pc algorithm that are actually pdags is small, even when conflicts are not detected but randomly resolved. This finding challenges the justification of phase three of the pc algorithm, the application of Meek's rules, which are designed to turn a valid pdag into a cpdag. Furthermore, the ordering in which the variables are passed to the pc algorithm has a strong influence on the estimated graph and ought to have positively influenced the data shown by Kalisch and Bühlmann [148].

The stable version of the pc algorithm is able to remedy its order-independence. However, some problems remain, for example the poor extendability of the resulting graphs and the small fraction of cpdags associated with this. On top of that, a new issue arises with the introduction of conflict edges. Their introduction makes problems explicit, but also necessitates their solution, which is one of the main topics of the next chapter.

5

Improvements to the PC algorithm

In a perfect world, the causal structure learning process would consist in running the PC algorithm on the given data once and obtaining the correct CPDAG. Unfortunately several sticking points exist in practice.

First of all, there are many configurations and parameters in the computation that determine the outcome: How the data are computed, cleaned up, and pre-processed? Which version of the PC algorithm is used – e.g. the stable version? Which conditional independence test is used? How are conflicts being handled internally? – and so on. Above all, the main tuning parameter of the PC algorithm, the level of significance α of the individual conditional independence tests that are conducted to determine the skeleton, must be set to an appropriate value.

While some of the decisions can be made by an experienced researcher based on the type and properties of the data used, for example the choice of an appropriate conditional independence test, the value of α is hard to estimate. Since in the decision that is to be made by the statistical test – edge or no edge – false positives and false negatives are more or less equally inadequate, the parameter α should not be as small as possible, as is typical in clinical studies where significant effects, deviations from the usual, are to be found (see the discussion in Section 4.1.1). In our use case, it is instead advisable to find a good balance between the error probabilities α and β. How this can be done is further discussed in Section 5.3.3.

Parameter tuning is usually done with the objective to improve the accuracy of the estimated graph. However, even more problematic than incorrect edges are cases where the overall graph is not valid, that is not a CPDAG.

According to our findings from the previous chapter, in the sample case the estimated graph is scarcely ever a CPDAG, and, what is more, often not even a PDAG (see Figure 4.12). In this chapter, we will investigate this issue further.

Section 5.1 is about the different "symptoms" of invalid DAGs, that is, in which regard a PC-estimated graph can be invalid. Section 5.2 gives examples how this can happen and how this happens. In Section 5.3, methods for tuning α are revised and new approaches are introduced. After the parameter tuning procedure, one will have found a graph that represents the causal relationships in the data as good as possible, but still might not be extendible and contain some conflict edges. In Section 5.4, a method to heuristically extend such a graph is presented, based on a new and efficient algorithm for determining a consistent extension of a graph that contains conflict edges.

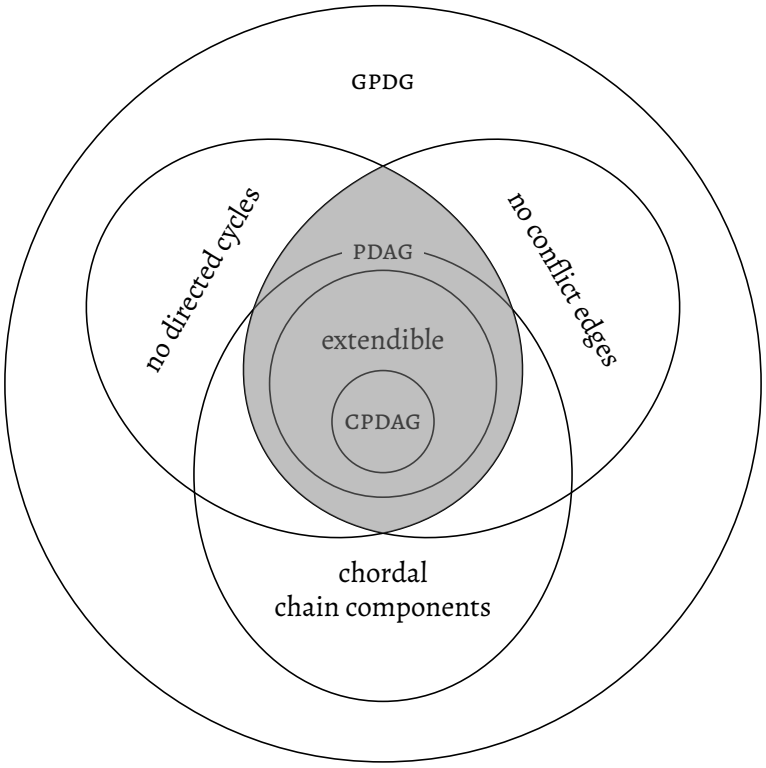


Figure 5.1: Different classes and properties of graphs that are relevant for the characterisation of the output of the PC algorithm.

5.1 Symptomatology

In the previous chapter, we have identified graph classes that in general do not characterise the output of the PC algorithm in the sample setting: the class of CPDAGs, the class of extendable graphs and the set of PDAGs. Also in this chapter, we will consider the PC algorithm in the sample setting and have a closer look at why the PC-estimated graph may belong to none of these classes.

Therefore, let us introduce a new class that properly characterises the output of the stable version of the PC algorithm– the class of general partially directed graphs:

Definition 13 *(General partially directed graphs (GPDGs))*.
A general partially directed graph is a graph that contains directed, undirected and conflict edges.

In this section, will study this class and its subclasses, including the above mentioned classes off PDAGs and CPDAGs. Their relationship is summarised in the Euler diagram in Figure 5.1. The symptoms that are relevant for the classification are considered in more detail in the next section. In particular, conflict edges are characterised in Section 5.2.1.

In Section 2.2.1, PDAGs were defined to be partially directed graphs without directed cycles. Furthermore, a partially directed graph is a graph that may contain directed and

undirected edges but no other edge types. All in all, in order to be a PDAG, the PC-estimated graph needs to be acyclic (referring to *directed* cycles) and free of conflict edges and thus lie in the grey area in Figure 5.1.

In Definition 10, the property of extendability is defined for PDAGs, such that the set of extendible PDAGs is a proper subset of the class of PDAGs. (After all, it is obvious that every graph that contains a directed cycle is not extendable to a DAG in the sense of Definition 10. Every non-PDAG is thus non-extendable.) Note that not every PDAG is extendible, because it might be impossible to direct the undirected edges without creating new directed cycles or v-structures.

A well-known characterisation of extendible PDAGs is that their undirected components are chordal [79]. The undirected components of a graph $\mathcal{G}$ are the connected components of the graph $\mathcal{G}'$ that results when all directed edges are removed from $\mathcal{G}$.

As can be seen in the Figure 5.1, the set of extendable PDAGs thus lies in the cut of

1. the set of graphs with no conflict edges
2. the set of graphs with no directed cycles
3. the set of graphs with chordal undirected components.

Representing a Markov equivalence class, each CPDAG is extendable to each DAG it represents. On the other hand, not every extendible PDAG is a CPDAG. For example, the PDAG $X \to Y - Z$ is extendible to $X \to Y \to Z$, but it is not a CPDAG since $Y \to Z$ is the only possible way to direct $Y - Z$. The set of CPDAGs thus is a proper subset of the set of extendible PDAGs.

To conclude, the PC-estimated graph $\hat{\mathcal{G}}$ might not be extendable, and thus not represent a DAG and a corresponding SEM for the following reasons. Firstly, it might not be a PDAG because it

1. contains conflict edges.
2. contains directed cycles.

Even if $\hat{\mathcal{G}}$ is a PDAG, it might not be extendible, in particular because

3. its undirected components are not chordal, but also because
4. its undirected components are chordal, but it is not extendible anyway.

These individual "symptoms" of non-extendability are further discussed below.

In this thesis, we are mainly concerned with the problem of non-extendability of the PC-estimated graph. However, the usual aim when applying the PC algorithm is to learn a CPDAG [148, 122, 52, 129]. Nevertheless, obtaining an extendable graph is an important first step in this endeavour. Certainly, a CPDAG is best suited to represent the set of all (indistinguishable) DAGs which describe the processes that have generated the data. However, to obtain a consistent estimation for one such DAG, it suffices to learn an extendable PDAG. If desired, such a PDAG can in principle be turned into a CPDAG via a consistent extension as described in [43].

All in all, regardless of whether the graph returned by the PC algorithm is the true DAG or not, it might not even be a *valid* solution to the causal question that is to be answered. In this case, it is not suitable for typical causal inference methods, like for example, IDA [172] or Pearl's intervention calculus [200] on which IDA is based.

We are therefore in need of a well-founded procedure to tune the parameters of the PC algorithm in order to achieve meaningful results, and transform these results into proper graphs of certain kinds, e.g. CPDAGs, without loosing too much information. Since such issues have not been profoundly described in the literature, this thesis aims at shedding some light on this part of the process of causal structure learning with the PC algorithm.

5.2
Reasons for Errors

We have seen in the previous section that the main problems concerning PC estimated graphs are conflict edges, cyclicity and non-extendability.

As a first step towards remedying them, in this section we will look into how these symptoms come about and why. We will consider which violations of the assumptions or specific errors in the computation can provoke individual symptoms.

However, it is important to not interpret symptoms like conflict edges or acyclicity as direct causes of latent confounders or cyclic relationships in the data. They are rather the result of complex combinations of statistical errors that *in part* can be due to violations of the assumptions, including acyclicity, causal sufficiency and unfaithfulness, but also non-Gaussianity. On top of that, the true DAG might be to dense to be consistently learned or the sample size might be too small.

5.2.1
Conflict Edges

As described above, conflict edges emerge when v-structures overlap in an inconsistent way. A simple example for this is depicted in Figure 5.2; if in the skeleton to the left, both $V \to W \leftarrow X$ and $W \to X \leftarrow Y$ are identified as v-structures, the edge connecting W and X will have to be directed as $X \to Y$ *and* $W \leftarrow X$, and thus be identified as a conflict edge. In this section, we will investigate under which circumstances such a contradiction of two v-structures occurs. Later, in Section 5.4, a method to resolve them is proposed.

Figure 5.2: An typical example for how a v-structure can arise in the PC algorithm with conflice detection.

How Conflict Edges Emerge in the PC Algorithm

In the original description of the PC algorithm by Spirtes et al. [263], it is not specified explicitly how conflicts in step two and three of the PC algorithm should be solved. In fact, it seems that the insertion of directed edges should be done iteratively. Hence for example in Figure 5.2 the v-structure that would be identified first would prevail, and the other one would no be identified at all because only undirected edges are turned into v-structures. The resulting graph would thus be either $V \to W \leftarrow X - Y$ or $V - W \to X \leftarrow Y$.

In the standard implementation in the R package `pcalg`, that is when using the option `solve.confl = FALSE`, the behaviour is contrastive: only the adjacency information is used to determine v-structures, without requiring the edges to be undirected. Therefore "later" v-structures overwrite previously inserted ones, which results in either $V \to W \leftarrow X \leftarrow Y$ or $V \to W \to X \leftarrow Y$ in the example from Figure 5.2.

Only if `solve.confl = TRUE`, all v-structures are determined beforehand and conflicts are identified. All edges that are part of several v-structures which suggest contradictory directions for it are then marked as bi-directed edges. This is done by inserting the value 2 in both corresponding positions of the adjacency matrix in the internal representation. These edges are the ones that are referred to as conflict edges here. When using the option `solve.confl = TRUE`, also in step three of the pc algorithm all modifications are collected first. Then conflicts are identified and marked with bi-directed edges.

Also the (mutually exclusive) options `conservative` and `maj.rule` (see the description on page 99) influence the occurrence of conflict edges. The conservative version of the pc algorithm [219] identifies v-structure in a more restricted manner, and thus also leads to fewer conflicts between them. In Figure 4.17, it can be seen that also the majority rule option reduces the number of conflict edges produced significantly. This is probably because the voting procedure in fact helps to identify the true v-structures while avoiding false positives.

A possible explanation for this in turn is the finding in Section 5.3.3 that usual choices of α strongly underestimate the optimal values and that thus many dependences are classified as independent. This seems to be especially frequent when there exists a separating set similar to the one considered. Because of the algorithmic structure of the skeleton phase in the pc algorithm, such a similar set would often be a superset. In other words, choosing α too low might cause the found separating sets to generally be too small. Thus it is likely that when the condition for v-structure identification is tested for a chain $X - Y - Z$, the variable Y is not in the separating set S_{XZ} that was found in the skeleton phase although it should be. Considering all subsets of the neighbours of X and Z and deciding based on all of them resolves this bias.

For a complete example, consider again the path $X - Y - Z$. If X and Z are considered marginally independent, this would actually be false. Yet, it does not entail an error in the skeleton, since in the next round, when considering the separating set $\{Y\}$, the edge would have been removed anyway.

However, the subgraph $X - Y - Z$ is falsely classified as a v-structure when the naïve method for v-structure identification is used that considers only the separating set that lead to the edge removal.

On the other hand, using the majority voting procedure, all subsets of the neighbours of X and Z would be evaluated as potential separating sets (see the paragraph on the stable version of the pc algorithm on page 99). Thus, if the applied conditional independence test has a general bias towards independence, for example due to a low value of α, this would apply to separating sets with and without Y alike. If all of them would turn out to separate X and Z, there would be equally many with and without Y, so that the v-structure would be classified as ambiguous and not be directed as a v-structure.

The same holds with respect to the application of Meek's rules in the third phase: Meek [182] defines that, for example, the edge directed by rule 1 must be an undirected

edge. However, in the `pcalg` implementation the last direction of an edge overrides earlier orientations when `u2pd = "rand"` or `u2pd = "retry"` is used. Using the option `solve.confl = TRUE`, the information about ambiguous edges is carried over to the third step of the PC algorithm. In this case, edges that are undirected but part of an ambiguous v-structure are not directed as other undirected edges. For instance, Meek's first rule is not applied to an ambiguous chain $X \to Y - Z$, and thus the edge $Y - Z$ is not directed as usual. On top of that, new conflict edges are created in case of contradicting orientations obtained by the application of Meek's rules. For example, in the instance shown in Figure 4.15, the edge connecting X and Y would be identified as a conflict edge if `solve.confl = TRUE` was used.

All in all, conflict edges are always due to conflicting information about edge directions, which in turn is attributable to incompatible v-structures, at least indirectly. The options `conservative` and `maj.rule`, which help to limit the number of v-structures that are identified, thus also help to reduce the number of conflict edges that are found.

Conflict Edges Due to Latent Confounders

Similar to bi-directed edges in MAGs, conflict edges may reflect latent confounders, as shown in the example in Figure 5.3: Consider the true graph on the left, in which H is unobservable. Because W and X can not be separated conditioned on observable nodes only, the PC algorithm would estimate the skeleton shown in the middle of the figure. Now, because W and X are colliders in the true graph, the paths $V - W - H - X$ and $W - H - X - Y$ are blocked unconditionally, so that in particular $W \notin S_{XV}$ and $X \notin S_{W,Y}$. As a result, both $V \to W \leftarrow X$ and $W \to X \leftarrow Y$ are directed as v-structures, such that the edge between W and X becomes a conflict edge.

Figure 5.3: An example where a conflict edge is due to a latent confounder.

This interpretation of conflict edges has lead to the FCI algorithm, an extension of the PC algorithm for causally insufficient settings [262]. This more general approach, however, has two main disadvantages: On the one hand, it requires more complex computations and thus has a much larger running time, making it infeasible when the number of variables n is large. On the other hand, the relaxation of the causal sufficiency assumption renders the problem even more underdetermined so that the identifiability worsens. In particular, there are often excessively many bidirected edges in the output graph, reducing its causal expressiveness.

Conflict Edges That Are False Positive Edges

The example in Figure 5.3 can be generalised to the case where the unobservable variable H is replaced by an observable variable Z, but an edge between W and X is erroneously included in the skeleton, as shown in Figure 5.4. This is a simplified version of an example from Wienöbst and Liśkiewicz [310, Figure 2], where the edge $W - X$ is classified as *incompatible* which implies that it should not be part of the skeleton.

Due to the fact that W and X are colliders on every path on which they lie, W and Y as well as V and X are marginally independent. Thus, both $V - W - X$ and $W - X - Y$ are identified as v-structures, such that the edge $W - X$ becomes a conflict edge.

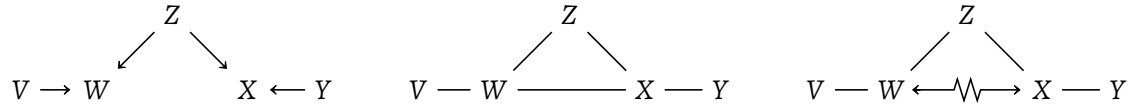


Figure 5.4: An example due to [310], where a false positive edge in the skeleton is identified as a conflict edge.

Conflict Edges Due to Unfaithfulness

As discussed in Section 3.2.1, the faithfulness assumption has been justified [287] with reference to the finding of Meek [185] that unfaithfulness is rare, in the sense that unfaithful distributions are of measure zero with respect to the Lebesque measure. However, this does not imply that unfaithfulness is not an issue in causal structure learning. In fact, Uhler et al. [291] show that because of the geometry of non-faithful distributions, "close-to-unfaithful" distributions can have very high measure. In the presence of sampling errors this is a very relevant finding.

An example for how unfaithfulness can give rise to conflict edges can be constructed from the thrombosis risk case described in Section 3.2.1: The contraceptive pill on the one hand increases the risk for thrombosis. On the other hand, it also decreases the chance of pregnancy, which in turn would increase the risk for thrombosis. If we introduce a variable X that mediates the former effect, we obtain the causal diagram shown in Figure 5.5, where C can be thought of as the contraceptive pill, P as pregnancy and T as thrombosis (although the diagram probably does not accurately reflect the medical aspects of the example).

We now assume that the paths $C \to X \to T$ and $C \to P \to T$ cancel exactly, so that $C \perp\!\!\!\perp T$ (unconditionally) in the distribution. Not that this contrasts the independences expressed via d-separation by the causal model because there are two non-blocked causal paths between C and T: $C \to X \to T$ and $C \to P \to T$. Due to the marginal independence in the distribution, the edge $C - T$ is removed already in round 0 of the pc algorithm, yielding $S_{CT} = \emptyset$. Provided that the right skeleton is estimated, this in turn implies that $C \to X \leftarrow T$ and $C \to P \leftarrow T$ are identified as v-structures (using the original method). Since additionally there is a v-structure $X \to T \leftarrow P$, the edges between X and T and P and T would become conflict edges. (If, on the other hand, the majority rule or conservative version of the v-structure identification would be used, $C - X - T$ and $C - P - T$ would not be classified as v-structures (although as ambiguous with the majority rule), and the true cpdag would result.)

To conclude, conflict edges express inconsistencies that occurred during the computation, which result in edges that according to the rules of causality must be directed in both one and the other direction. They can reflect violations of the assumptions about the data, like faithfulness and causal sufficiency. In most cases, however, they are caused

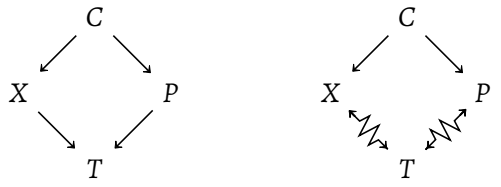


Figure 5.5: Example where conflict edges emerge due to non-faithfulness.

by statistical errors, and thus should not be interpreted causally (like, for example, bi-directed edges in MAGs that represent an unobserved confounder).

All in all, the number of conflict edges produced by the PC algorithm depends on different factors:

1. The data. If the data have not been generated according to a true DAG to which they are faithful, such a faithful DAG might not exist at all. Thus, the assumption that it can be learned from the data may lead to inconsistencies.
2. The test for conditional independence used. If the test does not fit to the data, it will give poor results which might entail conflicts.
3. The level of significance used. Generally, the higher the parameter α; the denser the graph and the denser the graph, the higher the probability of conflict edges. Furthermore, a badly tuned alpha might cause the conditional independence test to draw conflicting conclusions from the data.
4. The criterion for the constitution of v-structures: majority rule or conservative. We have seen theoretically, that these adaptations can improve the accuracy of the v-structure identification. This is significant also in practice: Figure 4.17 shows that in particular the conservative option is able to reduce the number of conflict edges significantly for dense graphs.

5.2.2 Cycles

Cycles in the estimated graph *may* reflect cyclic dependencies in the data generating processes as discussed in Section 3.2.3. However, cycles in the estimated graph can also form due to sampling errors.

An example where three erroneous conditional independence tests can cause a cycle to emerge can be constructed based on the true graph given on the left in Figure 5.6. For this, assume that in the round 0 of the skeleton phase of the PC algorithm all conditional independence tests yield the correct result except for two: Firstly, Z and X are deemed to be marginally independent, although for this the path $X - Y - Z$ would need to be blocked by conditioning on Y. Later on, this leads to the identification of a v-structure $X \to Y \leftarrow Z$ because $Y \notin S_{XZ}$. The second error in our example is that W and Y are not considered independent unconditionally, although they are, because the only path between them, $W \to X \leftarrow Y$, is blocked because X is a collider. If in the next round, W and Y are estimated to be independent conditioned on X and the v-structure $W \to X \leftarrow Y$ not identified (since $X \in S_{WY}$). Thus the graph on the right in Figure 5.6 would result, without any inconsistencies (like conflict edges) detected.

Note that the same result can be obtained with fewer errors with the naïve version of the pc algorithm (without the option `solve.confl = TRUE` in the `pcalg` implementation). Then only one error, namely considering Z and X to be marginally independent, can result in the cyclic result depending on the variable ordering. To be precise, the erroneous result is obtained if the v-structures $W \to X \leftarrow Y$ and $X \to Y \leftarrow Z$ are considered in this order, with the second one overriding the direction of the edge $X \leftarrow Y$ into $X \to Y$.

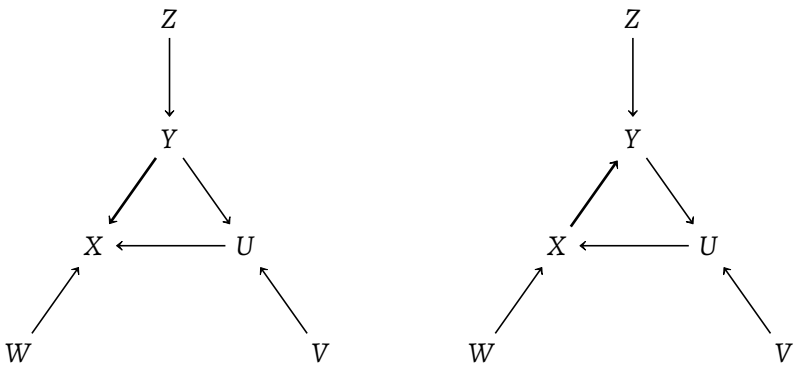


Figure 5.6: An example for how cycles can arise during the pc algorithm. For the graph on the left, three falsely assessed conditional independence queries, namely $Z \perp\!\!\!\perp X \mid \emptyset$, $W \not\perp\!\!\!\perp Y \mid \emptyset$ and $W \perp\!\!\!\perp y \mid \{X\}$, can cause the pc algorithm to learn the cyclic graph on the right.

5.2.3 Extendability

When the output of the pc algorithm does not represent a dag that might have generated the data, this formally is because the estimated graph is not extendable to a dag.

In the oracle model, the pc algorithm outputs a cpdag that represents the Markov equivalence class of the dag according to which the data were generated. This cpdag should in turn be extendable to all dags in the equivalence class. This means that if the estimated graph in the sample setting is not extendable, it does not represent a dag that might have generated the data. It is therefore crucial that the result of the causal structure learning is extendable.

The question whether a graph is extendable is closely related to chordality.

Definition 14.
A graph with chordal chain components is a graph in which no set of at least four vertices induces an undirected cycle. This means that every such subgraph that contains a cycle of length four or more also contains a chord, i. e. an edge connecting two non-subsequent nodes in the cycle.

Figure 5.7 illustrates that the extension of a chordless cycle of length four or longer must contain a v-structure, while a chord can serve as a shield to prevent the formation of a v-structure, thus permitting the formation of a (*shielded*) collider that breaks up the cycle. This entails that the so-called chain components of a cpdag, that is the components that are obtained by removing all the directed edges, are chordal graphs [11, 309]. However,

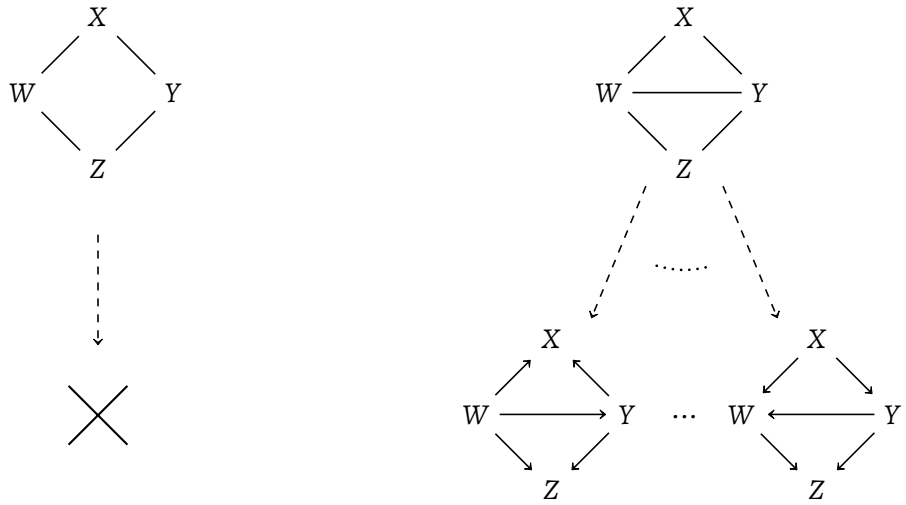


Figure 5.7: A cycle, which is not consistently extendable to a DAG, and a four cycle with a chord, along with two out of ten possible consistent extensions.

the connection is not biconditional: A PDAG with chordal chain components can be non-extendable, as the one in Figure 5.8.

$$A \longrightarrow B \longrightarrow C \longleftarrow D$$

Figure 5.8: A graph with chordal chain components which is non-extendable.

5.2.4 Accuracy of the Identification of v-Structures

Basically all the symptoms of errors that we identified in this section during the computation of the PC algorithm are due to conflicting edge directions. These may contradict directly – forming conflict edges– or indirectly – leading to non-extendability. Since edge directions are fundamentally based on v-structures, it suggests itself to evaluate the performance of the PC algorithm in identifying v-structures. For this, in turn, errors in the skeleton phase of the algorithm play a role.

Analogously to the evaluation of the estimated edges in the previous chapter, the true positive and true detection rate of this process are shown in Figure 5.9.

Here, the true positive rate is computed as the quotient of the numbers of correctly identified v-structures and v-structures in the true graph and, accordingly, the true detection rate is the fraction of true positive v-structures of the v-structures in the estimated pattern.

Regarding the false positive rate turned out problematic for edges due to the large number of non-existing edges in typical graphs. For v-structures, it makes even less sense to regard the "negative" ones, since the number of triples of three nodes that do *not* form a v-structure in the true graph is often even more disproportional.

For comparability, the same parameters as in the previous chapter were used: $n \in \{7, 40, 100\}$ nodes, an average degree of $d \in \{2, 5\}$ and $s \in \{10^{1.5}, 10^2, \dots, 10^{4.5}\}$ samples (without the extension to $s = 10^6$ as convergence is reached already for smaller values of s). The results in Figure 5.9 were produced for $\alpha = 0.01$, but do not differ significantly from those for other choices of $\alpha \in \{0.00001, 0.05, 0.1\}$. In addition, they were obtained using the stable version of the PC algorithm with the majority rule, but are similar when the conservative version or the original v-structure identification is used.

As in previous analyses, there is a fringe effect for small graphs in terms of nodes ($n = 7$), especially with few samples.

In accordance with the findings of Kalisch and Bühlmann [148], we can observe that the PC algorithm is much more successful in finding v-structures if the graph is very sparse (e. g. $d = 2$). In this case, however, even for very small values of α the algorithm still tends to find too many v-structures since the true positive rate is still close to one while the TDR is around 0.7.

For denser graphs ($d = 5$), the TDR is generally higher than the true positive rate, indicating that too few v-structures are found. (This effect does not show for $n = 7$, though, probably due to fringe effects.) In general, the high degree affects the number of potential v-structures in a super-linear way, since l directed edges with the same target node can form $\frac{l \cdot (l-1)}{2}$ v-structures. Accordingly, when the average degree and thus also the average in-degree of a graph is high, missing edges will on average cause even more missing v-structures. This may explain the comparatively steep decline of the true positive rate with increasing d. In particular, for a node with four ingoing edges (whose origins are not adjacent), there are six v-structures, but only 3 when one of the edges is missing. This illustrates how the true positive rate can easily drop to 50% with only small errors with respect to existence of edges.

All in all, the accuracy of the PC algorithm in identifying v-structures is much lower than the accuracy of the estimated skeleton, especially when the true graph is not very sparse. On the one hand, this might in part be due to error propagation, but it seems probable that the identification of v-structures is more error-prone in general. After all, in the synthetic data that were used, no assumptions like causal sufficiency are violated. The comparatively low true positive and detection rate that we encountered for denser settings ($d = 5$) can thus not be due to specific properties of the data, like latent confounders. This indicates that also the "symptoms" identified in the previous section, including conflict edges, might actually be due to sampling errors in most cases. How these can be reduced will be discussed in the following sections.

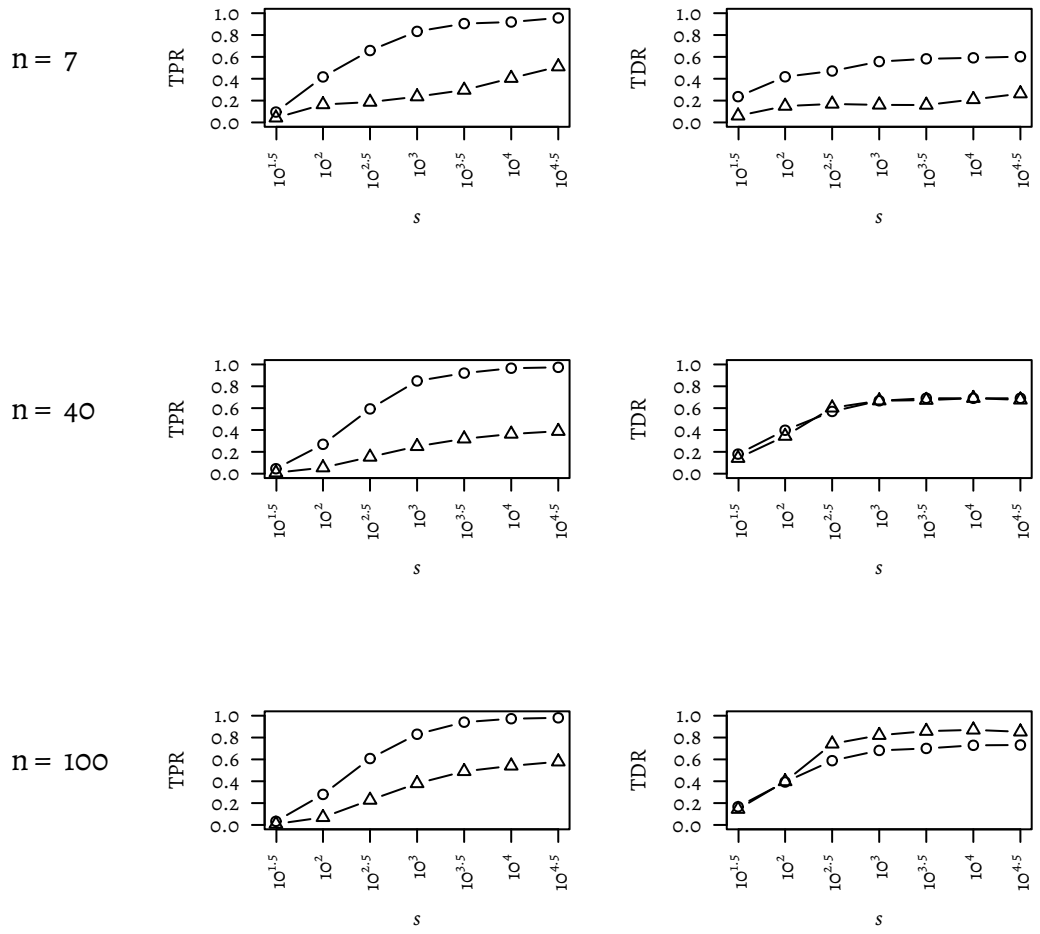


Figure 5.9: True positive rate and true detection rate of the identification of v-structures in the second phase of the PC algorithm dependent on the sample size s. The data were generated with the stable version using the majority rule and $\alpha = 0.01$, for $n \in \{7, 40, 100\}$ (plot rows). Circles int the plot correspond to $d = 2$, triangles to $d = 5$. Data points represent the mean over $r = 100$ (for $n = 7$) or $r = 20$ (for $n \in \{40, 100\}$) replications.

5.3 Optimisation of the Results of the pc Algorithm

Different strategies have been applied to improve the accuracy of the pc algorithm. On the one hand, it has been proposed to only or mainly consider the skeleton of the pc-estimated graph [148]. However, since most of the *causal* information is lost when the edge directions are discarded, for many applications this approach is as unsatisfactory as methods that randomly solve conflicts, like the implementation in the package `pcalg` when the options `u2pd = "rand"` or `u2pd = "retry"` are used.

Modifications that reduce the number of conflict edges in the output graph are more promising. Here, the stable version of the pc algorithm [52] is an important improvement.

If the assumption of Gaussian data is violated, non-parametric tests might solve many of the resulting problems. In particular, rpc [122] and copula-pc [63] have been devised for such settings. A different solution is applied by Aragam and Zhou [13], who apply a logarithm transform to their data which renders them "closer to Gaussian".

The sample size also plays an important role when it comes to the accuracy of the pc algorithm, as has been shown by Kalisch and Bühlmann [148] and further studied in Sections 4.3.2 and 4.3.3 of this thesis. For estimating the number of samples needed to learn with high probability a model that is "approximately" correct, Valiant [292] proposed the pac framework, which has been applied to the problem of causal structure learning [133, 101, 194, 319]. To work with small- or medium-sized samples, it has been proposed to improve the results by incorporating prior knowledge [31] or using permutation-based p-values [285].

Another option for the optimisation of the result produced by the pc algorithm is given in terms of the tuning parameter α. Furthermore, we have seen that the order-dependent version of the pc algorithm profits from being provided the data in the correct causal ordering of the variables. The idea to search the space of graphs obtained for different values of α and different variable ordering is realised in [67].

It is also possible to use the validity of the graph in the tuning procedure of the parameter α. For example, one could choose the highest value of α (from a set of candidate values) that yields a graph without conflict edges, an extendable graph or even a cpdag, although this might yield very sparse graphs. On the other hand, a well chosen value of α might reduce errors to such an extent that a comparatively dense graph with few conflict edges results, one that is (close to) extendible. For this reason, we will have a closer look at methods to tune α in this section.

Kalisch and Bühlmann [148], who popularised the pc algorithm for practical applications, derive a general estimate between $\alpha = 0.005$ and $\alpha = 0.01$ for the significance level α that should be used in the sample version of the pc algorithm in high-dimensional settings. They obtain this range by considering the structural Hamming distance (shd) averaged over different settings with $n \in \{7, 15, 40, 70, 100\}$ variables and a wide range of sample sizes $s \in \{30, 100, 300, 1000, 3000, 10000, 30000\}$.

However, as Malinsky and Danks [175] remark, "fixed-α conventions are not straightforwardly justified nor usually advisable in the model search context".

It is particularly surprising that Kalisch and Bühlmann determine α independent of the sample size. This is because, for proving the consistency of the pc algorithm, they

analytically derive an optimal value for the tuning parameter which is dependent on s, namely

$$\hat{\alpha} = 2(1 - \Phi(\sqrt{s} \cdot \frac{c_s}{2})). \tag{5.1}$$

However, this value for $\hat{\alpha}$ also depends on the unknown lower bound c_s of the partial correlations between variables this value is therefore not constructive. It does, however provide some interesting insights. On the one hand, α should decrease with s. For example, if would set $c_s = 0.1$ like the lower bound of the edge weights in the supervised PC, Equation (5.1) yields $\hat{\alpha} = 0.78$ for $s = 32$. This value decreases fast for larger sample sizes yielding values that decrease super-exponentially, yielding values that roughly follow the a scale that is used in Section 5.3.2 (see 5.2).

In this Section, I propose a refined procedure to choose α based on the optimum structural Hamming distance. It allows to predict a good value for α adaptively, *based on* the parameters of the setting, in particular the sample size.

On top of that, I studied the issue of finding adequate values for α from a more theoretical perspective by analysing the p-values that are considered during the course of the PC algorithm in a supervised setting. The results are presented in Section 5.3.3 and show that the optimal values for α in the individual conditional independence tests lie much higher than typical values chosen in the literature, e. g. $\alpha = 0.01$ [148, 171, 247, 54] or $\alpha = 0.05$ [233, 94] However, this is consistent with the finding of [122], that the largest value of α that they considered, $\alpha = 0.5$ yielded the best results. On top of that, the analysis presented here shows how hard it is to find the optimal value for α, even in the individual tests. This emphasises the difficulty of finding a well tuned α for the overall algorithm.

5.3.1 Previous Work on Parameter Tuning for the PC Algorithm

Koller and Friedman [158, Section 18.1.1] remark that it depends on the application whether it is better to choose a rather low value for α, thus obtaining more false negative edges (spurious independences), or a rather high value, such that more false positive edges (spurious dependences) arise. However, they also establish that – given a limited amount of data – it is in general better to learn a sparser graph, even if it is sparser than the true graph.

Malinsky and Danks [175] distinguish between two basic approaches to working with an unknown parameter.

1. Run the algorithm several times with different values of the parameter or different subsets of the data and extract the information that is stable over all iterations.
2. Systematically search the space of possible parameter values and choose the one that yields the best result according to some metric.

Analogously to the first approach, it is also possible to subsample the data (if the sample size permits it) and determine the parts of the causal structure that are consistently identified for all subsamples using a relatively high value for α [268].

The latter approach is pursued in Maathuis, Kalisch, and Bühlmann [172], applying the BIC score as a measure for the accuracy of the estimated graph. Using the BIC score is

quite common (e. g. [122]), although other measures for the suitability of the estimated graphs are possible.

Recently, Strobl [270] proposed different stability-based approach to parameter tuning. For each value α in some predefined set, he first computes the graph $\mathcal{G}_1$ that the standard pc algorithm yields. Then the procedure is repeated but the search for the separating set needed to remove an edge $X - Y$ is restricted to the neighbours of X or Y that *are also parents of X or Y, respectively, in* $\mathcal{G}_1$. The optimal α is then chosen to be the one for which the resulting graph most resembles $\mathcal{G}_1$ according to some metric, for example the structural Hamming distance or the structural intervention distance introduced in [212]. A Bayesian method for tuning α was proposed by [57]. Some analytical methods based on the p-values of the individual tests are mentioned in Section 5.3.3.

5.3.2 Parameter Tuning through a Fitted Model

In Section 4.3.4, a multiple regression model was derived that establishes a relation between the density $\hat{d}$ of the estimated graph an the other parameters of the supervised pc procedure shown in Listing 4.6: the number of variables n, the expected density d of the true graph and the sample size s.

A similar procedure can be conducted using the shd of the estimated graph to the cpdag of the true graph. Instead of learning how the shd depends on the above mentioned parameters, it is even more interesting to regard the value of α that yields the lowest shd in each case. More precisely, for each set of values for n, s and d, we want to find the value $\hat{\alpha}$ that among all regarded values of α minimises the shd of the resulting graph to the cpdag of the true graph.

Analogously to the case regarded in Section 4.3.4, for fixed values of n and d, the data can be represented in a heatmap like the one shown in Figure 5.10. The value $\hat{\alpha}$ then corresponds to the "brightest" cell in each column, and is marked by a dot in the figure. Similar to the values of the "best" density depicted as dots in Figure 4.21, also the values of $\hat{\alpha}$ roughly lie on a straight line (although not as perfectly). It thus suggests itself to determine a linear function that – given s and knowing n and d – computes the optimal value of α that should be used in the computation of the pc algorithm. After having collected data for heat maps as the one in Figure 4.21 for several values of n and d, these parameters can be included in a multiple linear regression.

We thus formulate a function to determine the *best* value of α depending on the number of variables n, s and the degree d of the true graph. Based on our finding that the degree of the estimated graph is about linear in $\log_{10} s$ and $\log_{10} - \log_{10} \alpha$, we assume this function to be of the form

$$\begin{aligned} \hat{\alpha} &= 10^{-10^{e_\alpha}}, \text{ where} \\ e_\alpha &= a'_n \cdot n + a'_d \cdot d + a'_s \cdot \log_{10}(s) + b' \end{aligned} \tag{5.2}$$

and estimate the parameters a'_n, a'_d, a'_s and b' by regression.

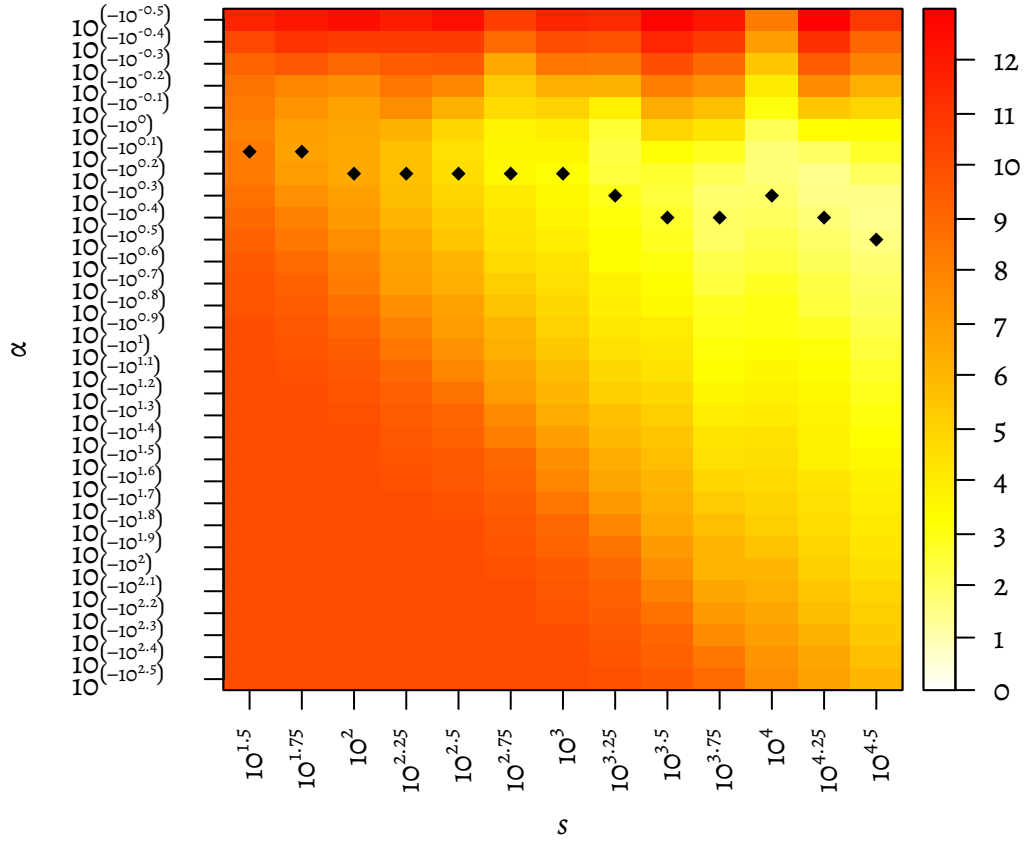


Figure 5.10: Structural Hamming Distances of the graph computed by the PC algorithm to the CPDAG of the true DAG depending on the parameters α and s. The graphs from which the data were sampled have $n = 60$ nodes and an expected degree of $d = 3$. In each column, the middle cell of the ones whose values are not more than $\tau = 1$ larger than the smallest value in the column is marked with a dot.

The data used for this were produced by considering all combinations of the following parameter values over $r = 100$ replications for $n \in \{10, 20, 30, 40\}$ and only $r = 20$ replications for the larger graph sizes:

$$\begin{aligned} &n \in \{10, 20, \dots, 60\} \\ &d \in \{2, 3, 4, 5\} \\ &s \in 10^{e_s}, \text{ for } e_s \in \{1.5, 1.75, \dots, 4.5\} \\ &\alpha \in 10^{-10^{e_\alpha}} \text{ for } e_\alpha \in \{2.5, 2.4, \dots, -0.5\} \end{aligned} \tag{5.3}$$

Again, we robustify our results by choosing the points, that the function is fitted to, as the median of a set of points fulfilling a certain criterion. In this case, the optimal point would be the one that corresponds to the minimal SHD. In the robust version the median of all points corresponding to an SHD that is no more than $\tau = 1$ larger than the optimal SHD is chosen.

Choosing $\tau = 1$, the resulting parameters are

$$\begin{aligned} a'_n &= 0.0034, \\ a'_d &= -0.081, \\ a'_s &= 0.10 \text{ and} \\ b' &= 0.08, \end{aligned} \tag{5.4}$$

which lead to a fit with an R^2 of 0.73.

In practice, Equation (5.2) can now be used with the parameters from Equation (5.4) to estimate a good value for the parameter α of the PC algorithm. This means that the chosen value of the exponent e_α of α should decrease with n and s and increase with d, so that α behaves inversely: It increases with d and decreases with n and s. If we compare the influence of the parameters as in Section 4.3.4 by multiplying the coefficient with the size of the range of the corresponding parameter, we find that changing d from $d = 2$ to $d = 5$ would decrease the exponent e_α by about 0.24, that is $\hat{\alpha}$ would increase two steps in the discrete scale that was used for it. (Note that due to the non-linear relation of α and the exponent e_α, the additive increase in $\hat{\alpha}$ depends on the initial value of α.) Analogously, $\hat{\alpha}$ would decrease by one and a half steps when $n = 10$ is changed to $n = 60$ and by more than three steps when s is increased from $s = 10^{1.5}$ to $s = 10^{4.5}$.

To confirm that the values are suitable, I ran a comparison study as follows. I repeated the identification of the optimal α by repeated experiments with different graphs, and in each setting determined the optimal α by linear search. For this I considered the following range of candidate values between about $8 \cdot 10^{-64}$ and 0.1:

$$\alpha \in 10^{-10^{e_\alpha}} \text{ for } e_\alpha \in \{1.8, 1.7, \dots, 0\}$$

Then for each setting with different values of the parameters $n \in \{7, 40, 75\}$, $d \in \{2, 2.5, \dots, 5\}$ and $s \in 10^{e_s}$, for $e_s \in \{1.5, 1.75, \dots, 6\}$, a metric for each value of α was evaluated as the mean over all $r = 10$ repetitions. For this, I considered both the SHD to the true CPDAG, which was also used to determine the model, and the F1 measure which is composed of the true positive and true detection rate (see Section 3.6.2).

The respective "optimal" values for α determined in this way are compared to the values given by the model in Figure 5.11 for a representative subset of true densities d.

Note that for this analysis it it not as crucial to cover also the limits of the scales for the parameters, in particular α, because fringe effects will not spoil the overall result (as in the regression). Instead it will possible to identify them and take them into consideration.

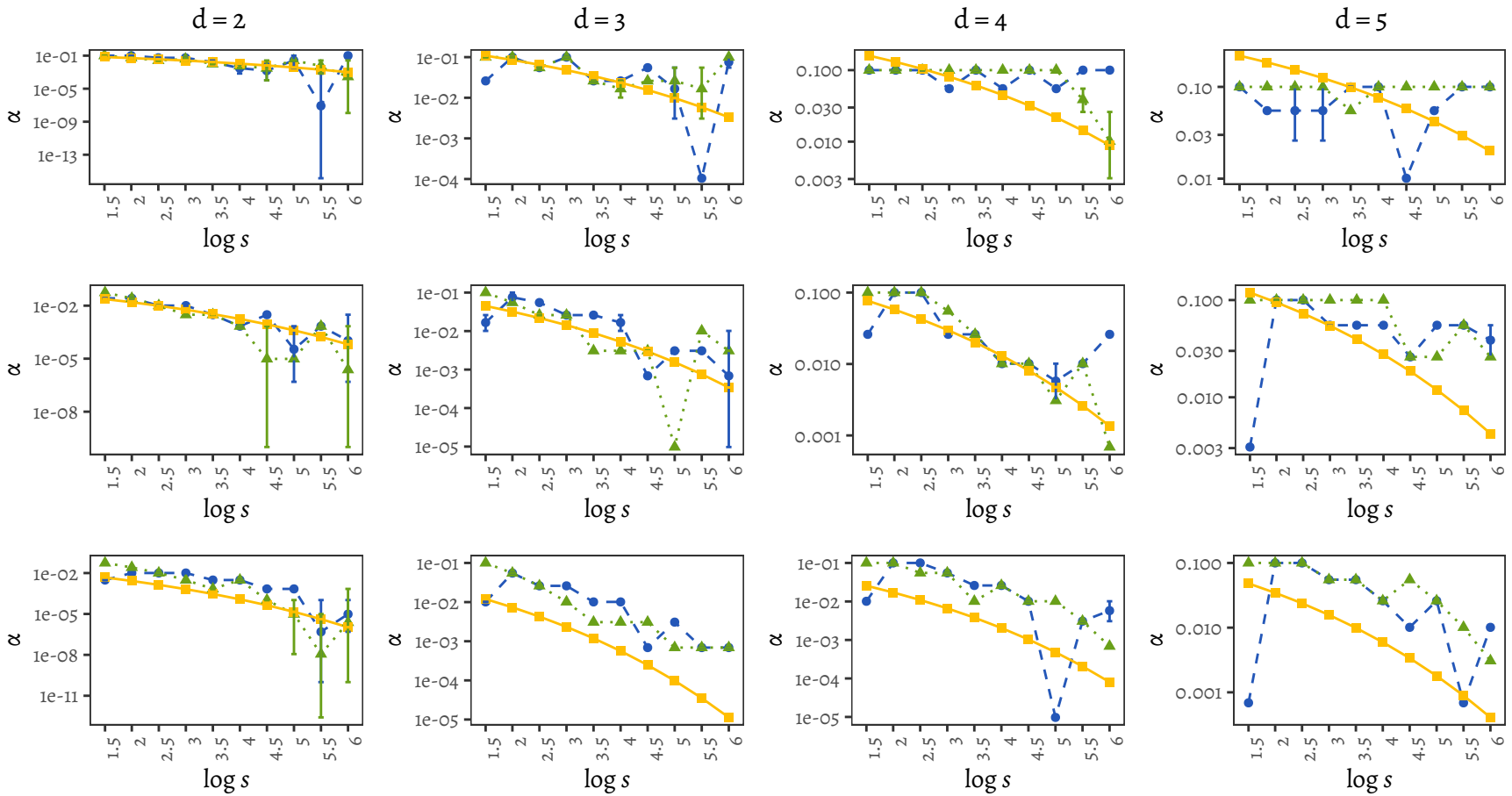


Figure 5.11: Comparison of the "optimal" values of α determined by the regression model (Equations 5.2 and 5.4, solid line) and a linear search for the optima of the SHD to the true CPDAG (dashed) and the F1 measure (dotted), dependent on the sample size s. Rows of the plot correspond to different graph sizes ($n \in \{7, 40, 75\}$, from top to bottom), columns to expected densities of sampled true graph ($d \in \{2, 3, 4, 5\}$, from left to right).

5.3.3 Analytical Identification of the Optimal Level of Significance

As discussed in Section 4.1.1, the parameter α is the level of significance for the *individual* conditional independence tests, but can not be seen as an overall level of significance. On top of that, the dichotomy "edge or no edge" is symmetric and thus does not comply with the concept of a level of significance. Rather, in the case of the pc algorithm, the parameter α should be chosen in such a way that a favourable balance between the two possible errors is attained: Erroneously keeping an edge (the type I error) or erroneously removing an edge (the type II error).

The first problem is called a multiple testing problem in statistics. There are general methods to solve this problem, for example Bonferroni adjustment and the Benjamini-Hochberg Method [301]. However, their applicability in the context of causal structure learning is limited. On the one hand, this is because the number of the conditional independence tests that the pc algorithm evaluates is hard to predict. On the other hand, the methods are based on the asymmetry of the statistical testing setting, which is not given in our case. Howev, Spirtes [257] remarks that if no further actions are taken, multiple testing might lead to overfitting.

To control the number of statistical errors, authors have considered bounding the false discovery rate by determining a suitable value for α [286, 166, 271]. A related method is the so-called p-value adjacency thresholding path [138, Section 3.2]. It estimates an optimal value for the tuning parameter α after one iteration of the pc algorithm, on the basis of the separating sets and maximal p-values found for each pair of variables.

Spirtes and Zhang [266, Section 18.5.1.2] discuss various methods for dealing with statistical errors, including methods for dealing with conflicting information. Such methods include the conservative version of the pc algorithm, as is theoretically motivated in [219]. Bromberg and Margaritis [30] propose a logical, so-called "argumentation" framework for working with conflicting conditional independence information that is based on the conditional independence axioms given in Equation (2.3).

In this section, we will further examine the second issue mentioned above, namely balancing the probabilities for type I and type II errors, α and β.

Analysis of Optimal Values for α

In the supervised case, that is if the true dag is known, it is possible to determine analytically which value for α is optimal to prevent mistakes. For this, we will have a look at the p-values produced by the conditional independence test that we want to use. Remember from Section 2.1.4 that the null hypothesis of conditional independence is rejected when p-value $<$ α. This means that p-values for true conditional dependences should be as small as possible (so that the null hypothesis is always rejected), while p-values for true independences should be large. In the supervised setting, we know the true state of the conditional (in)dependence from the d-separation relations in the true dag, and we can model the d-separation oracle as a test that always returns a p-value of 1 for independences and 0 for dependences.

Concerning a statistical test that we want to use in the sample version of the p-values, the objective is as follows: Find a value of α such that all p-values of true conditional independences returned by the test are above or equal to α, while all the other ones are

below α. Of course, this is only possible if such a threshold exists, which requires the p-values classified by the d-separation oracle to be linearly separable.

Figure 5.12 shows an example each for a separable and a non-separable set of labelled p-values. The data for this figure and the analyses in the remainder of this section were produced as follows. According to a randomly created graph, s samples of data were generated using the function `randomDAG` from the package `pcalg` (see Section 3.6.3). The data were used to assess the unconditional independence relations in the graph via a conditional independence test, in this case the `gaussCItest` from the package `pcalg`, which is a test for vanishing correlation based on Fisher's z-transform (see Section 2.1.4). The resulting p-values are sorted in ascending order yielding the sequence $(p_1, p_2, dots)$ (see the circles in the bottom panels of Figure 5.12). As the gold standard we consider the p-values computed by a d-separation oracle (implemented in the R function `dsepTest`, also from `pcalg`).

If a value for α is chosen, we interpret the relations with p-values above this threshold as independences and the ones below it as dependences; this classification can then be evaluated using the results of the d-separation oracle (shown as crosses in the bottom panels of the figure). The number of errors that result from choosing a value of α between two subsequent ones of the ordered p-values are shown in the top panels of the figure. The dependence of the number of errors on the p-value is referred to as the error function.

If the crosses in the bottom plot are linearly separable, the separation line corresponds to an interval for α in which the classification is perfect. In the example on the left in Figure 5.12 such a line could lie anywhere between the p-value for the independence query for nodes 4 and 5 and 3 and 6, yielding the interval between p_{min^1} and p_{max^0} (marked by black lines) for optimal values of α. Otherwise, it is interesting to find an interval with a low number of errors. In the setting on the right, there are two such intervals, marked by the minima of the error function. One of them corresponds to misclassifying the four rightmost true dependences, and the other one to misclassifying the two most "extreme" conditional independences *and* dependences, respectively.
We use the following measures to compare sets of p-values and draw conclusions about good choice for α. These are then computed p-values obtained for different parameters of the input graph (namely its number of nodes n and expected degree d), different sample sizes s and orders k of the conditional independence relations assessed by the test. All measures receive as an input a vector of p-values that are sorted in ascending order and are labelled with the true status of the (in)dependence relation as assessed by the d-separation oracle. The according values of this measures for the examples in Figure 5.12 are given in Table 5.13.

▷ Size of p-values The first quantity one can look at is the mean size of the p-values that the test produces. In particular, it is interesting to compare the mean size of true dependences and true independences. This gives an impression of whether it is possible to find a value for α to separate them.

▷ Separability This measure is a Boolean value answering the following question: Is there an interval (or a union of intervals) I such that for any $\alpha \in I$, the conditional independence test would not make errors in assessing the given conditional independence relations?

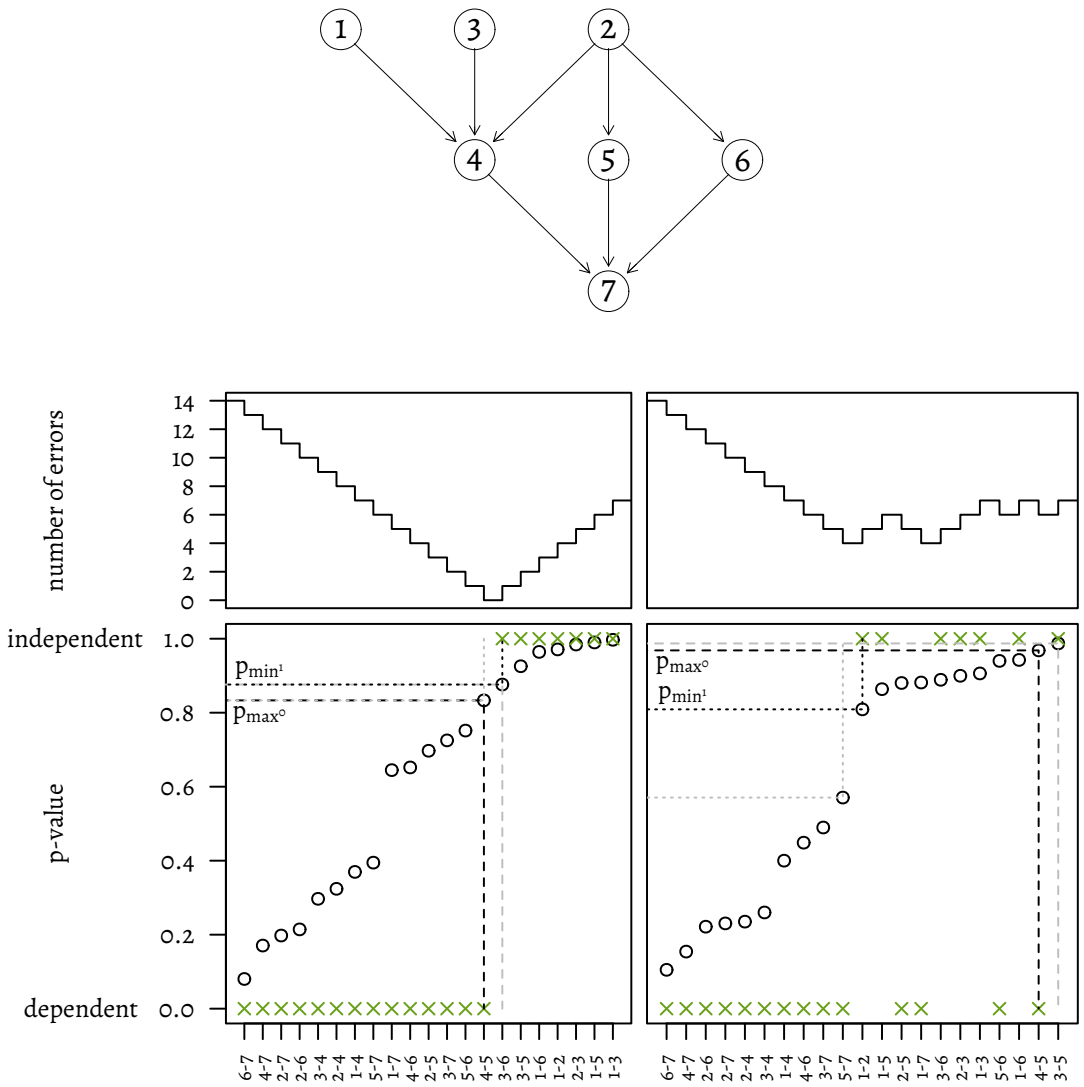


Figure 5.12: Example of the separability of p-values. On top, the randomly generated graph is given (with $n = 7, d = 2$); for simplicity of the notations the nodes are referred to by their indices in this example. For data with $s = 1000$ samples (left) and $s = 100$ samples (right), the resulting sorted p-values for conditional independences of order $k = 0$ are shown as circles in the lower panels, along with the desired p-values (returned by a d-separation oracle) as crosses. Above these, the respective number of errors that arise for a value of α between the p-values of subsequent indices in the ordering are shown. Note that the conditional (in)dependences (of the order $k = 0$) are denoted by X-Y instead of $X \perp\!\!\!\perp Y$. Additionally, the measures p_{min^1} (dotted black lines), p_{min^1-1} (dotted grey lines), p_{max^0} (dashed black lines) and p_{max^0+1} (dashed grey lines) are visualised. These and other measures are explained in the text. Their values for the examples shown can be found in Table 5.13.

measure	left instance in Figure 5.12, $s = 1000$	right instance in Figure 5.12, $s = 100$
separable	`true`	`false`
p_{max°	0.83	0.97
p_{min^1}	0.88	0.81
$p_{max^\circ+1}$	0.88	0.99
p_{min^1-1}	0.83	0.57
L_{CRIT}	0.042	0.42
L_{OPT}	0.042	0.25
$\min_{\text{err}}$	0	4
$\min^{\text{f}}_{\text{err}}$	0	0.19
$N_{\bar{\alpha}}$	1	2
$\bar{\alpha}_{\text{limits}}$	0.85	0.73

Table 5.13: Values of the measures defined in the text for the examples shown in Figure 5.12.

▷ p_{max° – VALUE OF α, SUCH THAT NO EDGES ARE FALSELY REMOVED Since in the PC algorithm removing edges is final, it is crucial not to erroneously remove an edge by choosing α too low. Therefore, it is interesting to regard the smallest value such that all true dependences have p-values below it. This value lies arbitrarily closely above the smallest p-value of a *true* conditional independence. We denote this p-value by p_{max°.

▷ p_{min^1} – VALUE OF α, SUCH THAT NO EDGES ARE FALSELY KEPT On the other hand, there are two reasons for tending to remove to many edges in case of doubt rather than too few, apart from the fact that making the graph sparse as soon as possible will improve the computational performance (confer Section 4.1.2). Firstly, conditional independence relations that are truly dependent given the conditional variables at hand but yield a high p-value often correspond to edges that should be removed given another separation set, possibly in a later round. That is, it is wrong to remove such edges in the current step, but right to remove them in general.

Secondly, the p-values of true independences are usually fairly high, so that the lower margin of reasonable values for α is usually quite distinct. This margin is given by the highest value of α such that all independences are classified correctly, which amounts to the smallest p-value of a true independence relation. This value is denoted by p_{min^1} here.

Note that, for separable instances, it holds that $p_{max^\circ} < p_{min^1}$, while otherwise, $p_{max^\circ} \geq p_{min^1}$. For an illustration of the two measures, see the black ledger lines in Figure 5.12.

▷ L_{CRIT}, p_{min^1-1}, $p_{max^\circ+1}$ – LENGTH AND BOUNDS OF THE CRITICAL RANGE FOR α For separable instances, p_{max° and p_{min^1} will correspond to two consecutive indices of conditional independence relations, and every alpha larger than p_{max° and smaller or equal to p_{min^1} will result in a correct estimation of all (in)dependence relations. That means that the optimal value for α is in $(p_{max^\circ}, p_{min^1}]$. In this case, p_{max° and p_{min^1} are consecutive values in the

ordered list, i. e. $p_{min^1} = p_{max^0+1}$ and $p_{max^0} = p_{min^1-1}$. Therefore the optimal value of α is usually not greatly above p_{max^0}, nor below p_{min^1}.

For instances that are not separable, p_{min^1} is smaller than p_{max^0} and the optimal interval for alpha, in which the fewest relations are misclassified, lies somewhere between these values.

In general, it holds for the optimal value for alpha that

1. it is larger than p_{min^1-1} and
2. it is smaller or equal to p_{max^0+1}.

Therefore, it is interesting to compare these values over various settings (see the grey lines in Figure 5.12). We refer to the interval $[p_{min^1-1}, p_{max^0+1})$ as the critical range for alpha and denote its size by L_{CRIT}.

▷ $\min_{\text{err}}$, $\min^{\text{f}}_{\text{err}}$, $N_{\bar{\alpha}}$ – Number and fraction of errors, number of minima of the error function In the top panels of Figure 5.12, the number of falsely classified relations is shown for each choice of α. With the relations on the abscissa, sorted by p-value of the conditional independence test, this results in a step function which increases or decreases whenever α is increased to a higher value than that of the next p-value. The minimum number of errors that cannot be avoided when classifying the given set of labelled p-values using a threshold α is denoted by $\min_{\text{err}}$. Dividing this number by the total number of relations, $\binom{n}{2} \cdot \binom{n-2}{k}$ in the k-th round of the PC algorithm, yields $\min^{\text{f}}_{\text{err}}$. We also regard the number $N_{\bar{\alpha}}$ of minima of the error function.

▷ L_{OPT}, $\bar{\alpha}_{\text{limits}}$ – Mean of optimal α limits The interval from which α needs to be chosen in order to obtain the minimum number of errors is denoted by L_{OPT}. However, L_{OPT} might not be continuous, but consist of several continuous subintervals. Moreover, the continuous parts of L_{OPT} might differ significantly in their position, which makes it difficult to compare the magnitude of a good choices of α over various settings. We therefore regard the average of the highest upper bound of an optimal interval for α and the smallest lower bound of a (possibly different) such interval and call it $\bar{\alpha}_{\text{limits}}$. Note that, if these two bounds are not part of the same subinterval, $\bar{\alpha}_{\text{limits}}$ might not be optimal itself (as in the right example in Figure 5.12).

However, as established below, $N_{\bar{\alpha}}$ is generally very small and in practice the optimal intervals lie close together, so $\bar{\alpha}_{\text{limits}}$ is in fact useful as an estimation of a good value for α.

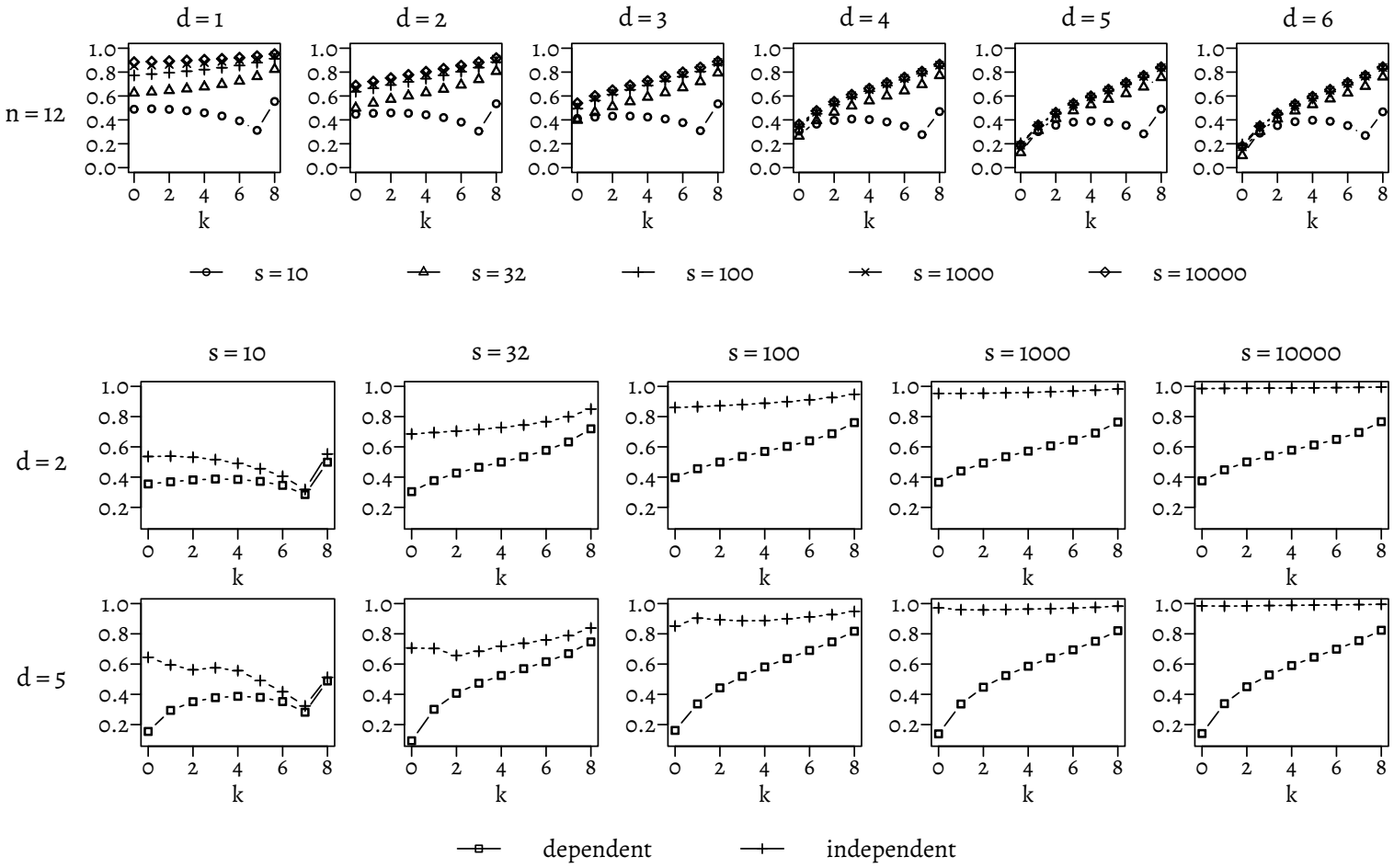


Figure 5.14: Mean of the p-values obtained from the test for partial correlation in different settings, overall (top) and separately (bottom) for true dependences (squares) and independences (plusses).

Results

The most interesting finding of the analysis of the measures defined above is that, in general, it may make sense to regard values of α that are much larger than typical values used in the literature – for example, in [52] the interval $[6.25 \cdot 10^{-4}, 0.04]$ is regarded and in [148] $[5 \cdot 10^{-5}, 0.1]$, Constantinou et al. [54] use $\alpha = 0.01$ and Ramsey, Zhang, and Spirtes [219] regard $\alpha = 0.05$. This is in accordance with the finding of Harris and Drton [122] that $\alpha = 0.5$, the largest value that they considered, yielded the best results.

Different analyses indicate that values of α larger than 0.5 might be adequate in some cases. The problem is that such high values for α can be infeasible in practice due to the "combinatorial explosion" for high values of k in dense graphs.

For the same reason, the analysis presented here was feasible for rather high orders, up to $k = 8$, only for pretty small graphs. Keeping the running time at a few days while at the same time largely avoiding fringe effects when k approaches n, graphs with $n = 12$ nodes are regarded in the following. To investigate the asymptotic behaviour for graphs with more nodes (up to $n = 50$, see Figure A.4 in the appendix), conditional independences were limited to the maximum order $k_{\max} = 2$, which according to our findings in Section 4.3 (see Figure 4.11) is the order for which the mosts tests are performed in the pc algorithm. Every setting was repeated with 10 different graphs, and all the results shown here are averages over these $r = 10$ replications.

Figure 5.14 gives a first impression of how large α might have to be chosen: In the top row of this figure, the mean overall p-values, including both true dependences and true independences, are shown (for n = 12). For reasonable choices of s, these lie above 0.5 most of the time. While at this point, one might hypothesise that this is due to the fact that there are many more true independences with very high p-values than true dependences with low ones, the lower part of the figure shows that this is not the case. Here, one can also see how large (or narrow) the corridor is in which a threshold separating dependences and independences would have to lie on average.

Note also that the obtained p-values converge against 1 with k so that, largely independent of the choice of α, many edges considered in a late round of the pc algorithm are removed, which explains the moderate values for $k_{\max}$ that we encountered for the pc algorithm (see Figure 4.10).

In Figure 5.15, it can be seen that also the values $\bar{\alpha}_{\text{limits}}$ (circles), which can be interpreted as the empirically estimated optimal values for α, are very high, except when s and d are very small. (Furthermore, $\bar{\alpha}_{\text{limits}}$ tends towards the lower of the theoretical bounds p_{max^0+1} and p_{min^1-1} for $d = 2$, but is nearly always indistinguishable from the upped bound when $d = 5$.) The size of the critical range decreases with s and is fairly constant over d and k.

As a background for the interpretation of this figure I studied the other measures that are defined above. The results are summarised here, figures are given in the appendix. Albeit with a fraction of 100% in one case, only for $k = 0$ there are separable instances. However, the region of "overlap" between true dependences and true independences is not too large (in terms of p-values), which can be seen as "almost separability". This overlap is described by the critical range between p_{min^1-1} and p_{max^0+1}, the upper and lower line in the plots in Figure 5.15, and the size of this region is measured by L_{CRIT}. The only exception, where the critical interval is large, is when s is very small (below $s = 100$).

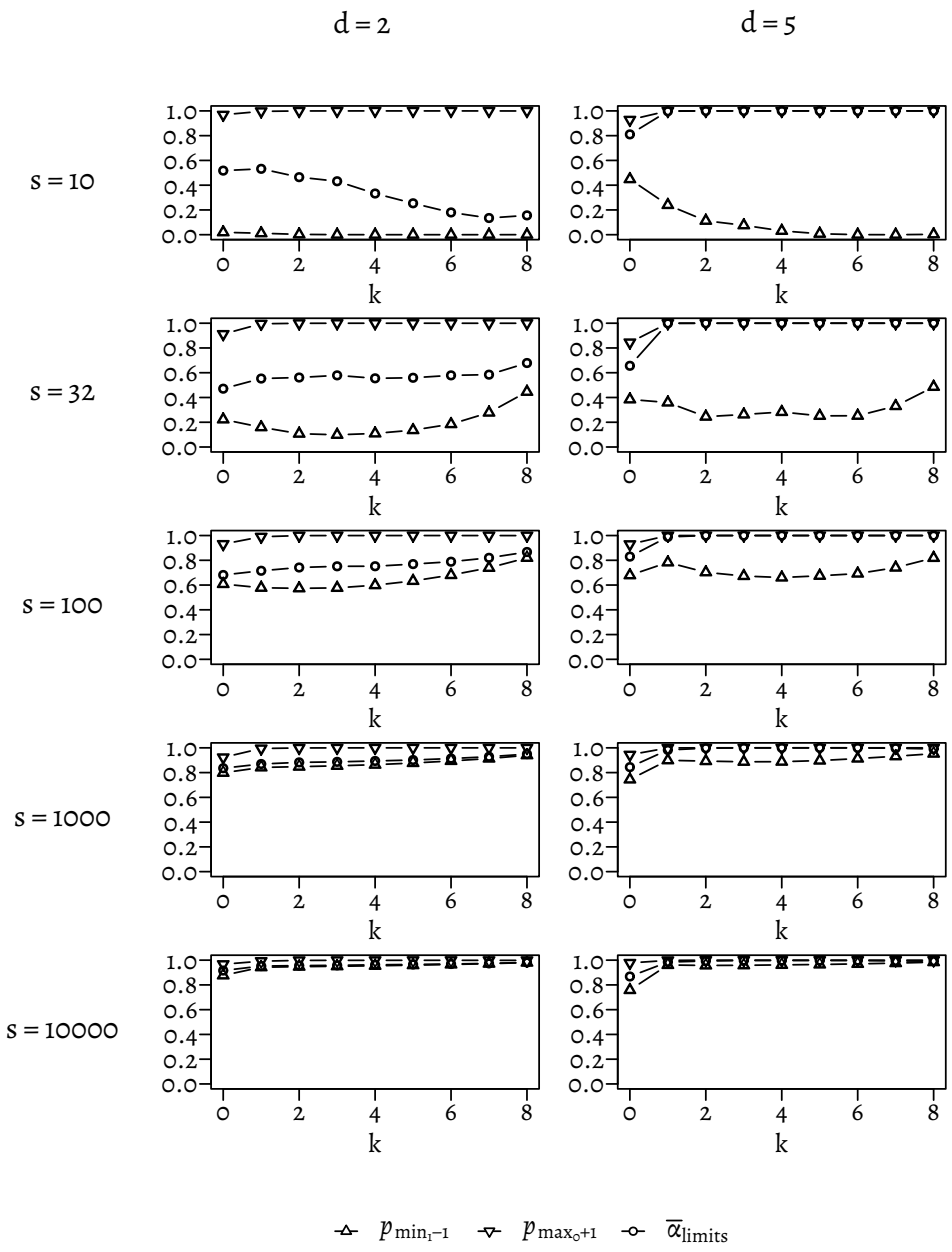


Figure 5.15: Measures that describe the interval in which the optimal value for α lies (see the text) for different sample sizes s (rows) and expected densities d of the true graph (columns). In all panels, the number of nodes is $n = 12$.

If the critical interval is small, the *optimal* interval, from which α needs to be chosen in order to make as few errors as possible, is even smaller. This range is nearly always of a length close to zero and generally becomes smaller for larger values of k. This indicates how difficult it might be in practise to find an optimal value of α. At the same time, it does

not permit any conclusion on how steeply the error function increases with increasing deviation from the optimal value.

It is interesting that the relative minimal number of errors is roughly constant for fixed d, and smaller (about 0.02 for most orders and sample sizes) for denser true graphs ($d = 5$). For sparse true graphs ($d = 2$), both $\min_{\text{err}}$ and $\min^{\text{f}}_{\text{err}}$ can clearly be seen to decrease with increasing sample size.

Determining the number of continuous intervals in which α is optimal can give an indication as to whether it is reasonable to regard $\bar{\alpha}_{\text{limits}}$ as the magnitude of a good choice of α. If there are several optimal intervals, and in particular if there are large non-optimal intervals between them, the mean of the upper bound of the highest and the lower bound of the lowest such interval might be far from optimal itself. Howeverm this is not an important issue here, as indicated by different facts. Firstly, the number $N_{\bar{\alpha}}$ of global minima of the error function is moderate throughout the settings considered for $n = 12$ and also for $n \in \{10, 20, \ldots 50\}$ (not shown; for $d = 2$, $s = 10000$ and $k \leq 3$ the mean number never exceeds two). Secondly, the bounds of the largest and smallest optimal α interval are so similar that they cannot be distinguished in plots like the ones shown here. After all, the total length of the optimal intervals is vanishingly small in most of the settings.

Thirdly, looking at the error function for specific examples shows that when the sample size is large enough, the region in which the minimum lies is demarcated and rather narrow (see Figure 5.16). This figure also explains why the error rate $\min^{\text{f}}_{\text{err}}$ is smaller for denser graphs, although we know from the previous chapter and the results by Kalisch and Bühlmann [148] that the pc algorithm is more accurate on sparse graphs. The reason is that for dense graphs, the number of true independences is very limited, so that the entropy of the decision that has to be made by the test is smaller. Put another way, for dense graphs even a test that always outputs zero would have a lower error rate than for sparser graphs in which the number of true dependences and true independences is more balanced.

The error functions in Figure 5.16 are representative for different replications (with different true graphs), but also for other orders than $k = 7$ (although they naturally vary in the number of data points). When the sample size is decreased, however, the curve for $d = 2$ flattens more and more, until it is nearly constant for $s = 10$ (see Figure A.3). For $d = 5$ on the other hand, the p-values of true independences are gradually "spread out" over the spectrum until for $s = 10$ they cover the hole range. This means that the lower bound of p-values for true independences, denoted by p_{min^1}, gradually decreases. This is reflected in the previous p-value p_{min^1-1}, which constitutes the lower bound of the critical range and is shown in Figure **??** (upwards pointing triangles). One can also see in the lower panels of Figure 5.14 that for $s = 10$ both true dependences and true independences yield p-values with a mean of about 0.5.

Since in practice, the number of variables is often much larger than $n = 12$, I also analysed how the measures develop with n, at least up to $n = 50$. For this, I only considered $d = 2$ and $s = 10000$ for orders up to $k_{\max} = 2$. The results for some interesting measures are given in the appendix.

The minimal error fraction $\min^{\text{f}}_{\text{err}}$ increases only marginally with n, but the size of the critical interval increases gradually and indicating that in a large graph it is more likely that a true independence with a very low p-value or true dependence with a very

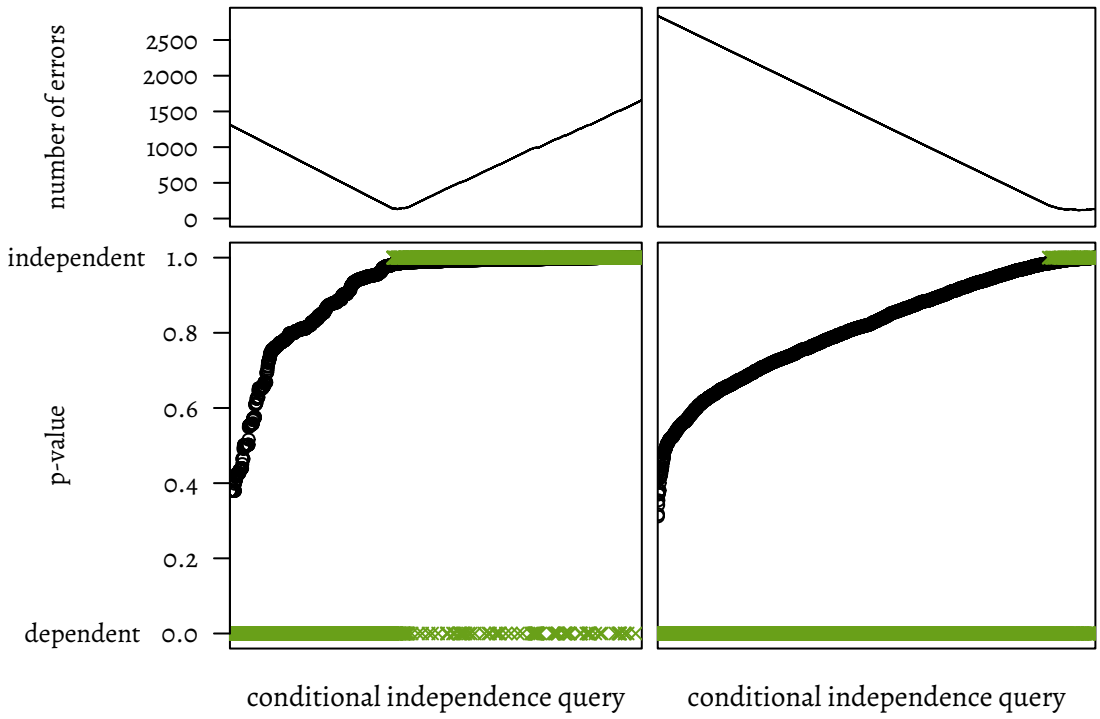


Figure 5.16: Example of p-values obtained from two graphs with $n = 12$ nodes and an expected density of $d = 2$ (left) and $d = 5$ (right), with $s = 10000$ samples and for conditional independence queries of order $k = 7$ (lower panels). Above, the error function is shown, which describes for each conditional independence query the number of errors that would result from the classification of the complete set of p-values if α is chosen from the interval between the p-value of this conditional independence query and the one next in order.

high p-value occurs. At the same time, the size L_{OPT} of the optimal interval decrease so that randomly choosing the optimal α becomes increasingly unlikely. The size of a good choice of α that is described by $\bar{\alpha}_{\text{limits}}$ decreases with n, which is consistent with the model from the previous Section.

Conclusion

In this Section we have seen the difficulties associated with the problem of choosing a good value for α. It turns out that, in particular for small sample sizes it is important to find a suitable value, because the variations in estimation accuracy are particularly large in this case (see Figure 5.10).

Moreover and in particular for small sample sizes, optimal values for α lie much higher that usually assumed. A fist indication is that the analytical derivation of a good choice of α given bei Kalisch and Bühlmann [148] (see Equation (5.1)) yields a value of 0.78 for $s = 32$ if the unknown lower bound of the partial correlations is set to $c_n = 0.1$. As this value probably overestimates the true value in many cases rather than the other way around, the true values might actually be even larger.

The analyses in the Section 5.3.3 showed that the values of α that are optimal for the *individual* conditional independence tests are very high, usually larger than 0.8. However, in this context, it is important to take into account the algorithmic structure of the pc algorithm. In particular, since an edge is tested multiple times with different separating sets, a type I error, that is erroneously keeping an edge, might not be problematic because repetitions of the tests for other separating sets can correct the error. This fact indicates that it might indeed be good to choose rather high values of α, in order to more strictly bound the probability of the type II error instead. Nevertheless, this will most probably lead to a prohibitively large the running time for medium sized or large numbers of nodes. On top of that, a contrary effect suggests favouring α, which might lead to erroneous edge deletions. Consider the right example ($s = 100$) in Figure 5.12. Three out of four of the largest p-values of true dependences correspond to a dependence $X \not\perp\!\!\!\perp Y$, for which there exists a single node Z such that $X \perp\!\!\!\perp Y$. In other words, the edges should not be removed in this round, but in the next. In this sense, removing the edge $X - Y$ is an error locally, but not globally.

The model derived in Section 5.3.2 optimises the structural Hamming distance (SHD) to the true CPDAG over a wide range of settings. The result is that $\hat{\alpha}$ should decrease with the sample size and the number of variables, but increase with the degree of the true graph. In practical applications the latter must be estimated from the context and still leaves some room for the choice of α. The evaluation presented in Figure 5.11 shows that that the values of $\hat{\alpha}$ obtained by the model do not only lie close to the one that optimises the SHD, but also to the optimal values according to the F1 measure, which combines the true positive and true detection rate of the edges (see Equation (3.6))

5.4
Turning the Output of the pc Algorithm into an Extendible Graph

While a CPDAG represents a class of DAGs which all may have produced data with the dependences and independences that are described by the CPDAG, non-extendable graphs or graphs containing conflict edges do not represent any such DAG that might have generated the data.

However, as we have seen in the previous section, it is not immediately clear what kind of graph we would like to find, with respect to the "approximation" computed by the pc algorithm.

We have seen that many obstacles prevent the pc algorithm from actually outputting a CPDAG in the sample version. Let us therefore relax the objective and try to learn an *extendible graph* instead, which also yields the extension itself, a DAG. (As noted before, this DAG could then be turned into its CPDAG if necessary.)

Two main problems have to be solved to achieve this goal. Firstly, we have to modify unextendable graphs so they become extendable. Secondly, we must define how conflict edges are treated in this procedure.

A conflict edge $X \leftrightsquigarrow Y$ in the pc-estimated GPDG expresses three facts about the true DAG $\mathcal{G}^*$ that have been established in the course of the pc algorithm:

1. X and Y are adjacent in $\mathcal{G}^*$ *and*
2. the edge between X and Y has to be directed as $X \to Y$ in $\mathcal{G}^*$ *and*
3. the edge between X and Y has to be directed as $X \leftarrow Y$ in $\mathcal{G}^*$.

Since the latter two statements contradict each other, for each conflict edge, we must accept at least one error and discard one of the above assumptions.

For some GPDG $\mathcal{G}$, let us consider the set $\mathcal{K}(\mathcal{G})$ of all PDGs that can be obtained from $\mathcal{G}$ by omitting one of the above assumptions for each conflict edge in $\mathcal{G}$, that is, each conflict edge is removed, or directed in one of the two possible ways. Since all combinations of directions and removal for all of the κ conflict edges are considered, $\mathcal{K}(\mathcal{G})$ has the cardinality 3^κ.

Since the skeleton is often more reliable than the edge directions, it also makes sense to regard the case where discarding assumption 1 is not allowed, that is conflict edges might only be directed in one or the other direction. The corresponding set of conflict graphs is of size 2^κ for κ conflict edges and denoted by $\overline{\mathcal{K}}(\mathcal{G})$, where the overline connotes the edge that is kept in the skeleton.

Let the set $\mathcal{F}(\mathcal{G})$ of *feasible conflict graphs* be the subset of extendable graphs in $\mathcal{K}(\mathcal{G})$ with respect to a graph $\mathcal{G}$:

$$\mathcal{F}(\mathcal{G}) = \{\mathcal{G}' \in \mathcal{K}(\mathcal{G}) \mid \mathcal{G}' \text{ is extendable}\}$$

and let $\mathcal{H}_h(\mathcal{G})$ be the set of GPDGs that can be obtained from $\mathcal{G}$ by removing h edges. Let $\mathcal{F}_h(\mathcal{G})$ be the set of extendable graphs among the conflict graphs for a GPDG in $\mathcal{H}_h(\mathcal{G})$, that is

$$\mathcal{F}_h(\mathcal{G}) = \bigcup_{\mathcal{H} \in \mathcal{H}_h(\mathcal{G})} \mathcal{F}(\mathcal{H}).$$

To avoid a potentially exponential growth (in h) of the size of $\mathcal{H}_h(\mathcal{G})$ and $\mathcal{F}_h(\mathcal{G})$, we only consider the removal of the "weakest" h edges according to some metric.

This *weakness metric* could measure the strength of the direct effect expressed by the respective edge, with the rationale that edges that represent marginal effects are less meaningful than other edges. Thus weakness would be measured inverse to edge weights.

Another statistically motivated measure is the maximal p-value that was determined for this edge using any separating set that was regarded during the computation of the skeleton. In the implementation of the PC algorithm in the R package `pcalg`, this value is returned for all edges in the slot `p.max`. Similarly, we denote it by $p_{\max}$ here; it is not to be confused with p_{max°, the maximal p-value of a true dependence from the supervised setting regarded in the previous section.

For each edge in the output graph, the value $p_{\max}$ will be below the level of significance that was used in the tests because otherwise the edge would have been removed. Thus, $p_{\max}$ measures how close the edge was to being considered false during the computation, with higher values of $p_{\max}$ indicating a higher uncertainty about the existence of the edge. In this sense, a high value of $p_{\max}$ for some edge thus indicates that it is a weak edge. To summarise, we consider this specific problem:

Problem 1 (*Best-Feasible-Graph(G,D,S)*).
Given a general partially directed graph $\mathcal{G}$, a data set D and a scoring function S, find a value h so that $\mathcal{F}_h(\mathcal{G})$ is non-empty, while for all $h' < h$, the set $\mathcal{F}_{h'}(\mathcal{G})$ is empty. Output a graph $\mathcal{G}_{\text{opt}} \in \mathcal{F}_h(\mathcal{G})$.

Such a graph always exists, since for example when all undirected edges and an edge from every directed cycle are removed, the resulting graph is always a DAG (possibly with several connected components). When deletion of conflict edges is not considered, this graph can be found naïvely by Algorithm 2. This algorithm is based on computing and evaluating the sets $\mathcal{H}_h(\mathcal{G})$ that can be obtained from $\mathcal{G}$ by removing h edges (according to the weakness measure).

Data: a GPDG $\mathcal{G}$
Result: a DAG from $\mathcal{F}_h(\mathcal{G})$, for the smallest possible value of h
1 $\mathcal{H}_0(\mathcal{G}) = \{\mathcal{G}\}$
2 **for** $h \in \{1, 2, \dots\}$ **do**
3 **for** *each graph* $\mathcal{G}'$ *in* $\mathcal{H}_{h-1}(\mathcal{G})$ **do**
4 **for** *each graph* $\mathcal{G}''$ *in* $\overline{\mathcal{K}}(\mathcal{G}')$ **do**
5 **if** $\mathcal{G}''$ *is extendible* **then**
6 **return** *a consistent extension of* $\mathcal{G}''$
7 **end**
8 **end**
9 **end**
10 $\mathcal{H}_h(\mathcal{G}) = \emptyset$
11 **for** *each graph* $\mathcal{G}'$ *in* $\mathcal{H}_{h-1}(\mathcal{G})$ **do**
12 remove the weakest edge from $\mathcal{G}'$ and add the resulting graph to $\mathcal{H}_h(\mathcal{G})$
13 **end**
14 **end**

Algorithm 2: Computing the "best feasible graph" (see Definition 1).

Note that when the weakest edge is always unique, the sets $\mathcal{H}_h(\mathcal{G})$ and thus also the sets $\mathcal{F}_{h-1}(\mathcal{G})$ are always singletons, that is they contain only one element. Nevertheless, it is useful to regard sets of graphs rather than single graphs, not only to provide for the case that there are several weakest edges. Also in the exhaustive case, when not only the weakest edge is removed, but all edge removals are considered, $\mathcal{H}_h(\mathcal{G})$ is non-singleton. Although the size of the sets increases so fast with h that it is infeasible to consider all graphs in $\mathcal{H}_h(\mathcal{G})$ even for $h = 2$ when the G has, for example, 100 edges, it might be possible for small or very sparse graphs or under certain constraints. For example, one might consider the removal of all edges with a certain minimal weakness.

Since for a graph $\mathcal{G}'$ with k conflict edges there are 2^k graphs in $\overline{\mathcal{K}}(\mathcal{G}')$ (and 3^k in $\mathcal{K}(\mathcal{G}')$), the for-loop in lines 4 to 8 has exponential complexity in the number of conflict edges in $\mathcal{G}'$. This can easily become very costly even though the number of conflict edges usually is significantly lower than the total number of edges (see, for example, Figure 4.17). In the next section, a new polynomial method is proposed to determine whether a GPDG, that possibly contains conflict edge, is extendible. This algorithm can replace the for-loop and determine in polynomial time whether any graph in $\overline{\mathcal{K}}(\mathcal{G}')$ is extendible.

5.4.1 Extension of GPDGs

We have defined above the set $\mathcal{F}(\mathcal{G})$ of feasible conflict graphs as a subset of the set of conflict graphs $\mathcal{K}(\mathcal{G})$ for a graph $\mathcal{G}$, demanding that on top of being an element in $\mathcal{K}(\mathcal{G})$, a graph in $\mathcal{F}(\mathcal{G})$ needs to be extendible. Based on this, will now define the notion of extendability for GPDGs as follows, so that the set $\mathcal{F}(\mathcal{G})$ of feasible conflict graphs is equal to the set of consistent extensions of a GPDG $\mathcal{G}$.

Definition 15 *(v-structures in a GPDG).*
A GPDG $\mathcal{G}$ contains as v-structures all its induced subgraphs of the nodes X, Y and Z that consist of

1. two directed edges: $X \to Y \leftarrow Z$ or
2. a directed edge and a conflict edge: $X \to Y \leftrightsquigarrow Z$ or
3. two conflict edges: $X \leftrightsquigarrow Y \leftrightsquigarrow Z$.

Definition 16 *(Consistent extension of a GPDG).*
The consistent extension of a GPDG $\mathcal{G}$ is a DAG $\mathcal{G}'$, so that all of the following hold:

1. Every directed edge that exists in $\mathcal{G}$ also exists in $\mathcal{G}'$.
2. For every edge $X - Y$ that exists in $\mathcal{G}$, there exists either $X \to Y$ or $X \leftarrow Y$ in $\mathcal{G}'$.
3. For every edge $X \leftrightsquigarrow Y$ that exists in G, there exists either $X \to Y$ or $X \leftarrow Y$ in $\mathcal{G}'$.
4. The set of v-structures contained in $\mathcal{G}'$ is a subset of the v-structures contained in $\mathcal{G}$.

Note that any v-structure in $\mathcal{G}$ that consists of two directed edges must also exist in $\mathcal{G}'$.

In particular, in the extension of a GPDG no v-structures may form by directing an undirected edge and a conflict edge towards their common node. It is, however, admissible to direct a conflict edge towards a directed edge that already existed in $\mathcal{G}$ (even if the triple is not shielded).

An example for the extension of a GPDG is given in Figure 5.17. Consider the GPDG on the left. To extend it, the edge between W and Z must be directed as $W - Z$ because otherwise, a new v-structure $V \to W \leftarrow Z$ would form. Then, to avoid a v-structure $W \to Z \leftarrow Y$, we must direct the edge between Z and Y towards Y. Now, the only way to prevent a directed cycle is to direct the conflict edge towards W, obtaining the DAG shown on the right. Note that in the last step a v-structure is formed, but since a conflict edge should form v-structures with all incident directed edges, this is legal.

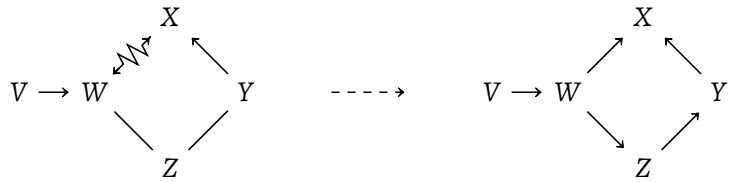


Figure 5.17: Example for the extension of a graph with conflict edges.

In the following, an algorithm is presented to compute such a consistent extension if it exists. It is based on an algorithm by Dor and Tarsi [79], which computes the consistent extension of a PDAG. The algorithm successively chooses a node V without (directed) outgoing edges whose undirected neighbourhood is a clique and forms a complete bipartite

graph with its directed neighbours. If such a node is found, all the edges incident to it are directed towards V. Then, for the rest of the computation, V is removed.

The node V that is repeatedly searched for in Dor and Tarsi's algorithm is referred to as a sink node and has to fulfil two conditions.

Definition 17 *(Valid sink node in Dor and Tarsi's algorithm).*
A node V is valid as to be chosen as the sink node in Dor and Tarsi's algorithm if

1. no directed edges originate from V *and*
2. every node that is connected to V by an *undirected edge* is connected to *all* other neighbours of V.

Intuitively, the two rules guarantee the following: Due to rule 18, the graph is recursively assembled from the sink. Rule 18 assures that the directing of edges that is done when forming a sink does not lead to new v-structures because all the colliders that form are shielded.

When we want to generalise the algorithm to graphs that contain conflict edges, we need to adapt the above criteria accordingly. In the adaptation of the algorithm, that we will refer to as the CECE algorithm (for consistent extension with conflict edges), we are going to direct not only the undirected, but also the conflict edges towards the chosen sink node V. The setting is depicted in Figure 5.18.

Therefore, conflict edges are – like undirected edges – not regarded in rule 18: Only *directed* edges originating from V must be forbidden because these cannot be redirected towards V.

Rule 18 prevents that new v-structures form when edges are directed towards the chosen sink node V. Since v-structures are allowed to form from conflict edges (actually, they should and will), we do not need to be careful about them here as we need to be about the undirected edges. However, since conflict edges are never deleted, only directed, they are valid as shields that prevent the formation of v-structures when undirected edges are directed. Therefore, also conflict edges are relevant when checking that the endpoints U of undirected edges that originate from the sink node V are connected to its other neighbours. (That is, the edges g and h in Figure 5.18 may be conflict edges.)

In addition, these nodes U need to be connected also to nodes that are connected to V by conflict edges (edge h in the figure). Namely, if such a conflict edge f is directed towards V, it may itself form v-structures with other directed or conflict edges, but not with edges that were undirected in the original graph (like e). Such colliders must therefore also be shielded by an edge h.
All in all, rule 18 must be adapted to require that the endpoints U of all undirected edges that originate from V must be connected to the other neighbours of V, be they connected by directed, conflict or other undirected edges. We thus maintain the above algorithm, or expressed more explicitly:

Definition 18 *(Valid sink node in the CECE algorithm:).*
A node V is valid as to be chosen as the sink node in the CECE algorithm if
No directed edges originate from V *and*
Every node that is connected to V by an *undirected edge* is connected to *all* other neighbours of V (adjacent to V by directed, undirected or conflict edges).

This yields Algorithm 3.

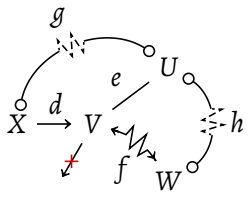


Figure 5.18: Illustration of the CECE algorithm. For the choice of V as the "sink" node it is required that there are no outgoing directed edges (rule 18). Furthermore any node U that is connected to V by an undirected edge e needs to be connected to each other neighbour of V, here X and U. The edges g and h that establish this connection can be directed, undirected or conflict edges.

Data: a GPDG $\mathcal{G}$
Result: a DAG from $\mathcal{F}(\mathcal{G})$, if $\mathcal{F}(\mathcal{G})$ is nonempty

```
1  G' = G
2  while there are nodes left in G' do
3  |  if there is a valid sink node v in G' then
4  |  |  direct all edges that are incident to v towards v in G remove v along with
   |  |    all its incident edges from G'.
5  |  end
6  |  else
7  |  |  return ⊥                          ▷ the input graph was not extendible
8  |  end
9  end
10 return G'
```

Algorithm 3: Main structure of Dor and Tarsi's algorithm [79], which with the definition of a sink node given in Definition 18 yields the CECE algorithm for the extension of a GPDG.

THEOREM 5.1.
The CECE algorithm establishes a consistent extension of a conflict graph if and only if it exists.

Proof. First, we will prove that the algorithm outputs a consistent extension. The correctness of the general procedure was proven in [79]. Only the impact of conflict edges needs to be investigated. It is obvious that all the conflict edges are directed if the algorithm terminates, so condition 3 of Definition 16 is fulfilled. Furthermore, the conflict edges do not restrict the choice of sink nodes since they are not regarded for this. This is also in accordance with the fact that conflict edges may be directed in either way. When evaluating rule 18, it is correct to treat conflict edges like directed edges: It is not necessary to avoid the forming of v-structures from conflict edges (if no undirected edges are involved). However, undirected edges should not form v-structures with conflict edges, just as they should not with directed edges. The DAG that is produced by the CECE algorithm thus fulfils the requirements given in Definition 16.

It remains to show that if a consistent extension exists, it will be found by the algorithm. This can be done by the same argument as in [79]: Assume that there was a consistent extension with an edge that is directed away from a node V that we chose as a

sink. These edges must have been undirected or conflict edges in the input graph since directed edges can never be flipped in a consistent extension.

Then, we could just redirect this edge towards V since no directed cycles can form by directing all incident edges towards V. (On top of that, this is ensured by our choice of the node V according to the above rules that no v-structures form.)

That is, if there is a consistent extension, the algorithm does not necessarily produce it (which is only natural since there can be more than one consistent extension for a graph, and the algorithm only outputs one), but it is not possible that the algorithm makes a decision that prevents it from finding a consistent extension.

This completes the proof. □

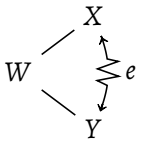


Figure 5.19: Illustration of why the "greedy" strategy of the CECE algorithm does not work when conflict edges can be removed.

Note that the proof does not cover the case where conflict edges are also considered to be removed, instead of directed in either way. (This corresponds to regarding $\mathcal{K}(\mathcal{G})$ instead of $\overline{\mathcal{K}}(\mathcal{G})$.) If conflict edges might be removed, it would not be possible to decide how to treat conflict edges in rule 18: Assume there is a node W that has two incident undirected edges whose endpoints are connected by a conflict edge e (see Figure 5.19). If e is kept, then W would be eligible as node V in the algorithm (assuming that it has no other incident nodes). But considering the possibility that e might be removed in the future, it would become ambiguous whether W may be chosen as a sink.

A heuristic approach for this case would be to mark such edges e as conflict edges that are not allowed to be removed anymore. However, a (in general exponential) backtracking procedure would be necessary for an exact algorithm if there is more than one possibility to chose the sink node V. It is an open question whether also the problem to determine for a graph $\mathcal{G}$ an extension in $\mathcal{K}(\mathcal{G})$ is solvable in polynomial time.

6

Causality in Proteins

In this chapter the PC algorithm is applied to learn networks of energetic paths in proteins with the goal of understanding allosteric communication. As proteins play an important role in a wide range of biological processes as enzymes, structural or signalling proteins, this problem is of high relevance in the life sciences. It is thus an important open problem in biology to understand the function of proteins.

Protein-protein interaction is omnipresent and often, proteins have multiple separate active sites or domains with different functions. One can think of the interaction of two proteins as an energetic perturbation that originates from the binding surface and disseminates through the molecules [169]. Such a perturbation can have long range, *allosteric* effects to different domains of a protein, that trigger a response there.

Signalling of hormones is commonly mediated by allosteric effects, as binding to the specific receptor can affect its properties at another site. Impairment of these signalling pathways can lead to severe diseases as seen, for instance, in type 2 diabetes. Allosteric effects also enable ion channels to transport ions into and out of cells, as well as viruses to enter host cells. Understanding communication in proteins therefore has various applications in drug design.

Questions that one might be able to answer with the help of the methods described in this thesis include the following:

- Why does a receptor fail to transfer a signal?
- How is a virus internalised by a host cell upon binding?
- Which mutations can cause an ion channel to loose its function, for example in cystic fibrosis patients?

A new approach to the study of allosteric regulation in proteins is proposed here, for which the protein is modelled as a causal network of its amino acids. This network conceptually does *not* correspond to the chemical bonds in the protein but rather represents functional dependences. These can be measured by different types of data, for example in terms of energies or evolutionary conservation, as described below.

Whatever the kind of data, modelling the amino acids as variables we expect the data in form of a set of observations for each of the n positions.

In principle, the variables of the positions can be categorical or ordinal, so for example, a multiple sequences alignment (MSA) could be used. In this case, a suitable conditional independence test would have to be chosen. However, also for evolutionary data, we only consider numerical variables here. Furthermore, we assume Gaussianity and

use Fisher's Z test (see Equation (2.6)) for conditional independence testing. For this and other assumptions made by the PC algorithm see Section 3.2.

In this chapter, two case studies are presented, each with a different type of numerical data: In Section 6.1, nuclear magnetic resonance (NMR) data are used to study the mitogen activated protein kinase p38γ. In Section 6.2 we consider the PDZ domain family on the basis of evolutionary data.

6.1
NMR Data

Aoto, Martin, and Wright [12] study the allosteric communication between the regulatory sites in the mitogen activated protein kinase p38γ. For this, they apply second order Markov model analysis to learn an undirected network between the residues. They then obtain clusters within the protein using a method called link communities [3], which determines a clustering of the edges in a graph.

The supplemental material of [12] contains the NMR data on which the paper is based, ^{1}H-^{13}C methyl chemical shifts for 78 variables. These chemical shifts "report on the time-averaged local chemical environment and therefore reflect conformation and dynamic processes on a broad timescale (ps-ms)" [12]. The observations in these data sets are experimental, but we do not use interventional information here. They consist of so-called soft mutations, that is, amino acids are mutated into chemically similar ones by methyl mutagenesis, e.g. from leucine to valine. Although perturbing the molecular dynamics only mildly, these mutations nevertheless produce long-range effects at distances up to more than 40 Å.

Although working with an undirected graph, Aoto, Martin, and Wright [12] are eventually interested in directed effects in the graph. It thus suggests itself to learn a directed causal model, for example using the PC algorithm. The results of this approach are presented here.

6.1.1
Methodology

As we have seen in the previous chapters, the PC algorithm is very unreliable for small sample sizes. I thus combined all the data from [12], which are for four different states of the protein (active, inactive, with the energy providing molecule ATP, and the inhibiting ligand BIRB796), to obtain a data set with $s = 58$ samples (for the $n = 78$ variables). Then, all variables with a variance lower that 1% of the maximal variance over all variables were removed, so that $n = 49$ variables remained. After that, the data were standardised to unit variance and zero mean.

With these data, I ran the PC algorithm with a value of $\alpha = 0.09$ determined for $d = 5$ with the model from Section 5.3.2. I used the stable version of the PC algorithm with v-structure identification by majority rule. This yields an extendible graph with 25 directed edges, 20 undirected edges and no conflict edges. (For comparison: Using the conservative version of the v-structure identification, only 10 edges are directed, the other ones

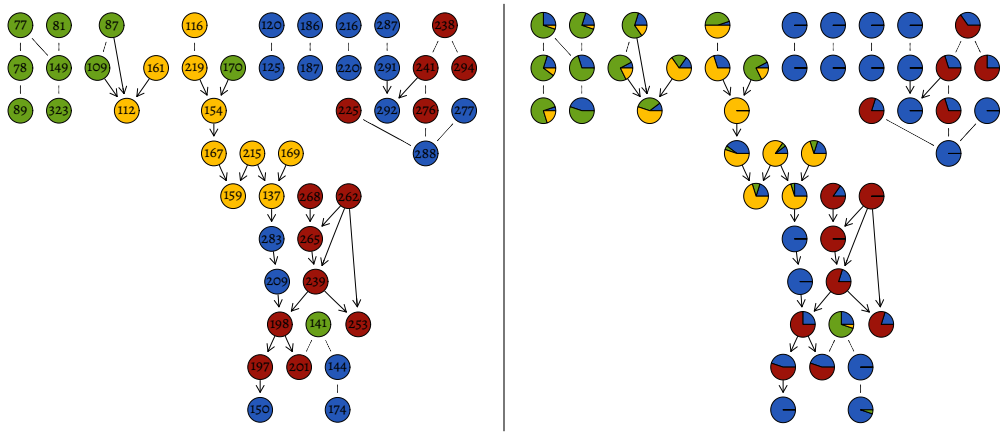


Figure 6.1: The PC-estimated graph, node colours of the right graph are as specified in [12] The node colours in the left graph correspond to dominant color of the respective node in the right graph. Bidirected edges represent undirected edges.

remain undirected. With the original, unstable version of the PC algorithm, 11 conflict edges form, but there are also 21 directed edges.)

It is notable that the graph does not change when the maximum order of independence tests is restricted to a value $k_{max} \geq 2$, due to the sparsity of the resulting graph. The computation only takes a few seconds.

6.1.2 Results

The PC algorithm was run with Meek's rules and detection of conflict edges (`solve.confl = TRUE`). The resulting graph is shown in Figure 6.1. The colouring of the nodes is based on the results by Aoto, Martin, and Wright [12] as follows: In Figure S3 (supplementary information) they show a clustering of the edges of their graph. This entails a colour distribution for each node which corresponds to the colour distribution of the edges that are incident to this node. These node colours are reproduced in the right graph in Figure 6.1 (the colour distributions were read off by eye; blue corresponds to purple). The node colours of the left graph correspond to the *most frequent* cluster that edges incident to the respective node have in Figure S3 .

Aoto, Martin, and Wright also show the location of the clusters that they identify in the 3D structure of the protein (top right panel of Figure S3). One can see there that the purple cluster is distributed over the whole protein while the other clusters are restricted to smaller subdomains. It might thus be reasonable that this cluster is subdivided over several connected components in the PC-estimated graph. Otherwise, the edges in this graph correspond well with the clusters identified in Figure S3 of [12].
A new clustering can be obtained from the PC-estimated graph in terms of its connected components. The resulting clusters are shown in a 3D model in Figure 6.2 (left). It is also

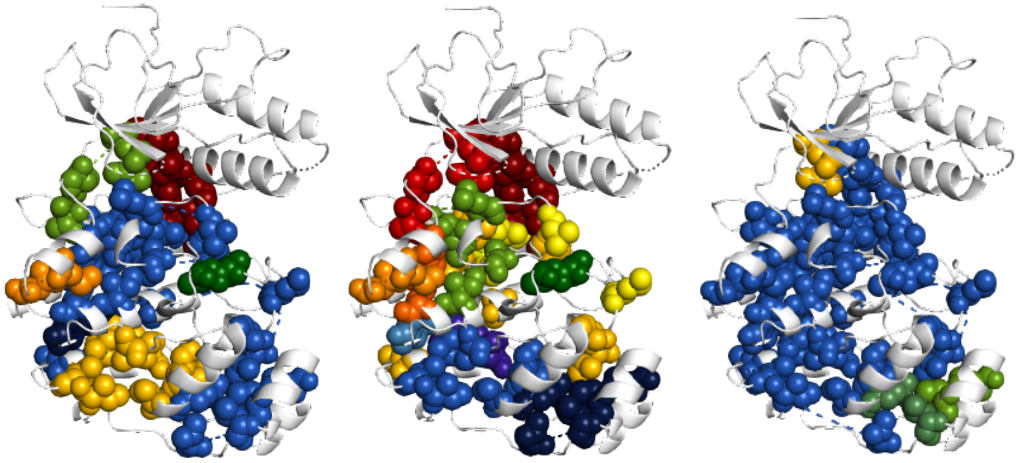


Figure 6.2: The node clusters obtained from the graph shown in 6.1; on the left, clusters correspond to the connected components of the graph, in the middle to the clusters identified by infomap clustering of the graph, on the right by a link communities analysis.

possible to use graph clustering methods to further subdivide clusters corresponding to the connected components. Rosvall and Bergstrom [228] propose a method for this which is based on maps of random walks on the graph structure, and known as *infomap*. This method yields 13 clusters (as opposed to 7 connected components), which are shown in the middle of Figure 6.2. Also a clustering with the link communities method that is used by Aoto, Martin, and Wright [12] is given in the right part of figure, although the method is better suited for denser graphs and thus yields very few clusters of uneven size.

All these clusterings in general yield smaller clusters that the ones presented in [12], but it can be seen that positions in a common cluster that are not contiguous in the primary structure (given by the indices), often are proximal in the tertiary structure. This indicates that meaningful clusters are identified. Moreover, the large clusters that are identified as a connected component and by the link communities method, respectively (both shown in sky blue), which span large parts of the protein, are analogous to the purple cluster identified in Figure S3 of [12].

However, also sporadic larger "gaps" in a cluster are plausible, since energy can be transferred in a "seemingly arbitrary manner" [169] and the resulting longe range effects are detectable by the kind of NMR data used here [177].

Apart from clustering methods, there are various other possibilities for further analysis based on the graph obtained by the PC algorithm. Depending on the protein under consideration, particular domains or residues might be of interest. Moreover, one could consider particular paths that are robustly found in the graph for different sizes of α and analyse their course in the 3D-structure. It is also possible to estimate causal effects based on the PC-estimated graph, as will be further discussed in the next Section.

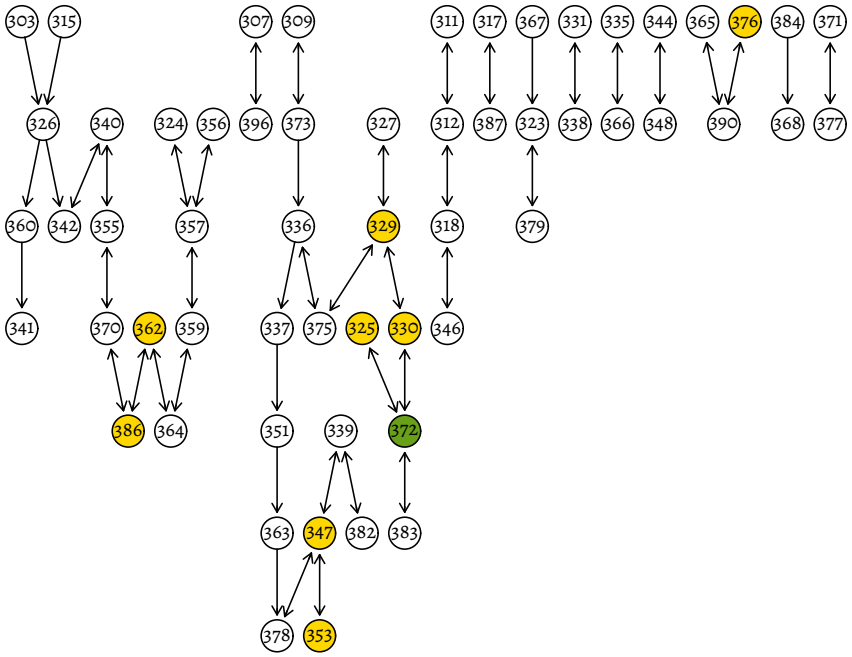


Figure 6.3: The resulting graph after modification by Algorithm 2 when the conservative v-structure identification is used. Note that the variables are identified by their index, which according to the underlying sequence data is numbered between 303 and 396. Position 372 (green) correspond to position 76 studied in [169] and found to be coupled with the positions that are coloured yellow here. Bidirected edges represent undirected edges.

6.2 Evolutionary Data

Lockless and Ranganathan [169] consider measures to quantify the evolutionary conservation of a residue as well as the statistical coupling of two residues. For this, they use evolutionary data given in multiple sequence alignment (MSA) that contains amino acid sequences of the members of a protein family. Since the function of such proteins has been conserved through evolution, the MSA yields interesting information about the function of the protein. In particular, it is possible to derive which residues are required for the protein to work, but also about which positions in the protein are likely to "collaborate": If two positions co-evolve, this indicates that the protein is dysfunctional or unstable and thus abolished by selection when only one of the positions is mutated. If however, the other position also mutates, the communication between the positions can be retained and the protein maintains its functionality. In this sense, evolutionary coupling can permit conclusions about energetic coupling as measured by NMR in the previous section.

To assess evolutionary coupling, Lockless and Ranganathan [169] consider the following quantities based on an MSA with n positions and m sequences and the overall base frequencies of the different amino acids (over different proteins and species). The latter

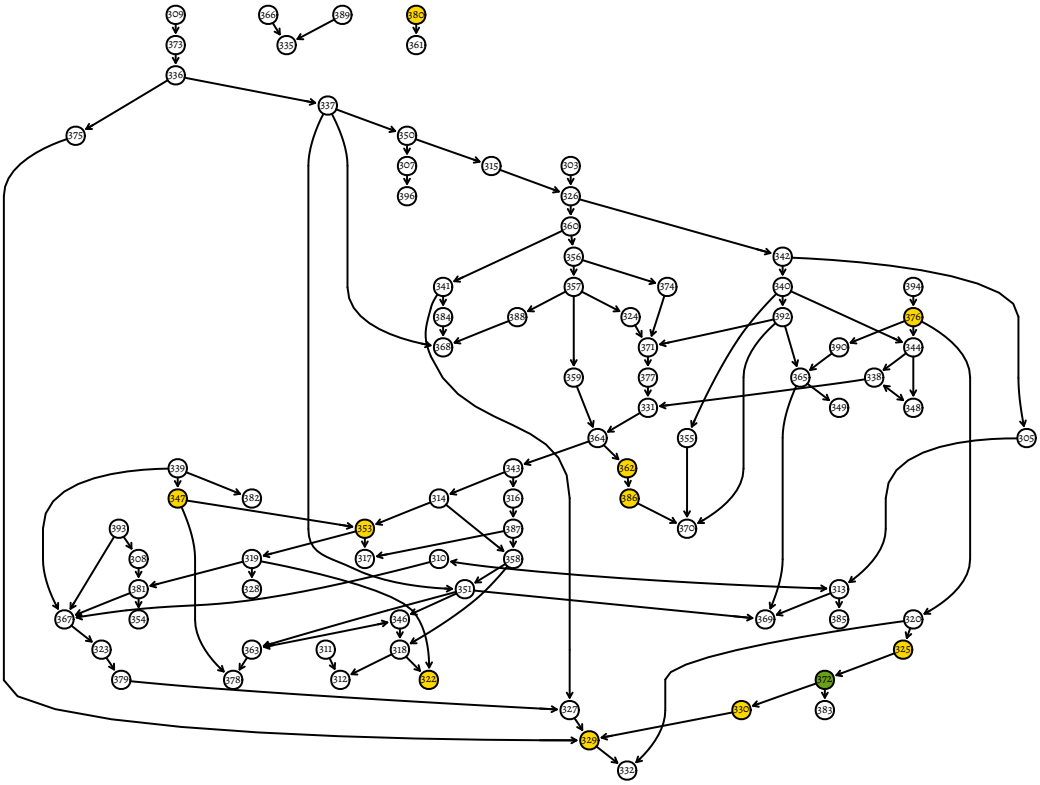


Figure 6.4: The resulting graph after modification by Algorithm 2 when the majority rule is used for v-structure identification. Note that the variables are identified by their index, which according to the underlying sequence data is numbered between 303 and 396. Position 372 (green) correspond to position 76 studied in [169] and found to be coupled with the positions that are coloured yellow here. Bidirected edges represent undirected edges.

are considered as estimates for the probability of each amino acid to randomly occur in a sequence. Let these probabilities be denoted as p_x for each amino acid x. Furthermore, f_i^x is the number of occurrences of amino acid x at position i in the given MSA. Then the probability for the number of occurrences f_i^x of an amino acid x at position i can be computed (in retrospective) according to a binomial distribution:

$$P_i^x = (p_x)^{f_i^x} \cdot (1 - p_x)^{m' - f_i^x},$$

where m' is the total number of other amino acids at position i (which differs from the number of observations by the number of gaps). Moreover, the quantity P_{MSA}^x is defined by considering instead of position i a hypothetical position in which all amino acids occur in the mean frequency that they have in the complete MSA.

Now, the conservation of a position i in the MSA is measured by the evolutionary conservation parameter ΔG_i^{stat},

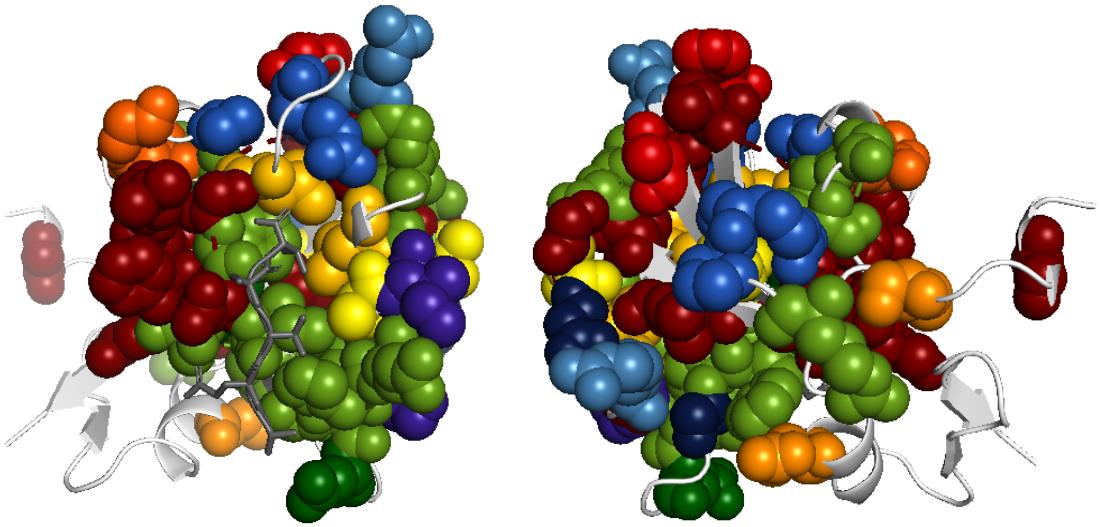


Figure 6.5: The node clusters obtained from the connected components of $\mathcal{G}'_{\text{cons}}$ shown in Figure 6.3. The two images show opposite sides of the protein (turned by 180° around the vertical axis).

$$\Delta G_i^{\text{stat}} = kT^* \sqrt{\sum_x \left(\ln \frac{P_i^x}{P_{\text{MSA}}^x} \right)^2},$$

in terms of an arbitrary energy unit kT^*.

Furthermore, perturbations δ_j^y of a position j with an amino acid y are defined as selecting the subset of rows of the MSA, in which the amino acid y occurs at position j (see also Figure 1 in [274]). P_i^x and P_{MSA}^x computed in this sub-alignment are referred to as $P_{i|\delta_j^y}^x$ and $P_{\text{MSA}|\delta_j^y}^x$, respectively, and referred to as probabilities *under the perturbation* δ_j^y.

Now, the correlation between two sites i and j can be described in terms of the change in conservation of site i when site j is perturbed. This leads to the statistical coupling energy $\Delta\Delta G_{i,j}^{\text{stat}}$, which we regard here dependent on an amino acid y for the perturbation

$$\Delta\Delta G_{i,j,y}^{\text{stat}} = kT^* \sqrt{\sum_x \left(\ln \frac{P_{i|\delta_j^y}^x}{P_{\text{MSA}|\delta_j^y}^x} - \ln \frac{P_i^x}{P_{\text{MSA}}^x} \right)^2}.$$

This measure (for a fixed y) is the core of the analysis in [169] as well as in [274], which is based upon the former. To obtain input data for the PC algorithm, I computed this measure for each position, considering each other position j and each amino acid y that has a relative frequency between τ_f and $1 - \tau_f$ at position j. In the resulting data set, the variables correspond to the positions i.

For PDZ, $\tau_f = 5\%$ on average yields 5 perturbations per site. To be precise, the resulting data set consists of $n = 92$ variables corresponding to the positions of the protein, that are labelled with indices between 303 and 396, and 460 observations corresponding

to perturbations. Since the data set is larger and has smaller variations than the one considered in the previous section, a variance threshold is not necessary here, but standardisation was performed anyway. The resulting data generally lead to denser graphs so I set the target density for the determination of α according to the model from Section 5.3.2 to $d = 2$ here, which yields $\alpha = 0.0003$.

6.2.1 Results

Like in the previous section, the PC algorithm was run with Meek's rules and conflict detection (`solve.confl = TRUE`). Also here, the original identification method for v-structures leads to a large amount of conflict edges (57 out of 120 edges). While the majority rule is able reduce this number to 8 conflict edges (and increase the number of directed edges from 53 to 108), the conservative version of the PC algorithm yields no conflict edges at all. Nevertheless, it also reduces the number of identified directed edges from 53 to only 22. We will analyse both results in the following, referring to the graph estimated with the conservative version as $\mathcal{G}_{\text{cons}}$and the one for which the majority rule was used as $\mathcal{G}_{\text{maj}}$.

Both graphs are not extendible. Both contain a (partially directed) cycle, $\mathcal{G}_{\text{maj}}$has conflict edges, $\mathcal{G}_{\text{cons}}$is not chordal. Therefore, I used the method described in Section 5.4 to obtain extendable graphs $\mathcal{G}'_{\text{cons}}$and $\mathcal{G}'_{\text{maj}}$, respectively. These are shown in Figure 6.3 and Figure 6.4. The node colouring represents the results of Lockless and Ranganathan [169]: They report that the yellow nodes are strongly coupled with the green node (position 372 here, position 77 in [169]).

The low density of $\mathcal{G}_{\text{cons}}$is due to the modification to make the graph extendible. Both graphs initially had a degree of 2.61, not too far from the target value of $d = 2$. Since conflict edges in a way serve as wildcard edges in the modification process described in Algorithm 2, much fewer edges needed to be removed from $\mathcal{G}'_{\text{maj}}$than from $\mathcal{G}'_{\text{cons}}$.

Due to the sparsity of $\mathcal{G}'_{\text{cons}}$, it consists of many separate connected components, which – like in the previous section – can be interpreted as clusters. These are visualised in Figure 6.5. In this case, the clusters are less locally constrained, which might be connected with the type of data or the protein itself. Although cluster analyses as shown in the previous section are also possible in this setting, we will pursue a more causality-based approach here and use the learned graphs to estimate causal effects.

6.2.2 Causal Effect Estimation

As an extension of causal structure learning, Maathuis, Kalisch, and Bühlmann [172] propose a method to learn causal effects on the basis of a CPDAG, instead of one true DAG. This method is called IDA for "intervention-calculus when the DAG is absent". The basic idea is that it is possible to at least give bounds on the effects in the true DAG, which belongs to the equivalence class represented by the CPDAG. This can be done in the simplest form by enumerating all these DAGs, an evaluating the corresponding effects individually, for example using Pearl's intervention calculus [200]. This leads to a set of effects of different size; however since all effects are eventually based on a limited set parameters of the

model, usually different graphs yield the same size for some of the effects. The analysis thus provides a multiset of effect sizes, which can be aggregated, for example to obtain a lower bound.

In addition to this rather naïve procedure, Maathuis, Kalisch, and Bühlmann also propose a fast alternative that avoids the exhaustive enumeration and is still able to determine the set of effect sizes that the Markov equivalent DAGs entail, albeit without being able to count to how many of the DAGs each one of the effect sizes corresponds.

Applications of the method can be found, for example, in [171] and [198]. Recently, Pensar et al. [209] and Castelletti and Consonni [37] proposed Bayesian adaptations of the method. A related, SAT-based method is presented in [141].

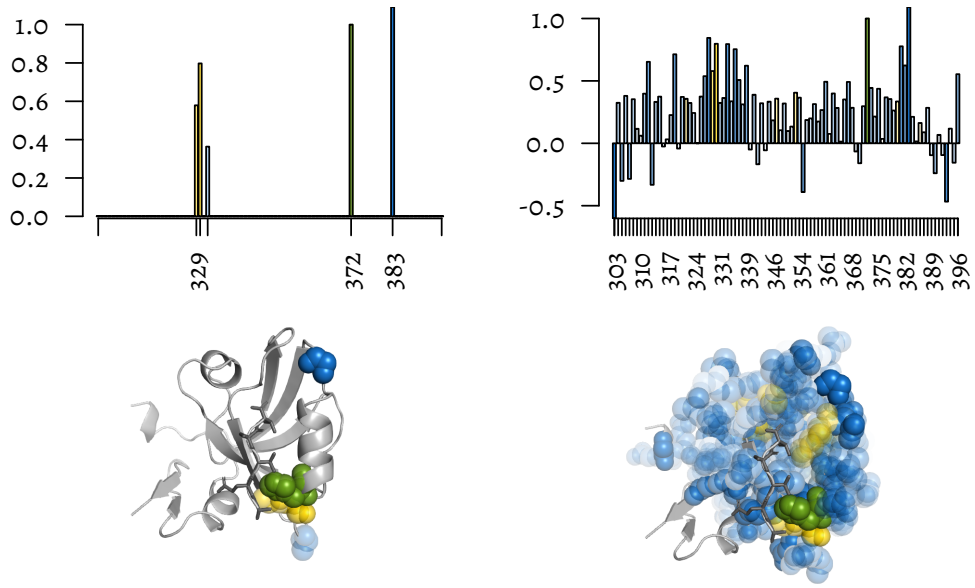


Figure 6.6: Comparison of the new IDA-adaptation that resets causally implausible effects to zero (left) and the original method (right): size of the estimated effects of an intervention in position 372 on all other positions in the protein. In particular in the 3D model (bottom), effect sizes are represented in terms of the opacity of each position. Colours highlight the intervention position (green) and the positions that are found to be coupled with this position by Lockless and Ranganathan [169] (yellow).

Using IDA, it is possible to perform "theoretical experiments" with the considered variables. For example, the mutation conducted in [169] can be modelled by computing the causal effects of position 372 based on the graph that was learned by the PC algorithm.

However, the results produced by IDA can be implausible in one regard, namely that it usually yields causal effects even if no causal path exists. The principle of causal modelling with DAGs is that directed edges represent causation. Consequently, a sequence of directed edges, a directed path is necessary for a variable to have a causal effect on another variable. (This is not to be confused with d-separation which assesses dependence, but not causation.) In particular, a variable cannot have a causal effect on a variable in a different connected component of the graph, which typical for effects estimated by IDA.

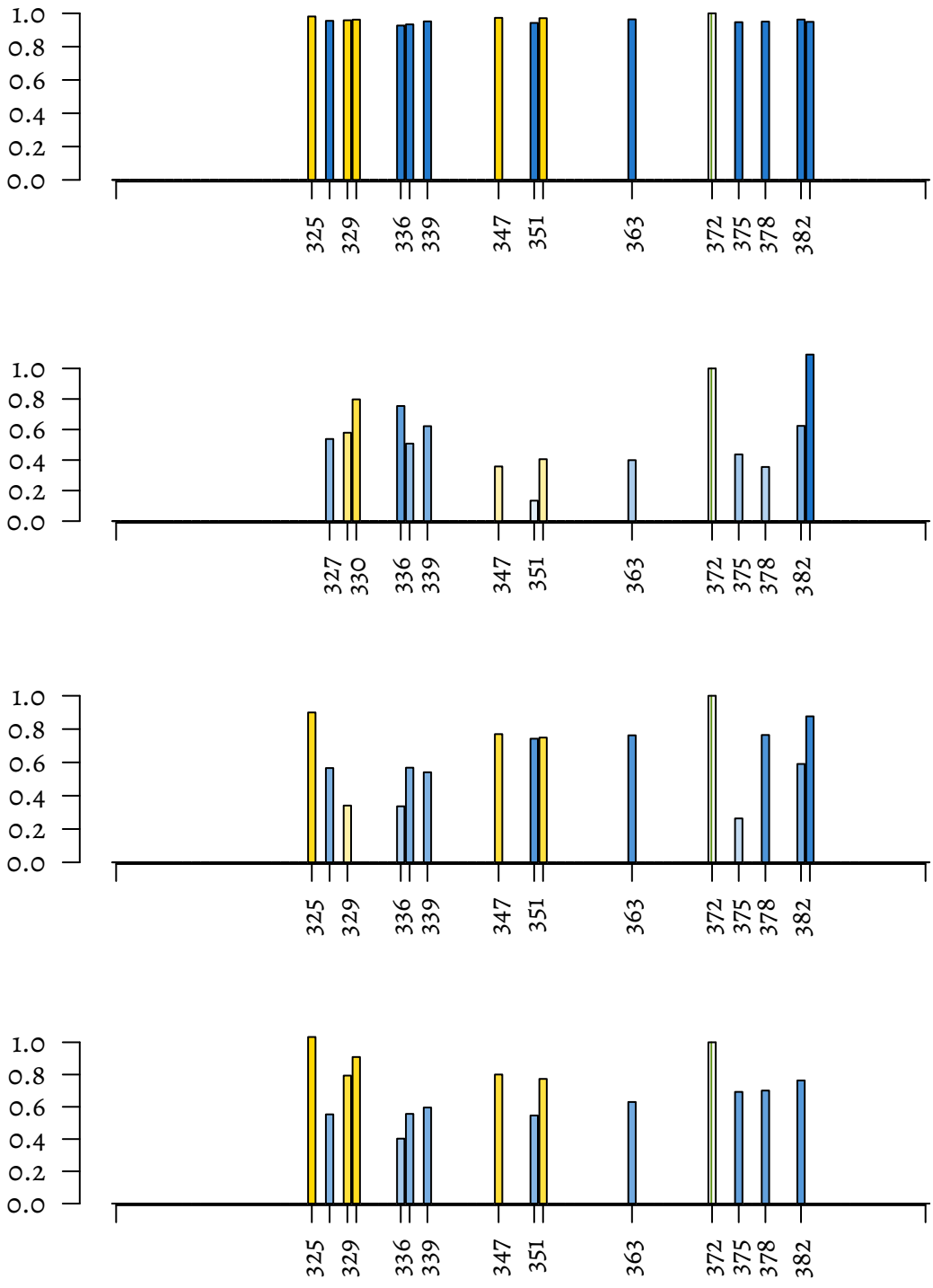


Figure 6.7: Effects of position 372 as determined by IDA when resetting effects that are not causally justified given the graph. The intervention position is marked in green, the yellow positions are found to be coupled with this position by Lockless and Ranganathan [169].

The implausible effects are due to the local character of the computation of IDA. The only information that is extracted from the graph are the parent sets of each node, which in isolation do not provide information about connected components and causal paths. However, the effects for non-connected nodes are difficult to interpret causally and can rather be seen as direct conclusions from the data, which primarily only provide information about correlation, not causation.

I therefore set all causal effects that were estimated by ida, but for which no corresponding causal path exists, to zero. Figure 6.6 shows that this makes a huge difference in practice. In the Figure, the causal effects of the above mentioned position 372 are shown for $\mathcal{G}_{\text{maj}}$ – with the adapted version of IDA (left) and the original method (right).
For $\mathcal{G}'_{\text{cons}}$, IDA yields 4 different effect sizes for a theoretical intervention on position 372, which are shown as bars in Figure 6.7 and visualised in the tertiary structure of the protein in Figure 6.8. Although the results do not perfectly reproduce what was found by Lockless and Ranganathan [169], they seem very plausible for two reasons. On the one hand, we can see an effect on many of the positions that Lockless and Ranganathan identify as coupled with position 372. On the other hand, the spacial structure of the positions that position 372 affects according to the analysis presented here has a reasonable spacial structure and is similar to a functional sector identified by Halabi et al. [119] with different but related evolutionary data (see the red sector in Figure 7A).

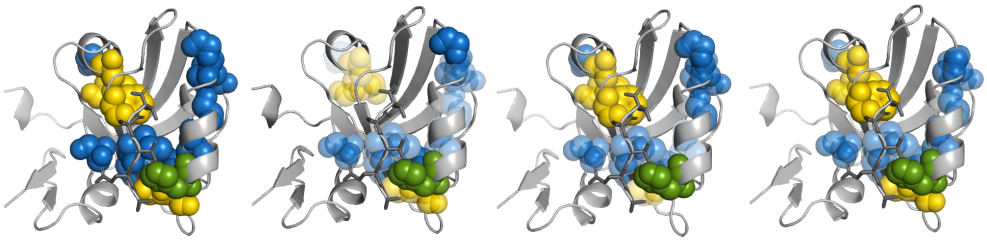


Figure 6.8: Effects of position 372 as determined by IDA when resetting effects that are not causally justified given the graph. The opacity of a node corresponds to the size of the effect of position 372 on this node. The intervention position 372 is marked in green, the yellow positions are found to be coupled with this position by Lockless and Ranganathan [169]; all other positions are coloured blue.

6.3 Conclusion

In this chapter I have shown how causal structure learning, in particular the PC algorithm, can be applied to different types of data in order to gain a better understanding of energy transfer in proteins and thus protein function.

In the analysis some of the problems that were discussed from a more theoretical perspective in the previous chapters came up, and some of the methods presented in the previous chapters have proven useful, in particular the model for tuning the parameter α (see Section 5.3.2) and Algorithm 2 for obtaining an extendible graph on the basis of the output of the PC algorithm.

Learning a causal structure is only the first step in causal analysis. Although the graph itself certainly can provide the basis for different types of analyses, including the identification of clusters that was of interest in our context, a causal structure provides specific causal information that can be used to conduct "theoretical experiments". When the true causal structure is not known beforehand but learned from data, the non-identifiably of the true graph entails uncertainty about the effects. Nevertheless, it is possible to obtain bounds on effect sizes, and sometimes even more precise estimates. The most popular methods for these kind of analysis is IDA [172].

Both on the basis of the causal structure and in terms of effects estimated with IDA, I was able to reproduce and complement published results on sectors of allosteric networks in proteins. This can be seen as a proof of concept and encourages future applications of causal methods in the field of protein function.

7

Conclusion

Causal modelling allows to transcend the world of association and thus facilitates the prediction of effects of interventions and counterfactuals. Surprisingly, such analyses are possible from purely observational data, albeit with some limitations. The analysis is then typical subdivided into two steps: Firstly, learning the causal structure that describes the relationships between the variables, as well as the parameters of the model and secondly, estimating effects on the basis of this model. This thesis is almost exclusively concerned with the first step of the analysis, causal structure learning. For this, the canonical PC algorithm is studied. This algorithm was first proposed by Spirtes et al. [263] and made popular for practical applications by [148]. In this thesis the algorithm and its variations, in particular the order-independent version proposed by Colombo and Maathuis [52] are evaluated. The order-dependence of the original version of the PC algorithm is considered in various aspects in Sections 4.2 and4.3, with new results concerning the oracle version and the positive effect of topologically sorted input variables.

Another pervading topic is the question of how the value for the tuning parameter of the PC algorithm should be chosen. This tuning parameter is used as the level of significance for the individual conditional independence tests that constitute the interface to the data. In the context of parameter tuning, the accuracy of the estimated graph (in particular Section 5.3.2), but also its density (Section 4.3.4) is considered, and a model is derived with which a good value for α can be computed based on the parameters of the setting, i. e. the number of nodes, the sample size and an estimation of the density of the true graph.

This "black box"-analysis is complemented by a supervised empirical study of the results of the conditional independence test that is used, Fisher's Z-test. This yields a completely different perspective on the choice of α. The main finding of the analyses is that it is often advisable to choose a value that is much larger than typical values used in the literature, for example $\alpha = 0.01$. This fact is due to the symmetry of the question that the test is applied to answer: Should there be an edge between a pair of nodes or not? It is therefore not generally advisable to choose the level of significance as low as possible, as usual in statistical testing when an asymmetric setting is considered. (This is usually given by the null hypothesis describing the norm and the alternative hypothesis an abnormality.) In the case of the PC algorithm, α should instead be chosen in a way that yields a good balance between the α and β error.

The main problem considered in this thesis is the type of output that the PC algorithm yields. For subsequent causal methods it is usually assumed that the PC algorithm outputs a CPDAG. Such a graph represents a class of DAGs that is principally identifiable from

observational data. In Section 4.3, the fraction of graphs returned by the PC algorithm, that actually are CPDAGs is determined in various settings. It is often close to zero for denser graphs. The question of the accuracy of the PC-estimated graph thus generalises to the question of validity.

There are various reasons for a graph to be invalid as representation of a causal model. These "symptoms" of invalidity are classified in Section 5.1 (see Figure 5.1) and traced back to possible errors during the computation or violations of the assumptions. One particular problem is the occurrence of conflicting edge directions, which can be made explicit a special type of edges that are called conflict edges here. A general characterisation of the output of the PC algorithm is proposed in terms of the class of general partially directed graphs (GPDGs). In addition to directed and undirected edges, such graphs can contain conflict edges.

Although they are modelled in the implementation of the PC algorithm in the R package `pcalg`, conflict edges are not extensively treated in the literature and the only currently available algorithm to remove them from a graph is by randomly brute-forcing all possible edge directions until a valid graph is found (option `retry` in the `pcalg` implementation). In Section 5.4.1 a new algorithm is presented to systematically extend a GPDG by considering all possible directions for the conflict edges in polynomial time. Based on this algorithm, a general procedure for obtaining a valid graph that is similar to the PC-estimated graph is given in Algorithm 2.

Finally in Chapter 6, it is shown how causal structure learning can be applied to characterise allosteric communication in proteins. As proteins are involved in a wide range of biological processes and protein function often works through allosteric effects, this idea has a variety of applications throughout biology and medicine.

By considering the amino acid positions in a protein as random variables and computing a causal structure between them, the network of energetic communication in the protein can be modelled from different types of data. Based on this, one can identify functional sectors of the protein as connected components or clusters of the graph. On top of that, in terms of interventions it is possible to predict the effects of experiments, for example mutations of individual positions.

This shows the practical value of the PC algorithm despite the problems that were identified in the previous chapters, as well as of the new methods presented in this thesis. Hopefully it will help researchers to understand the reasons for protein dysfunction or the mechanisms of the infection of pathogens and thus facilitate the design of new drugs. Moreover, proteins can help solve problems in other domains, for example provide new sources of renewable energy. Since the amount of available observational data is likely to increase in the future, causal structure learning will probably become feasible in more and more domains and yield more robust results. Causal inference might thus help us to limit time consuming, expensive and possibly unethical experiments to a minimum and revolutionise the way scientific advancement is accomplished.

Bibliography

[1] Acid, S. and de Campos, L. M. Searching for Bayesian Network Structures in the Space of Restricted Acyclic Partially Directed Graphs. In: *Journal of Artificial Intelligence Research* 18:445–490, 2003 (CITED ON PAGES 54, 64, 65, 67).

[2] Adel, T. and de Campos, C. P. Learning Bayesian Networks with Incomplete Data by Augmentation. In: *AAAI-17 – Proceedings of the Thirty-first Conference on Artificial Intelligence*. 2017, pp. 1684–1690 (CITED ON PAGE 61).

[3] Ahn, Y.-Y., Bagrow, J. P., and Lehmann, S. Link Communities Reveal Multiscale Complexity in Networks. In: *Nature* 466(7307):761–764, 2010 (CITED ON PAGES 67, 146).

[4] Akaike, H. Information Theory and an Extension of the Maximum Likelihood Principle. In: *Proceedings of the Second International Symposium on Information Theory*. 1973, pp. 267–281 (CITED ON PAGE 62).

[5] Albert, M., Laurent, B., Marrel, A., and Meynaoui, A. Adaptive Test of Independence Based on HSIC Measures. arXiv:1902.06441. 2019 (CITED ON PAGE 13).

[6] Albert, R. and Barabási, A.-L. Statistical Mechanics of Complex Networks. In: *Reviews of Modern Physics* 74(1):47, 2002 (CITED ON PAGE 67).

[7] Aliferis, C. F., Statnikov, A., Tsamardinos, I., Mani, S., and Koutsoukos, X. D. Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification Part I: Algorithms and Empirical Evaluation. In: *Journal of Machine Learning Research* 11(1):171–234, 2010 (CITED ON PAGES 50, 51).

[8] Aliferis, C. F., Statnikov, A., Tsamardinos, I., Mani, S., and Koutsoukos, X. D. Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification Part II: Analysis and Extensions. In: *Journal of Machine Learning Research* 11(1):235–284, 2010 (CITED ON PAGE 50).

[9] Aliferis, C. F., Tsamardinos, I., and Statnikov, A. R. HITON: A Novel Markov Blanket Algorithm for Optimal Variable Selection. In: *AMIA 2003, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 8-12, 2003*. 2003 (CITED ON PAGE 50).

[10] Alonso-Barba, J. I., Gámez, J. A., Puerta, J. M., et al. Scaling up the Greedy Equivalence Search Algorithm by Constraining the Search Space of Equivalence Classes. In: *International Journal of Approximate Reasoning* 54(4):429–451, 2013 (CITED ON PAGE 56).

[11] Andersson, S. A., Madigan, D., and Perlman, M. D. A Characterization of Markov Equivalence Classes for Acyclic Digraphs. In: *The Annals of Statistics* 25(2):505–541, 1997 (CITED ON PAGES 15, 21, 23–25, 49, 117).

[12] Aoto, P. C., Martin, B. T., and Wright, P. E. NMR Characterization of Information Flow and Allosteric Communities in the MAP Kinase p38γ. In: *Scientific Reports* 6:28655, 2016 (CITED ON PAGES 146–148).

[13] Aragam, B. and Zhou, Q. Concave Penalized Estimation of Sparse Gaussian Bayesian Networks. In: *The Journal of Machine Learning Research* 16(1):2273–2328, 2015 (CITED ON PAGES 56, 65, 68, 121).

[14] Baba, K., Shibata, R., and Sibuya, M. Partial Correlation and Conditional Correlation as Measures of Conditional Independence. In: *Australian & New Zealand Journal of Statistics* 46(4):657–664, 2004 (CITED ON PAGE 12).

[15] Barabási, A.-L. and Albert, R. Emergence of Scaling in Random Networks. In: *Science* 286(5439):509–512, 1999 (CITED ON PAGE 67).

[16] Bartlett, M. and Cussens, J. Integer Linear Programming for the Bayesian Network Structure Learning Problem. In: *Artificial Intelligence* 244:258–271, 2017 (CITED ON PAGES 57, 58).

[17] Beal, M. J. and Ghahramani, Z. The Variational Bayesian EM Algorithm for Incomplete Data: With Application to Scoring Graphical Model Structures. In: *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*. 453-464. 2003, p. 210 (CITED ON PAGE 61).

[18] Berkson, J. Limitations of the Application of Fourfold Table Analysis to Hospital Data. In: *Biometrics Bulletin* 2(3):47–53, 1946 (CITED ON PAGE 37).

[19] Berkson, J. Smoking and Lung Cancer: Some Observations on Two Recent Reports. In: *Journal of the American Statistical Association* 53(281):28–38, 1958 (CITED ON PAGE 1).

[20] Berkson, J. The Statistical Study of Association between Smoking and Lung Cancer. In: *Proceedings of Staff Meetings of the Mayo Clinic*. Vol. 30. 15. 1955, pp. 319–48 (CITED ON PAGE 1).

[21] Bessler, D. A. and Lee, S. Money and prices: US data 1869–1914 (a study with directed graphs). In: *Empirical Economics* 27(3):427–446, 2002 (CITED ON PAGE 59).

[22] Bongers, S., Forré, P., Peters, J., Schölkopf, B., and Mooij, J. M. Foundations of Structural Causal Models with Cycles and Latent Variables. arXiv:161a1.06221. 2020 (CITED ON PAGES 29, 38, 40).

[23] Bongers, S. and Mooij, J. M. From Random Differential Equations to Structural Causal Models: The Stochastic Case. arXiv:1803.08784. 2018 (CITED ON PAGE 38).

[24] Bouckaert, R. R. Bayesian Belief Networks: From Construction to Inference. PhD thesis. University of Utrecht, 1995 (CITED ON PAGE 53).

[25] Bouckaert, R. R. Probabilistic Network Construction Using the Minimum Description Length Principle. In: *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*. Springer. 1993, pp. 41–48 (CITED ON PAGE 62).

[26] Bouckaert, R. R. Properties of Bayesian Belief Network Learning Algorithms. In: *UAI-94 – Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*. 1994, pp. 102–109 (CITED ON PAGE 46).

[27] Bozdogan, H. Model Selection and Akaike's information Criterion (AIC): The General Theory and Its Analytical Extensions. In: *Psychometrika* 52(3):345–370, 1987 (CITED ON PAGE 62).

[28] Bravais, A. *Analyse mathématique sur les probabilités des erreurs de situation d'un point*. Impr. Royale, 1844 (CITED ON PAGE 11).

[29] Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N., Scott, S. L., et al. Inferring Causal Impact Using Bayesian Structural Time-Series Models. In: *Annals of Applied Statistics* 9(1):247–274, 2015 (CITED ON PAGE 59).

[30] Bromberg, F. and Margaritis, D. Improving the Reliability of Causal Discovery from Small Data Sets Using Argumentation. In: *Journal of Machine Learning Research* 10(2), 2009 (cited on page 127).

[31] Buntine, W. L. A Guide to the Literature on Learning Probabilistic Networks from Data. In: *IEEE Transactions on Knowledge and Data Engineering* 8(2):195–210, 1996 (cited on pages 44, 121).

[32] Buntine, W. L. Theory Refinement on Bayesian Networks. In: *UAI-91 – Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 1991, pp. 52–60 (cited on pages 44, 63).

[33] Campos, L. M. de and Huete, J. F. A New Approach for Learning Belief Networks Using Independence Criteria. In: *International Journal of Approximate Reasoning* 24(1):11–37, 2000 (cited on pages 50, 74, 79).

[34] Cano, A., Gómez-Olmedo, M., and Moral, S. A Score Based Ranking of the Edges for the PC Algorithm. In: *PGM-2008 – Proceedings of the Fourth International Conference on Probabilistic Graphical Models*. 2008, pp. 41–48 (cited on page 76).

[35] Carvalho, A. S. M. d. *Scoring Functions for Learning Bayesian Networks*. Tech. rep. 2009 (cited on page 62).

[36] Casella, G. and Berger, R. L. *Statistical Inference*. Cengage Learning, 2021 (cited on page 7).

[37] Castelletti, F. and Consonni, G. Bayesian Inference of Causal Effects from Observational Data in Gaussian Graphical Models. In: *Biometrics* 77(1):136–149, 2021 (cited on pages 75, 153).

[38] Chechetka, A. and Guestrin, C. Efficient Principled Learning of Thin Junction Trees. In: *Advances in Neural Information Processing Systems* 20:273–280, 2007 (cited on page 56).

[39] Chen, J. and Chen, Z. Extended Bayesian Information Criteria for Model Selection with Large Model Spaces. In: *Biometrika* 95(3):759–771, 2008 (cited on page 64).

[40] Chickering, D. M. A Transformational Characterization of Equivalent Bayesian Network Structures. In: *UAI-95 – Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 1995, pp. 87–98 (cited on pages 22, 24).

[41] Chickering, D. M. Learning Bayesian Networks is NP-Complete. In: *Learning From Data: Artificial Intelligence and Statistics V*. 1996, pp. 121–130 (cited on page 46).

[42] Chickering, D. M. Learning Equivalence Classes of Bayesian Network Structures. In: *UAI-96 – Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. 1996, pp. 150–157 (cited on page 54).

[43] Chickering, D. M. Learning Equivalence Classes of Bayesian Network Structures. In: *Journal of Machine Learning Research* 2(Feb):445–498, 2002 (cited on pages 21, 23, 26, 54, 111).

[44] Chickering, D. M. Optimal Structure Identification with Greedy Search. In: *Journal of Machine Learning Research* 3(Nov):507–554, 2002 (cited on page 62).

[45] Chickering, D. M. Statistically Efficient Greedy Equivalence Search. In: *UAI-20 – Proceedings of the Thirty-sixth Conference on Uncertainty in Artificial Intelligence*. Vol. 124. 2020, pp. 241–249 (cited on page 55).

[46] Chickering, D. M., Heckerman, D., and Meek, C. Large-sample learning of Bayesian networks is NP-hard. In: *Journal of Machine Learning Research* 5, 2004 (CITED ON PAGES 46, 48).

[47] Chickering, D. M. and Meek, C. Finding Optimal Bayesian Networks. In: *UAI-02 – Proceedings of the Eighteenth International Conference on Uncertainty in Artificial Intelligence*. 2002, pp. 94–102 (CITED ON PAGE 55).

[48] Chickering, D. M. and Meek, C. Selective Greedy Equivalence Search: Finding Optimal Bayesian Networks Using a Polynomial Number of Score Evaluations. In: *UAI-15 – Proceedings of the Thirty-first Conference on Uncertainty in Artificial Intelligence*. 2015, pp. 211–219 (CITED ON PAGES 55, 56).

[49] Chow, C. K. and Liu, C. N. Approximating Discrete Probability Distributions with Dependence Trees. In: *IEEE Transactions on Information Theory* 14(3):462–467, 1968 (CITED ON PAGES 56, 62).

[50] Chu, T. and Glymour, C. Search for Additive Nonlinear Time Series Causal Models. In: *Journal of Machine Learning Research* 9(5):967–991, 2008 (CITED ON PAGE 59).

[51] Claassen, T., Mooij, J. M., and Heskes, T. Learning Sparse Causal Models is not NP-hard. In: *UAI-13 – Proceedings of the Twenty-ninth Conference on Uncertainty in Artificial Intelligence*. 2013 (CITED ON PAGES 46, 50).

[52] Colombo, D. and Maathuis, M. H. Order-Independent Constraint-Based Causal Structure Learning. In: *The Journal of Machine Learning Research* 15(1):3741–3782, 2014 (CITED ON PAGES 67, 76, 83, 88, 99, 111, 121, 133, 157).

[53] Colombo, D., Maathuis, M. H., Kalisch, M., and Richardson, T. S. Learning High-dimensional Directed Acyclic Graphs with Latent and Selection Variables. In: *The Annals of Statistics*:294–321, 2012 (CITED ON PAGES 37, 50).

[54] Constantinou, A. C., Liu, Y., Chobtham, K., Guo, Z., and Kitson, N. K. Large-scale Empirical Validation of Bayesian Network Structure Learning Algorithms with Noisy Data. In: *International Journal of Approximate Reasoning* 131:151–188, 2021 (CITED ON PAGES 47, 53, 61, 65, 66, 122, 133).

[55] Cooper, G. F. and Herskovits, E. A Bayesian Method for the Induction of Probabilistic Networks from Data. In: *Machine Learning* 9(4):309–347, 1992 (CITED ON PAGES 52, 56, 59).

[56] Cooper, G. F. and Yoo, C. Causal Discovery from a Mixture of Experimental and Observational Data. In: *UAI-99 – Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. 1999, pp. 116–125 (CITED ON PAGE 60).

[57] Córdoba, I., Garrido-Merchán, E. C., Hernández-Lobato, D., Bielza, C., and Larranaga, P. Bayesian Optimization of the PC Algorithm for Learning Gaussian Bayesian Networks. In: *Conference of the Spanish Association for Artificial Intelligence*. Springer. 2018, pp. 44–54 (CITED ON PAGE 123).

[58] Cowell, R. G., Dawid, P., Lauritzen, S. L., and Spiegelhalter, D. J. *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks*. Springer Science & Business Media, 2006 (CITED ON PAGE 17).

[59] Cox, D. R. and Wermuth, N. *Multivariate dependencies: Models, analysis and interpretation*. Chapman and Hall/CRC, 2014 (CITED ON PAGE 28).

[60] Cruz-Ramírez, N., Acosta-Mesa, H.-G., Barrientos-Martínez, R.-E., and Nava-Fernández, L.-A. How Good Are the Bayesian Information Criterion and the Minimum Description Length Principle for model selection? A Bayesian net-

work analysis. In: *Mexican International Conference on Artificial Intelligence*. Springer. 2006, pp. 494–504 (cited on page 62).

[61] Cugnata, F., Kenett, R. S., and Salini, S. Bayesian networks in survey data: Robustness and sensitivity issues. In: *Journal of Quality Technology* 48(3):253–264, 2016 (cited on page 47).

[62] Cui, R., Groot, P., and Heskes, T. Copula PC Algorithm for Causal Discovery from Mixed Data. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2016, pp. 377–392 (cited on pages 13, 39, 60, 75).

[63] Cui, R., Groot, P., and Heskes, T. Learning causal structure from mixed data with missing values using Gaussian copula models. In: *Statistics and Computing*:1–23, 2018 (cited on pages 39, 61, 75, 121).

[64] Cussens, J. Bayesian Network Learning with Cutting Planes. In: *UAI-11 – Proceedings of the Twenty-seventh Conference on Uncertainty in Artificial Intelligence*. 2011, pp. 153–160 (cited on pages 57, 58).

[65] Darwiche, A. *Modeling and Reasoning with Bayesian Networks*. Cambridge university press, 2009 (cited on page 17).

[66] Dasgupta, S. Learning Polytrees. In: *UAI-99 – Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. 1999, pp. 134–141 (cited on page 46).

[67] Dash, D. and Drużdżel, M. J. A Hybrid Anytime Algorithm for the Construction of Causal Models From Sparse Data. In: *UAI-99 – Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. 1999, pp. 142–149 (cited on pages 57, 76, 99, 121).

[68] Dawid, A. P. Conditional Independence for Statistical Operations. In: *The Annals of Statistics*:598–617, 1980 (cited on page 8).

[69] Dawid, A. P. Conditional Independence in Statistical Theory. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 41(1):1–15, 1979 (cited on page 8).

[70] de Brito, C. E. F. Graphical Methods for Identification in Structural Equation Models. PhD thesis. University of California, Los Angeles, 2004 (cited on page 32).

[71] de Brito, C. E. F. Instrumental Sets. In: *Heuristics, Probability and Causality – A Tribute to Judea Pearl*. 2010, pp. 295–307 (cited on page 32).

[72] de Campos, C. P. and Ji, Q. Efficient Structure Learning of Bayesian Networks using Constraints. In: *The Journal of Machine Learning Research* 12:663–689, 2011 (cited on page 54).

[73] de Campos, L. M. A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests. In: *Journal of Machine Learning Research* 7(Oct):2149–2187, 2006 (cited on page 62).

[74] de Campos, L. M., Fernandez-Luna, J. M., Gámez, J. A., and Puerta, J. M. Ant Colony Optimization for Learning Bayesian Networks. In: *International Journal of Approximate Reasoning* 31(3):291–311, 2002 (cited on page 53).

[75] De Jongh, M. and Drużdżel, M. J. A Comparison of Structural Distance Measures for Causal Bayesian Network Models. In: *Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science series*:443–456, 2009 (cited on pages 65, 88).

[76] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 39(1):1–22, 1977 (CITED ON PAGES 44, 61).

[77] Diestel, R. *Graph Theory*. 4th ed. Vol. 173. Springer, 2012 (CITED ON PAGES 13, 17).

[78] Doll, R. Uncovering the Effects of Smoking: Historical Perspective. In: *Statistical methods in medical research* 7(2):87–117, 1998 (CITED ON PAGE 1).

[79] Dor, D. and Tarsi, M. *A Simple Algorithm to Construct a Consistent Extension of a Partially Oriented Graph*. Tech. rep. R-185. Cognitive Systems Laboratory, UCLA, 1992 (CITED ON PAGES 4, 26, 111, 140, 142).

[80] Downey, R. G. and Fellows, M. R. *Parameterized Complexity*. Springer, 2012 (CITED ON PAGE 56).

[81] Duncan, O. D. *Introduction to Structural Equation Models*. 1st ed. Academic Press, 1975 (CITED ON PAGE 28).

[82] Eberhardt, F. Introduction to the Foundations of Causal Discovery. In: *International Journal of Data Science and Analytics* 3(2):81–91, 2017 (CITED ON PAGE 36).

[83] Eberhardt, F., Hoyer, P. O., and Scheines, R. Combining Experiments to Discover Linear Cyclic Models with Latent Variables. In: *AISTATS-2010 – Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. 2010, pp. 185–192 (CITED ON PAGES 57, 60).

[84] Eddington, A. S. *The Mathematical Theory of Relativity*. The University Press, 1923 (CITED ON PAGE 45).

[85] Edwards, D. *Introduction to Graphical Modelling*. Springer Science & Business Media, 2000 (CITED ON PAGE 28).

[86] Eén, N. and Sörensson, N. An Extensible SAT-Solver. In: *SAT-2003 – Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2003, pp. 502–518 (CITED ON PAGE 58).

[87] Eichler, M. Causal Inference in Time Series Analysis. In: *Causality*. 2012. Chap. 22, pp. 327–354 (CITED ON PAGE 59).

[88] Elidan, G. and Gould, S. Learning Bounded Treewidth Bayesian Networks. In: *Journal of Machine Learning Research* 9(12), 2008 (CITED ON PAGE 56).

[89] Entner, D. and Hoyer, P. O. On Causal Discovery From Time Series Data Using FCI. In: *PGM-2010 – Proceedings of the Fifth International Conference on Probabilistic Graphical Models*. 2010, pp. 121–128 (CITED ON PAGE 59).

[90] Fernández, A., Nielsen, J. D., and Salmerón, A. Learning Bayesian Networks for Regression from Incomplete Databases. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 18(01):69–86, 2010 (CITED ON PAGE 61).

[91] Fisher, F. M. *The Identification Problem in Econometrics*. McGraw-Hill, 1966 (CITED ON PAGE 21).

[92] Fisher, R. A. Cancer and smoking. In: *Nature* 182(4635):596–596, 1958 (CITED ON PAGE 1).

[93] Fisher, R. A. Dangers of cigarette-smoking. In: *British Medical Journal* 2(5039):297, 1957 (CITED ON PAGES 1, 2).

[94] Foraita, R., Friemel, J., Günther, K., Behrens, T., Bullerdiek, J., Nimzyk, R., Ahrens, W., and Didelez, V. Causal Discovery of Gene Regulation with Incomplete Data. In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 183(4):1747–1775, 2020 (CITED ON PAGES 61, 75, 122).

[95] Forré, P. and Mooij, J. M. Constraint-based Causal Discovery for Non-Linear Structural Causal Models with Cycles and Latent Confounders. In: *UAI-18 – Proceedings of the Thirty-fourth Conference on Uncertainty in Artificial Intelligence*. 2018, pp. 269–278 (cited on pages 39, 40, 58, 60).

[96] Forré, P. and Mooij, J. M. Markov Properties for Graphical Models with Cycles and Latent Variables. arXiv:1710.08775. 2017 (cited on page 38).

[97] Foygel, R. and Drton, M. Extended Bayesian Information Criteria for Gaussian Graphical Models. In: *NeurIPS-10 – Proceedings of the Twenty-fourth International Conference on Neural Information Processing Systems*. 2010, pp. 604–612 (cited on page 64).

[98] Friedman, N. Learning Belief Networks in the Presence of Missing Values and Hidden Variables. In: *ICML-1997 – Proceedings of the Fourteenth International Conference on Machine Learning*. 1997, pp. 125–133 (cited on page 61).

[99] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. Using Bayesian networks to analyze expression data. In: *Journal of Computational Biology* 7(3-4):601–620, 2000 (cited on page 47).

[100] Friedman, N., Nachman, I., and Pe'er, D. Learning Bayesian Network Structure From Massive Datasets: The "Sparse Candidate" Algorithm. In: *UAI-99 – Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. 1999, pp. 206–215 (cited on pages 47, 57).

[101] Friedman, N. and Yakhini, Z. On the Sample Complexity of Learning Bayesian Networks. In: *UAI-96 – Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. 1996, pp. 274–282 (cited on page 121).

[102] Frot, B., Nandy, P., and Maathuis, M. H. Robust Causal Structure Learning with Some Hidden Variables. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 81(3):459–487, 2019 (cited on pages 37, 51).

[103] Frydenberg, M. The Chain Graph Markov Property. In: *Scandinavian Journal of Statistics*:333–353, 1990 (cited on pages 15, 22).

[104] Fukumizu, K., Gretton, A., Sun, X., and Schölkopf, B. Kernel Measures of Conditional Dependence. In: *NeurIPS-07 – Proceedings of the Twenty-first International Conference on Neural Information Processing Systems*. Vol. 20. 2008 (cited on page 13).

[105] Gao, B. and Cui, Y. Learning Directed Acyclic Graphical Structures with Genetical Genomics Data. In: *Bioinformatics* 31(24):3953–3960, 2015 (cited on pages 65, 75).

[106] Garey, M. R. and Johnson, D. S. *Computers and Intractability – A Guide to the Theory of NP Completeness*. W. H. Freeman, 1979 (cited on page 45).

[107] Geiger, D. and Heckerman, D. Learning Gaussian Networks. In: *UAI-94 – Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 1994, pp. 235–243 (cited on page 64).

[108] Geiger, D. and Pearl, J. Logical and Algorithmic Properties of Independence and Their Application to Bayesian Networks. In: *Annals of Mathematics and Artificial Intelligence* 2(1):165–178, 1990 (cited on page 8).

[109] Geiger, D. and Pearl, J. On the Logic of Causal Models. In: *UAI-88 – Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*. 1990, pp. 3–14 (cited on page 17).

[110] Geiger, D., Verma, T., and Pearl, J. Identifying Independence in Bayesian Networks. In: *Networks* 20(5):507–534, 1990 (cited on page 17).

[111] Ghassami, A., Salehkaleybar, S., Kiyavash, N., and Bareinboim, E. Budgeted Experiment Design for Causal Structure Learning. In: *ICML-2018 – Proceedings of the Thirty-fifth International Conference on Machine Learning*. PMLR. 2018, pp. 1724–1733 (CITED ON PAGE 26).

[112] Gillispie, S. B. and Perlman, M. D. The Size Distribution for Markov Equivalence Classes of Acyclic Digraph Models. In: *Artificial Intelligence* 141(1-2):137–155, 2002 (CITED ON PAGE 26).

[113] Glover, F. and Laguna, M. *Tabu Search*. Kluwer Academic Publishers, 1997 (CITED ON PAGE 53).

[114] Glymour, C. N. *The Mind's Arrows: Bayes Nets and Graphical Causal Models in psychology*. MIT Press, 2001 (CITED ON PAGE 76).

[115] Granger, C. W. J. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. In: *Econometrica, Journal of the Econometric Society* 37(3):424–438, 1969 (CITED ON PAGE 59).

[116] Gretton, A., Fukumizu, K., Teo, C., Song, L., Schölkopf, B., and Smola, A. J. A Kernel Statistical Test of Independence. In: *NeurIPS-07 – Proceedings of the Twenty-first International Conference on Neural Information Processing Systems*. Vol. 20. 2008 (CITED ON PAGE 13).

[117] Gu, J. and Zhou, Q. Learning Big Gaussian Bayesian Networks: Partition, Estimation and Fusion. In: *Journal of Machine Learning Research* 21(158):1–31, 2020 (CITED ON PAGES 64, 67, 68, 75).

[118] Haavelmo, T. The statistical implications of a system of simultaneous equations. In: *Econometrica, Journal of the Econometric Society*:1–12, 1943 (CITED ON PAGE 28).

[119] Halabi, N., Rivoire, O., Leibler, S., and Ranganathan, R. Protein Sectors: Evolutionary Units of Three-Dimensional Structure. In: *Cell* 138(4):774–786, 2009 (CITED ON PAGE 155).

[120] Halpern, J. Y. Axiomatizing causal reasoning. In: *Journal of Artificial Intelligence Research* 12:317–337, 2000 (CITED ON PAGE 38).

[121] Hammersley, J. M. and Clifford, P. Markov Fields on Finite Graphs and Lattices. 1971 (CITED ON PAGE 27).

[122] Harris, N. and Drton, M. PC Algorithm for Nonparanormal Graphical Models. In: *The Journal of Machine Learning Research* 14(1):3365–3383, 2013 (CITED ON PAGES 13, 39, 75, 88, 111, 121–123, 133).

[123] Hauser, A. and Bühlmann, P. Characterization and Greedy Learning of Interventional Markov equivalence Classes of Directed Acyclic Graphs. In: *Journal of Machine Learning Research* 13:2409–2464, 2012 (CITED ON PAGES 26, 35, 55, 60, 88).

[124] He, Y.-B. and Geng, Z. Active Learning of Causal Networks with Intervention Experiments and Optimal Designs. In: *Journal of Machine Learning Research* 9(Nov):2523–2547, 2008 (CITED ON PAGE 26).

[125] He, Y., Jia, J., and Yu, B. Counting and Exploring Sizes of Markov Equivalence Classes of Directed Acyclic Graphs. In: *The Journal of Machine Learning Research* 16(1):2589–2609, 2015 (CITED ON PAGE 27).

[126] Heckerman, D. A Tutorial on Learning with Bayesian Networks. In: *Learning in Graphical Models*. Vol. 89. 1998, pp. 301–354 (CITED ON PAGE 44).

[127] Heckerman, D., Geiger, D., and Chickering, D. M. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data.pdf. In: *AAAI-94 – Proceedings*

of the Twelfth National Conference on Artificial Intelligence. 1994, pp. 85–96 (cited on page 67).

[128] Heckerman, D., Geiger, D., and Chickering, D. M. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data.pdf. In: *Machine Learning* 20(3):197–243, 1995 (cited on pages 53, 59, 62, 64, 67).

[129] Heinze-Deml, C., Maathuis, M. H., and Meinshausen, N. Causal Structure Learning. In: *Annual Review of Statistics and Its Application* 5:371–391, 2018 (cited on pages 47, 88, 111).

[130] Hernán, M. Á. and Robins, J. M. *Causal Inference – What If*. CRC Press, 2010 (cited on pages 6, 59).

[131] Herskovits, E. and Cooper, G. F. Kutató: An Entropy-driven System for Construction of Probabilistic Expert Systems from Databases. In: *UAI-90 – Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*. 1990, pp. 117–128 (cited on pages 52, 62).

[132] Hesslow, G. Two Notes on the Probabilistic Approach to Causality. In: *Philosophy of Science* 43(2):290–292, 1976 (cited on page 36).

[133] Höffgen, K.-U. Learning and Robust Learning of Product Distributions. In: *COLT-1993 – Proceedings of the Sixth Annual Conference on Computational Learning Theory*. 1993, pp. 77–83 (cited on pages 46, 121).

[134] Holland, P. W. *Causal Inference, Path Analysis and Recursive Structural Equations Models*. Tech. rep. 88-81. Educational Testing Service, 1988 (cited on page 30).

[135] Hoyer, P., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. Nonlinear Causal Discovery with Additive Noise Models. In: *NeurIPS-08 – Proceedings of the Twenty-second International Conference on Neural Information Processing Systems*. 2008, pp. 689–696 (cited on page 39).

[136] Hoyer, P. O., Hyvärinen, A., Scheines, R., Spirtes, P., Ramsey, J., Lacerda, G., and Shimizu, S. Causal discovery of linear acyclic models with arbitrary distributions. In: *UAI-08 – Proceedings of the Twenty-fourth Conference on Uncertainty in Artificial Intelligence*. 2008, pp. 282–289 (cited on page 39).

[137] Hoyer, P. O., Shimizu, S., Kerminen, A. J., and Palviainen, M. Estimation of causal effects using linear non-Gaussian causal models with hidden variables. In: *International Journal of Approximate Reasoning* 49(2):362–378, 2008 (cited on page 39).

[138] Huang, J. and Zhou, Q. Partitioned Hybrid Learning of Bayesian Network Structures. arXiv:2103.12188. 2021 (cited on pages 75, 127).

[139] Hyttinen, A., Eberhardt, F., and Hoyer, P. O. Learning linear cyclic causal models with latent variables. In: *Journal of Machine Learning Research* 13:3387–3439, 2012 (cited on pages 58, 60).

[140] Hyttinen, A., Eberhardt, F., and Järvisalo, M. Constraint-based Causal Discovery: Conflict Resolution with Answer Set Programming. In: *UAI-14 – Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. 2014, pp. 340–349 (cited on pages 58, 60).

[141] Hyttinen, A., Eberhardt, F., and Järvisalo, M. Do-calculus when the True Graph Is Unknown. In: *UAI-15 – Proceedings of the Thirty-first Conference on Uncertainty in Artificial Intelligence*. 2015, pp. 395–404 (cited on pages 58, 153).

[142] Hyttinen, A., Hoyer, P. O., Eberhardt, F., and Järvisalo, M. Discovering Cyclic Causal Models with Latent Variables: A General SAT-Based Procedure. In: *UAI-13 – Proceedings of the Twenty-ninth Conference on Uncertainty in Artificial Intelligence*. 2013 (CITED ON PAGES 57, 58, 60).

[143] Hyttinen, A., Plis, S., Järvisalo, M., Eberhardt, F., and Danks, D. Causal Discovery from Subsampled Time Series Data by Constraint Optimization. In: *PGM-2016 – Proceedings of the Eighth International Conference on Probabilistic Graphical Models*. 2016, pp. 216–227 (CITED ON PAGES 58, 59).

[144] Igual, L. and Seguí, S. *Introduction to Data Science*. Springer, 2017, pp. 1–4 (CITED ON PAGE 65).

[145] Ising, E. Beitrag zur Theorie des Ferromagnetismus. In: *Zeitschrift für Physik* 31(1):253–258, 1925 (CITED ON PAGE 27).

[146] Jaakkola, T. S., Sontag, D. A., Globerson, A., and Meila, M. Learning Bayesian Network Structure Using LP Relaxations. In: *AISTATS-2010 – Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. 2010, pp. 358–365 (CITED ON PAGE 57).

[147] Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabási, A.-L. The Large-scale Organization of Metabolic Networks. In: *Nature* 407(6804):651–654, 2000 (CITED ON PAGE 67).

[148] Kalisch, M. and Bühlmann, P. Estimating High-dimensional Directed Acyclic Graphs with the PC-Algorithm. In: *Journal of Machine Learning Research* 8:613–636, 2007 (CITED ON PAGES 4, 12, 34, 40, 65–67, 71, 72, 75, 81–83, 85, 88, 98, 101, 107, 111, 119, 121, 122, 133, 135, 136, 157).

[149] Kalisch, M., Hauser, A., and Mächler, M. *R-package pcalg: Estimating the Skeleton and Equivalence Class of a DAG*. URL: `https://cran.r-project.org` (visited on 10/02/2019) (CITED ON PAGE 47).

[150] Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., and Bühlmann, P. Causal Inference Using Graphical Models with the R Package pcalg. In: *Journal of Statistical Software* 47(11):1–26, 2012 (CITED ON PAGES 75, 80).

[151] Kano, Y. and Shimizu, S. Causal Inference Using Nonnormality. In: *Proceedings of the International Symposium on Science of Modeling*. 2003, pp. 261–270 (CITED ON PAGE 39).

[152] Karger, D. R. and Srebro, N. Learning Markov Networks: Maximum bounded Tree-width Graphs. In: *SODA-2001 – Proceedings of the Twelfth Annual Symposium on Discrete Algorithms*. 2001, pp. 392–401 (CITED ON PAGE 56).

[153] Karp, R. M. Reducibility among Combinatorial Problems. In: *Complexity of Computer Computations*. 1972, pp. 85–103 (CITED ON PAGE 46).

[154] Kocaoglu, M., Jaber, A., Shanmugam, K., and Bareinboim, E. Characterization and Learning of Causal Graphs with Latent Variables from Soft Interventions. In: *NeurIPS-19 – Proceedings of the Thirty-third International Conference on Neural Information Processing Systems*. 2019, pp. 14346–14356 (CITED ON PAGE 35).

[155] Kočka, T., Bouckaert, R. R., and Studený, M. On Characterizing Inclusion of Bayesian Networks. In: *UAI-01 – Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. 2001, pp. 261–268 (CITED ON PAGE 22).

[156] Koivisto, M. Advances in Exact Bayesian Structure Discovery in Bayesian Networks. In: *UAI-06 – Proceedings of the Twenty-second Conference Conference on Uncertainty in Artificial Intelligence*. 2006 (cited on page 54).

[157] Koivisto, M. and Sood, K. Exact Bayesian structure discovery in Bayesian networks. In: *Journal of Machine Learning Research* 5(May):549–573, 2004 (cited on page 54).

[158] Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009 (cited on pages 27, 28, 30, 34, 35, 39, 41, 43, 47, 61–63, 67, 102, 122).

[159] Koster, J. T. A. Markov Properties of Nonrecursive Causal Models. In: *Annals of Statistics* 24(5):2148–2177, 1996 (cited on page 38).

[160] Lacerda, G., Spirtes, P. L., Ramsey, J. D., and Hoyer, P. O. Discovering Cyclic Causal Models by Independent Components Analysis. In: *UAI-08 – Proceedings of the Twenty-fourth Conference on Uncertainty in Artificial Intelligence*. 2008, pp. 366–374 (cited on page 58).

[161] Lam, W. and Bacchus, F. Learning Bayesian Belief Networks: An Approach Based on the MDL Principle. In: *Computational intelligence* 10(3):269–293, 1994 (cited on page 62).

[162] Larranaga, P., Poza, M., Yurramendi, Y., Murga, R. H., and Kuijpers, C. M. H. Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(9):912–926, 1996 (cited on page 53).

[163] Lauritzen, S. L. *Graphical Models*. Vol. 17. Clarendon Press, 1996 (cited on pages 17, 19, 28).

[164] Lauritzen, S. L., Dawid, A. P., Larsen, B. N., and Leimer, H.-G. Independence Properties of Directed Markov Fields. In: *Networks* 20(5):491–505, 1990 (cited on page 19).

[165] Le, T. D., Hoang, T., Li, J., Liu, L., Liu, H., and Hu, S. A Fast PC Algorithm for High Dimensional Causal Discovery with Multi-Core PCs. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 16(5):1483–1495, 2019 (cited on page 75).

[166] Li, J. and Wang, Z. J. Controlling the False Discovery Rate of the Association/-Causality Structure Learned with the PC Algorithm. In: *Journal of Machine Learning Research* 10(2), 2009 (cited on page 127).

[167] Little, R. J. A. and Rubin, D. B. *Statistical Analysis with Missing Data*. 1st ed. Wiley, 1987 (cited on page 60).

[168] Liu, Z., Malone, B., and Yuan, C. Empirical Evaluation of Scoring Functions for Bayesian Network Model Selection. In: *BMC Bioinformatics* 13(15):S14, 2012 (cited on pages 62, 64).

[169] Lockless, S. W. and Ranganathan, R. Evolutionarily Conserved Pathways of Energetic Connectivity in Protein Families. In: *Science* 286(5438):295–299, 1999 (cited on pages 145, 148–155).

[170] Losiewicz, P. B. Complexity, Ontology, and the Causal Markov Assumption. In: *ACM SIGART Bulletin* 7(4):13–18, 1996 (cited on page 36).

[171] Maathuis, M. H., Colombo, D., Kalisch, M., and Bühlmann, P. Predicting Causal Effects in Large-scale Systems from Observational Data. In: *Nature Methods* 7(4):247, 2010 (cited on pages 75, 122, 153).

[172] Maathuis, M. H., Kalisch, M., and Bühlmann, P. Estimating High-dimensional Intervention Effects from Observational Data. In: *The Annals of Statistics* 37(6A):3133–3164, 2009 (CITED ON PAGES 3, 5, 26, 75, 111, 122, 152, 153, 156).

[173] Madsen, A. L., Jensen, F., Salmerón, A., Langseth, H., and Nielsen, T. D. A Parallel Algorithm for Bayesian Network Structure Learning from Large Data Sets. In: *Knowledge-Based Systems* 117:46–55, 2017 (CITED ON PAGE 75).

[174] Magliacane, S., Claassen, T., and Mooij, J. M. Ancestral Causal Inference. In: *NeurIPS-16 – Proceedings of the Thirtieth International Conference on Neural Information Processing Systems*. 2016, pp. 4466–4474 (CITED ON PAGE 58).

[175] Malinsky, D. and Danks, D. Causal discovery algorithms: A practical guide. In: *Philosophy Compass* 13(1):e12470, 2018 (CITED ON PAGES 121, 122).

[176] Malinsky, D. and Spirtes, P. L. Causal Structure Learning from Multivariate Time Series in Settings with Unmeasured Confounding. In: *CD@KDD 2018 – Proceedings of the 2018 ACM SIGKDD Workshop on Causal Discovery*. Vol. 92. 2018, pp. 23–47 (CITED ON PAGE 59).

[177] Mallagaray, A., Creutznacher, R., Dülfer, J., Mayer, P. H., Grimm, L. L., Orduña, J. M., Trabjerg, E., Stehle, T., Rand, K. D., Blaum, B. S., et al. A Post-translational Modification of Human Norovirus Capsid Protein Attenuates Glycan Binding. In: *Nature Communications* 10(1):1–14, 2019 (CITED ON PAGE 148).

[178] Margaritis, D. Learning Bayesian Network Model Structure From Data. PhD thesis. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2003 (CITED ON PAGE 50).

[179] Margaritis, D. and Thrun, S. *Bayesian network induction via local neighborhoods*. Tech. rep. CMU-CS-99-134. School of Computer Science, Carnegie Mellon University, 1999 (CITED ON PAGE 50).

[180] Massey, J. L. Causal Interpretations of Random Variables. In: *Problemy Peredachi Informatsii* 32(1):131–136, 1996 (CITED ON PAGE 17).

[181] Meek, C. Causal Inference and Causal Explanation with Background Knowledge. In: *UAI-95 – Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. 1995, pp. 403–410 (CITED ON PAGES 16, 22–24, 49, 59).

[182] Meek, C. *Complete Orientation Rules for Patterns*. Tech. rep. CMU-PHIL-61. Carnegie Mellon University, Department of Philosophy, 1995 (CITED ON PAGES 23, 70, 113).

[183] Meek, C. Finding a Path Is Harder than Finding a Tree. In: *Journal of Artificial Intelligence Research* 15:383–389, 2001 (CITED ON PAGE 46).

[184] Meek, C. Graphical Models: Selecting Causal and Statistical Models. PhD thesis. Carnegie Mellon University, 1997 (CITED ON PAGE 54).

[185] Meek, C. Strong Completeness and Faithfulness in Bayesian Networks. In: *UAI-95 – Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. 1995, pp. 411–418 (CITED ON PAGES 17, 37, 115).

[186] Mittag, H.-J. *Statistik: Eine Einführung mit interaktiven Elementen*. Springer-Verlag, 2017 (CITED ON PAGE 11).

[187] Mooij, J. and Heskes, T. Cyclic Causal Discovery from Continuous Equilibrium Data. In: *UAI-13 – Proceedings of the Twenty-ninth Conference on Uncertainty in Artificial Intelligence*. 2013 (CITED ON PAGES 39, 51, 58, 60).

[188] Mooij, J. M., Janzing, D., Heskes, T., and Schölkopf, B. On Causal Discovery with Cyclic Additive Noise Models. In: *NeurIPS-11 – Proceedings of the Twenty-fifth Interna-*

tional Conference on Neural Information Processing Systems. 2011, pp. 639–647 (cited on pages 39, 58).

[189] Mooij, J. M., Magliacane, S., and Claassen, T. Joint Causal Inference from Multiple Contexts. In: *Journal of Machine Learning Research* 21(99):1–108, 2020 (cited on page 60).

[190] Moore, A. W. and Wong, W. Optimal Reinsertion: A New Search Operator for Accelerated and More Accurate Bayesian Network Structure Learning. In: *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*. 2003, pp. 552–559 (cited on page 53).

[191] Murphy, K. P. Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis. University of California, Berkeley, 2002 (cited on page 59).

[192] Nagarajan, R., Scutari, M., and Lebre, S. *Bayesian Networks in R with Applications in Systems Biology*. ISBN 978-1461464457. Springer, 2013 (cited on page 47).

[193] Nandy, P., Hauser, A., and Maathuis, M. H. High-dimensional Consistency In Score-based and Hybrid Structure Learning. In: *The Annals of Statistics* 46(6A):3151–3183, 2018 (cited on page 55).

[194] Narasimhan, M. and Bilmes, J. A. PAC-learning Bounded Tree-width Graphical Models. In: *UAI-04 – Proceedings of the Twenteeth Conference on Uncertainty in Artificial Intelligence*. 2004, pp. 410–417 (cited on page 121).

[195] Neapolitan, R. E. *Learning Bayesian Networks*. Vol. 38. Prentice Hall, 2004 (cited on pages 22, 23, 39, 44, 47, 61).

[196] Nielsen, J. D., Kočka, T., and Peña, J. M. On Local Optima in Learning Bayesian Networks. In: *UAI-03 – Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*. 2003, pp. 435–442 (cited on page 55).

[197] Ordyniak, S. and Szeider, S. Parameterized Complexity Results for Exact Bayesian Network Structure Learning. In: *Journal of Artificial Intelligence Research* 46:263–302, 2013 (cited on page 56).

[198] Ospina-Forero, L., Castañeda, G., and Guerrero, O. A. Estimating Networks of Sustainable Development Goals. In: *Information & Management* to appear:103342, 2021 (cited on pages 75, 153).

[199] Pearl, J. A Constraint-propagation Approach to Probabilistic Reasoning. In: *UAI-85 – Proceedings of the First Conference on Uncertainty in Artificial Intelligence*. 1985, pp. 31–42 (cited on page 17).

[200] Pearl, J. *Causality: Models, Reasoning and Inference*. 1st ed. Cambridge University Press, 2000 (cited on pages 29–31, 111, 152).

[201] Pearl, J. *Causality: Models, Reasoning and Inference*. 2nd ed. Cambridge University Press, 2009 (cited on pages 8, 29, 32, 39).

[202] Pearl, J. *On the Statistical Interpretation of Structural Equations*. Tech. rep. R-200. Cognitive Systems Lab, UCLA, 1993 or 1994 (cited on page 38).

[203] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1st ed. Morgan Kaufmann, 1988 (cited on pages 6, 17, 18).

[204] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 2nd ed. Elsevier, 1997 (cited on page 50).

[205] Pearl, J. and Dechter, R. Identifying Independencies In Causal Graphs with feedback. In: *UAI-96 – Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. 1996, pp. 420–426 (cited on page 38).

[206] Pearl, J. and Mackenzie, D. *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018 (CITED ON PAGES 1, 2, 30).

[207] Pearl, J. and Paz, A. Graphoids: Graph-based Logic for Reasoning about Relevance Relations or When Would x Tell You More About y if You Already Know z? In: *ECAI-86 – Proceedings of the Seventh European Conference on Artificial Intelligence*. Vol. 2. 1986, pp. 357–363 (CITED ON PAGES 8, 17).

[208] Pearl, J. and Verma, T. S. A Theory of Inferred Causation. In: *KR-91 – Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*. 1991, pp. 441–452 (CITED ON PAGE 47).

[209] Pensar, J., Talvitie, T., Hyttinen, A., and Koivisto, M. A Bayesian Approach for Estimating Causal Effects from Observational Data. In: *UAI-20 – Proceedings of the Thirty-sixth Conference on Uncertainty in Artificial Intelligence*. 2020 (CITED ON PAGES 75, 153).

[210] Perković, E., Textor, J., Kalisch, M., and Maathuis, M. H. Complete Graphical Characterization and Construction of Adjustment Sets in Markov Equivalence Classes of Ancestral Graphs. In: *The Journal of Machine Learning Research* 18(220):1–62, 2018 (CITED ON PAGES 21, 32).

[211] Perrier, E., Imoto, S., and Miyano, S. Finding Optimal Bayesian Network Given a Super-Structure. In: *Journal of Machine Learning Research* 9(10), 2008 (CITED ON PAGE 65).

[212] Peters, J. and Bühlmann, P. Structural Intervention Distance for Evaluating Causal Graphs. In: *Neural Computation* 27(3):771–799, 2015 (CITED ON PAGES 67, 123).

[213] Peters, J., Bühlmann, P., and Meinshausen, N. Causal Inference by Using Invariant Prediction: Identification and Confidence Intervals. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*:947–1012, 2016 (CITED ON PAGE 60).

[214] Peters, J., Janzing, D., and Schölkopf, B. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT press, 2017 (CITED ON PAGES 9, 13, 16, 28, 31, 32, 39, 41, 47, 59).

[215] Pfister, N., Bühlmann, P., Schölkopf, B., and Peters, J. Kernel-based tests for joint independence. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 80(1):5–31, 2018 (CITED ON PAGE 13).

[216] Post, E. L. Recursively enumerable sets of positive integers and their decision problems. In: *Bulletin of the American Mathematical Society* 50(5):284–316, 1944 (CITED ON PAGE 45).

[217] Radhakrishnan, A., Solus, L., and Uhler, C. Counting Markov Equivalence Classes by Number of Immoralities. In: *UAI-17 – Proceedings of the Thirty-third Conference on Uncertainty in Artificial Intelligence*. 2017 (CITED ON PAGE 27).

[218] Ramsey, J., Glymour, M., Sanchez-Romero, R., and Glymour, C. 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. In: *International Journal of Data Science and Analytics* 3(2):121–129, 2017 (CITED ON PAGE 55).

[219] Ramsey, J., Zhang, J., and Spirtes, P. L. Adjacency-faithfulness and Conservative Causal Inference. In: *UAI-06 – Proceedings of the Twenty-second Conference Conference on Uncertainty in Artificial Intelligence*. 2006 (CITED ON PAGES 113, 127, 133).

[220] Raskutti, G. and Uhler, C. Learning Directed Acyclic Graph Models Based on Sparsest Permutations. In: *Stat* 7(1):e183, 2018. e183 sta4.183 (cited on page 56).

[221] Reichenbach, H. *The Direction of Time*. Vol. 65. University of California Press, 1956 (cited on page 6).

[222] Richardson, T. and Spirtes, P. L. Ancestral graph Markov models. In: *The Annals of Statistics* 30(4):962–1030, 2002 (cited on pages 27, 37, 50).

[223] Richardson, T. S. A Discovery Algorithm for Directed Cyclic Graphs. In: *UAI-96 – Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. 1996, pp. 454–461 (cited on pages 36, 51).

[224] Robinson, R. W. Counting Labeled Acyclic Digraphs. In: *New Directions in the Theory of Graphs*:239–273, 1973 (cited on page 45).

[225] Robinson, R. W. Counting Unlabeled Acyclic Digraphs. In: *Combinatorial mathematics VII : proceedings of the Fifth Australian Conference on Combinatorial Mathematics*. 1977, pp. 28–43 (cited on page 45).

[226] Rosenbaum, P. R. *Design of Observational Studies*. Springer, 2020 (cited on page 30).

[227] Rosenström, T., Jokela, M., Puttonen, S., Hintsanen, M., Pulkki-Råback, L., Viikari, J. S. A., Raitakari, O. T., and Keltikangas-Järvinen, L. Pairwise measures of causal direction in the epidemiology of sleep problems and depression. In: *PLOS One* 7(11):e50841, 2012 (cited on page 2).

[228] Rosvall, M. and Bergstrom, C. T. Maps of Random Walks on Complex Networks Reveal Community Structure. In: *Proceedings of the National Academy of Sciences* 105(4):1118–1123, 2008 (cited on page 148).

[229] Rubin, D. B. Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies. In: *Journal of Educational Psychology* 66(5):688, 1974 (cited on page 30).

[230] Rubin, D. B. Inference and Missing Data. In: *Biometrika* 63(3):581–592, 1976 (cited on page 60).

[231] Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. In: *Science* 308(5721):523–529, 2005 (cited on pages 58, 60).

[232] Scanagatta, M., de Campos, C. P., Corani, G., and Zaffalon, M. Learning Bayesian Networks with Thousands of Variables. In: *NeurIPS-15 – Proceedings of the Twenty-ninth International Conference on Neural Information Processing Systems*. 2015, pp. 1864–1872 (cited on pages 54, 56).

[233] Schmidberger, M., Lennert, S., and Mansmann, U. Conceptual Aspects of Large Meta-analyses with Publicly Available Microarray Data: A Case Study in Oncology. In: *Bioinformatics and Biology Insights* 5:BBI–S5537, 2011 (cited on pages 75, 122).

[234] Schölkopf, B. and Smola, A. J. *Learning with Kernels*. MIT Press, 2002 (cited on page 13).

[235] Schwarz, G. Estimating the Dimension of a Model. In: *The Annals of Statistics* 6(2):461–464, 1978 (cited on page 64).

[236] Scutari, M. An Empirical-Bayes Score for Discrete Bayesian Networks. In: *Conference on Probabilistic Graphical Models*. 2016, pp. 438–448 (cited on page 64).

[237] Scutari, M. Bayesian Network Constraint-based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. In: *Journal of Statistical Software* 77(2), 2017 (CITED ON PAGE 75).

[238] Scutari, M. From Incomplete, Dynamic Data to Bayesian Networks. arXiv:1906.06513. 2019 (CITED ON PAGE 61).

[239] Scutari, M. Learning Bayesian Networks with the bnlearn R Package. In: *Journal of Statistical Software* 35(3):1–22, 2010 (CITED ON PAGE 47).

[240] Scutari, M. and Denis, J.-B. *Bayesian Networks with Examples in R*. ISBN 978-1482225587. CRC Press, 2014 (CITED ON PAGE 47).

[241] Scutari, M., Graafland, C. E., and Gutiérrez, J. M. Who Learns Better Bayesian Network Structures: Constraint-Based, Score-based or Hybrid Algorithms? In: *Proceedings of the Ninth International Conference on Probabilistic Graphical Models*. Vol. 72. 2018, pp. 416–427 (CITED ON PAGES 47, 64).

[242] Scutari, M., Vitolo, C., and Tucker, A. Learning Bayesian Networks from Big Data with Greedy Search: Computational Complexity and Efficient Implementation. In: *Statistics and Computing* 29(5):1095–1108, 2019 (CITED ON PAGE 34).

[243] Shachter, R. D. Bayes-Ball – The Rational Pastime (for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams). In: *UAI-98 – Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. 1998, pp. 480–487 (CITED ON PAGE 17).

[244] Shimizu, S. Non-Gaussian methods for causal structure learning. In: *Prevention Science* 20(3):431–441, 2019 (CITED ON PAGE 2).

[245] Shimizu, S., Hoyer, P. O., Hyvärinen, A., and Kerminen, A. A Linear Non-Gaussian Acyclic Model for Causal Discovery. In: *Journal of Machine Learning Research* 7(72):2003–2030, 2006 (CITED ON PAGE 39).

[246] Shojaie, A., Jauhiainen, A., Kallitsis, M., and Michailidis, G. Inferring Regulatory Networks by Combining Perturbation Screens and Steady State Gene Expression Profiles. In: *PLOS One* 9(2):e82393, 2014 (CITED ON PAGES 56, 59, 60).

[247] Shojaie, A. and Michailidis, G. Penalized Likelihood Methods for Estimation of Sparse High-Dimensional Directed Acyclic Graphs. In: *Biometrika* 97(3):519–538, 2010 (CITED ON PAGES 59, 122).

[248] Silander, T. and Myllymäki, P. A Simple Approach for Finding the Globally Optimal Bayesian Network Structure. In: *UAI-06 – Proceedings of the Twenty-second Conference Conference on Uncertainty in Artificial Intelligence*. 2006 (CITED ON PAGE 54).

[249] Singh, A. P. and Moore, A. W. *Finding Optimal Bayesian Networks by Dynamic Programming*. Tech. rep. CMU-CALD-05-106. School of Computer Science, Carnegie Mellon University, 2005 (CITED ON PAGE 54).

[250] Singh, M. and Valtorta, M. An Algorithm for the Construction of Bayesian Network Structures from Data. In: *UAI-93 – Proceedings of the Ninth International Conference on Uncertainty in Artificial Intelligence*. 1993, pp. 259–265 (CITED ON PAGE 57).

[251] Sloane, N. J. A. *The On-Line Encyclopedia of Integer Sequences*. A003024: Number of Acyclic Digraphs (or DAGs) with n Labeled Nodes. URL: https://oeis.org/A003024 (CITED ON PAGE 45).

[252] Solus, L. Distributional Invariances and Interventional Markov Equivalence for Mixed Graph Models. arXiv:1911.10114. 2019 (CITED ON PAGE 35).

[253] Solus, L., Wang, Y., and Uhler, C. Consistency Guarantees for Greedy Permutation-Based Causal Inference Algorithms. In: *Biometrika*, 2021 (cited on page 56).

[254] Sondhi, A. and Shojaie, A. The Reduced PC-Algorithm: Improved Causal Structure Learning in Large Random Networks. In: *Journal of Machine Learning Research* 20(164):1–31, 2019 (cited on pages 50, 74, 79).

[255] Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L., and Cowell, R. G. Bayesian Analysis in Expert Systems. In: *Statistical Science*:219–247, 1993 (cited on pages 44, 64).

[256] Spiegelhalter, D. J. and Lauritzen, S. L. Sequential Updating of Conditional Probabilities on Directed Graphical Structures. In: *Networks* 20(5):579–605, 1990 (cited on page 44).

[257] Spirtes, P. Introduction to Causal Inference. In: *Journal of Machine Learning Research* 11(5), 2010 (cited on page 127).

[258] Spirtes, P. L. *A Necessary and Sufficient Condition for Conditional Independencies to Imply a Vanishing Tetrad Difference*. Tech. rep. CMU-LCL-89-3. Laboratory for Computational Linguistics, Carnegie Mellon University, 1989 (cited on page 48).

[259] Spirtes, P. L. Directed Cyclic Graphical Representations of Feedback Models. In: *UAI-95 – Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. 1995, pp. 491–498 (cited on pages 19, 38, 39).

[260] Spirtes, P. L. Directed Cyclic Graphs, Conditional Independence, and Nonrecursive Linear Structural Equation Models. 1993 (cited on page 38).

[261] Spirtes, P. L., Glymour, C. N., Scheines, R. P., and Glymour, C. Causality from Probability. In: *Proceedings of Advanced Computing for the Social Sciences*. Citeseer. 1990 (cited on pages 1, 16, 47–49).

[262] Spirtes, P. L., Glymour, C. N., Scheines, R. P., Heckerman, D., Meek, C., Cooper, G., and Richardson, T. *Causation, Prediction, and Search*. 1st ed. MIT Press, 1993 (cited on pages 31, 76, 114).

[263] Spirtes, P. L., Glymour, C. N., Scheines, R. P., Heckerman, D., Meek, C., Cooper, G., and Richardson, T. *Causation, Prediction, and Search*. 2nd ed. MIT Press, 2000 (cited on pages 3, 21, 30, 36, 37, 47–50, 72, 74, 112, 157).

[264] Spirtes, P. L., Meek, C., and Richardson, T. S. Causal Inference in the Presence of Latent Variables and Selection Bias. In: *UAI-95 – Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. 1995, pp. 499–506 (cited on page 50).

[265] Spirtes, P. L. and Zhang, K. Causal Discovery and Inference: Concepts and Recent Methodological Advances. In: *Applied informatics* 3(1):1–28, 2016 (cited on page 47).

[266] Spirtes, P. L. and Zhang, K. Search for Causal Models. In: *Handbook of Graphical Models*. 1st ed. 2018. Chap. Search for Causal Models (cited on pages 10, 50, 51, 75, 127).

[267] Steck, H. Constraint-based Structural Learning in Bayesian Networks using Finite Data Sets. PhD thesis. Technische Universität München, 2001 (cited on pages 40, 44, 57, 62, 63).

[268] Stekhoven, D. J., Moraes, I., Sveinbjörnsson, G., Hennig, L., Maathuis, M. H., and Bühlmann, P. Causal Stability Ranking. In: *Bioinformatics* 28(21):2819–2823, 2012 (cited on page 122).

[269] Strobl, E. V. A Constraint-based Algorithm for Causal Discovery with Cycles, Latent Variables and Selection Bias. In: *International Journal of Data Science and Analytics* 8(1):33–56, 2019 (CITED ON PAGES 40, 51).

[270] Strobl, E. V. Automated Hyperparameter Selection for the PC Algorithm. arXiv:2011.01889. 2020 (CITED ON PAGES 88, 123).

[271] Strobl, E. V., Spirtes, P. L., and Visweswaran, S. Estimating and Controlling the False Discovery Rate for the PC Algorithm Using Edge-specific p-Values. In: *ACM Transactions on Intelligent Systems and Technology* 10(5):46:1–46:37, 2019 (CITED ON PAGE 127).

[272] Strobl, E. V., Visweswaran, S., and Spirtes, P. L. Fast Causal Inference with Non-random Missingness by Test-wise Deletion. In: *International Journal of Data Science and Analytics* 6(1):47–62, 2018 (CITED ON PAGE 61).

[273] Studený, M. Conditional Independence and Basic Markov Properties. In: *Handbook of Graphical Models*. 1st ed. 2018 (CITED ON PAGES 17, 22).

[274] Süel, G. M., Lockless, S. W., Wall, M. A., and Ranganathan, R. Evolutionarily Conserved Networks of Residues Mediate Allosteric Communication in Proteins. In: *Nature Structural & Molecular Biology* 10(1):59–69, 2003 (CITED ON PAGE 151).

[275] Suzuki, J. An Efficient Bayesian Network Structure Learning Strategy. In: *New Generation Computing* 35(1):105–124, 2017 (CITED ON PAGE 54).

[276] Teyssier, M. and Koller, D. Ordering-Based Search: A Simple and Effective Algorithm for Learning Bayesian Networks. In: *UAI-05 – Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*. 2005, pp. 548–549 (CITED ON PAGES 56, 59).

[277] Tillman, R. E., Danks, D., and Glymour, C. Integrating Locally Learned Causal Structures with Overlapping Variables. In: *NeurIPS-08 – Proceedings of the Twenty-second International Conference on Neural Information Processing Systems*. 2008, pp. 1665–1672 (CITED ON PAGE 60).

[278] Tillman, R. E. and Spirtes, P. Learning Equivalence Classes of Acyclic Models with Latent and Selection Variables from Multiple Datasets with Overlapping Variables. In: *AISTATS-2011 – Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Vol. 15. 2011, pp. 3–15 (CITED ON PAGE 60).

[279] Ting, K. M. Confusion Matrix. In: *Encyclopedia of Machine Learning*. 2017 (CITED ON PAGE 65).

[280] Triantafillou, S. and Tsamardinos, I. Constraint-based Causal Discovery from Multiple Interventions over Overlapping Variable Sets. In: *Journal of Machine Learning Research* 16:2147–2205, 2015 (CITED ON PAGES 57, 58, 60).

[281] Triantafillou, S., Tsamardinos, I., and Tollis, I. Learning Causal Structure from Overlapping Variable Sets. In: *AISTATS-2010 – Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 860–867 (CITED ON PAGES 57, 58, 60).

[282] Tsagris, M. A New Scalable Bayesian Network Learning Algorithm with Applications to Economics. In: *Computational Economics* 57(1):341–367, 2021 (CITED ON PAGES 60, 64, 75, 76).

[283] Tsamardinos, I., Aliferis, C. F., and Statnikov, A. Time and Sample Efficient Discovery of Markov Blankets and Direct Causal Relations. In: *Proceedings of the*

Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM. 2003, pp. 673–678 (cited on pages 50, 57).

[284] Tsamardinos, I., Aliferis, C. F., and Statnikov, A. R. Algorithms for Large Scale Markov Blanket Discovery. In: *FLAIRS-2003 – Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*. Vol. 2. 2003, pp. 376–380 (cited on page 50).

[285] Tsamardinos, I. and Borboudakis, G. Permutation Testing Improves Bayesian Network Learning. In: *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III*. Vol. 6323. 2010, pp. 322–337 (cited on page 121).

[286] Tsamardinos, I. and Brown, L. E. Bounding the False Discovery Rate in Local Bayesian Network Learning. In: *AAAI-08 – Proceedings of the Twenty-third Conference on Artificial Intelligence*. 2008, pp. 1100–1105 (cited on page 127).

[287] Tsamardinos, I., Brown, L. E., and Aliferis, C. F. The max-min hill-climbing Bayesian network structure learning algorithm. In: *Machine Learning* 65(1):31–78, 2006 (cited on pages 50, 57, 64, 69, 115).

[288] Tsamardinos, I., Triantafillou, S., and Lagani, V. Towards Integrative Causal Analysis of Heterogeneous Data Sets and Studies. In: *The Journal of Machine Learning Research* 13:1097–1157, 2012 (cited on page 60).

[289] Tu, R., Zhang, C., Ackermann, P., Mohan, K., Kjellström, H., and Zhang, K. Causal Discovery in the Presence of Missing Data. In: *AISTATS-2019 – Proceedings of the Twenty-second International Conference on Artificial Intelligence and Statistics*. Vol. 89. 2019, pp. 1762–1770 (cited on page 61).

[290] Turing, A. M. Systems of Logic Based on Ordinals. In: *Proceedings of the London Mathematical Society* 2(1):161–228, 1939 (cited on page 45).

[291] Uhler, C., Raskutti, G., Bühlmann, P., and Yu, B. Geometry of the faithfulness assumption in causal inference. In: *The Annals of Statistics*:436–463, 2013 (cited on page 115).

[292] Valiant, L. G. A Theory of the Learnable. In: *Communications of the ACM* 27(11):1134–1142, 1984 (cited on page 121).

[293] van der Zander, B. Algorithms of Identifying Causal Effects in Graphical Models. PhD thesis. Universität zu Lübeck, 2020 (cited on page 27).

[294] van der Zander, B. and Liśkiewicz, M. Separators and Adjustment Sets in Markov Equivalent DAGs. In: *AAAI-16 – Proceedings of the Thirtieth Conference on Artificial Intelligence*. Vol. 30. 1. 2016 (cited on page 21).

[295] Verma, T. and Pearl, J. An Algorithm for Deciding if a Set of Observed Independencies Has a Causal Explanation. In: *UAI-92 – Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*. Elsevier. 1992, pp. 323–330 (cited on pages 23, 49).

[296] Verma, T. and Pearl, J. Causal Networks: Semantics and Expressiveness. In: *UAI-88 – Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*. 1990, pp. 69–78 (cited on pages 17, 29).

[297] Verma, T. and Pearl, J. Equivalence and Synthesis of Causal Models. In: *UAI-90 – Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*. 1990, pp. 255–270 (cited on pages 16, 21, 22, 47–49).

[298] Verma, T. S. *Graphical Aspects of Causal Models*. Tech. rep. R-191. UCLA, 1993 (CITED ON PAGE 50).

[299] Vitale, V., Musella, F., Vicard, P., and Guizzi, V. Modelling an Energy Market with Bayesian Networks for Non-normal Data. In: *Computational Management Science* 17(1):47–64, 2020 (CITED ON PAGE 75).

[300] Wang, J. Partial Correlation Coefficient. In: *Encyclopedia of Systems Biology*. 2013, pp. 1634–1635 (CITED ON PAGE 11).

[301] Wasserman, L. *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media, 2013 (CITED ON PAGE 127).

[302] Watts, D. J. and Strogatz, S. H. Collective Dynamics of "Small-world" Networks. In: *Nature* 393(6684):440–442, 1998 (CITED ON PAGE 67).

[303] Wermuth, N. Linear Recursive Equations, Covariance Selection, and Path Analysis. In: *Journal of the American Statistical Association* 75:963–, 1980 (CITED ON PAGES 48, 59).

[304] Wermuth, N. and Cox, D. R. Graphical Markov Models: Overview. In: *International Encyclopedia of the Social and Behavioral Sciences*. 2015 (CITED ON PAGE 27).

[305] Wermuth, N. and Lauritzen, S. L. Graphical and Recursive Models for Contingency Tables. In: *Biometrika*:537–552, 1983 (CITED ON PAGES 48, 59).

[306] West, M. Bayesian Factor Regression Models in the "large p, small n" Paradigm. In: *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*. 2003, pp. 733–742 (CITED ON PAGE 34).

[307] Wiener, N. The Theory of Prediction. In: *Modern Mathematics for Engineers*, 1956 (CITED ON PAGE 59).

[308] Wienöbst, M., Bannach, M., and Liśkiewicz, M. Extendability of Causal Graphical Models: Algorithms and Computational Complexity. 2021 (CITED ON PAGE 26).

[309] Wienöbst, M., Bannach, M., and Liśkiewicz, M. Polynomial-Time Algorithms for Counting and Sampling Markov Equivalent DAGs. arXiv:2012.09679. 2021 (CITED ON PAGES 17, 26, 117).

[310] Wienöbst, M. and Liśkiewicz, M. Recovering Causal Structures from Low-Order Conditional Independencies. In: *AAAI-20 – Proceedings of the Thirty-fourth Conference on Artificial Intelligence*. 2020, pp. 10302–10309 (CITED ON PAGES 50, 74, 79, 114, 115).

[311] Wollschläger, D. *Grundlagen der Datenanalyse mit R*. Springer, 2010 (CITED ON PAGE 11).

[312] Wright, S. Correlation and Causation. In: *Journal of Agricultural Research* 20(7):557–585, 1921 (CITED ON PAGES 1, 11).

[313] Wright, S. The relative importance of heredity and environment in determining the piebald pattern of guinea-pigs. In: *Proceedings of the National Academy of Sciences* 6(6):320, 1920 (CITED ON PAGE 1).

[314] Yang, K., Katcoff, A., and Uhler, C. Characterizing and Learning Equivalence Classes of Causal DAGs under Interventions. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5541–5550 (CITED ON PAGE 35).

[315] Zarebavani, B., Jafarinejad, F., Hashemi, M., and Salehkaleybar, S. cuPC: CUDA-Based Parallel PC Algorithm for Causal Structure Learning on GPU. In: *IEEE Transactions on Parallel and Distributed Systems* 31(3):530–542, 2020 (CITED ON PAGE 75).

[316] Zhang, J. Causal reasoning with ancestral graphs. In: *Journal of Machine Learning Research* 9:1437–1474, 2008 (cited on pages 27, 37, 50).

[317] Zhang, J. On the Completeness of Orientation Rules for Causal Discovery in the Presence of Latent Confounders and Selection Bias. In: *Artificial Intelligence* 172(16-17):1873–1896, 2008 (cited on page 50).

[318] Zhang, K., Peters, J., Janzing, D., and Schölkopf, B. Kernel-based Conditional Independence Test and Application in Causal Discovery. In: *UAI-11 – Proceedings of the Twenty-seventh Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2011, pp. 804–813 (cited on page 13).

[319] Zuk, O., Margel, S., and Domany, E. On the Number of Samples Needed to Learn the Correct Structure of a Bayesian Network. In: *UAI-06 – Proceedings of the Twenty-second Conference Conference on Uncertainty in Artificial Intelligence*. 2006 (cited on page 121).

A

Appendix

The following Figures complement the analysis in Section 5.3.3.

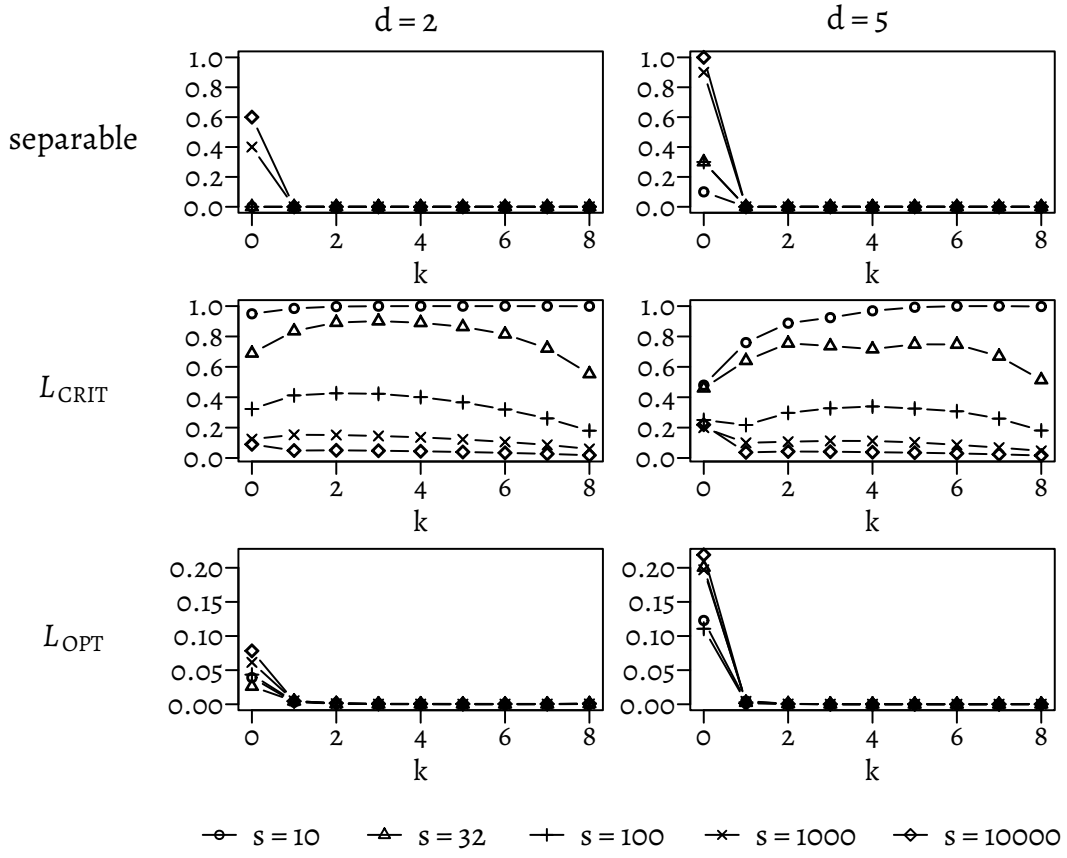


Figure A.1: Separability and sizes of the critical and optimal ranges for increasing k, different sample sizes s (see the legend) and expected densities d of the true graph (columns).In all panels, the number of nodes is $n = 12$. For the definition of the measures see the text.

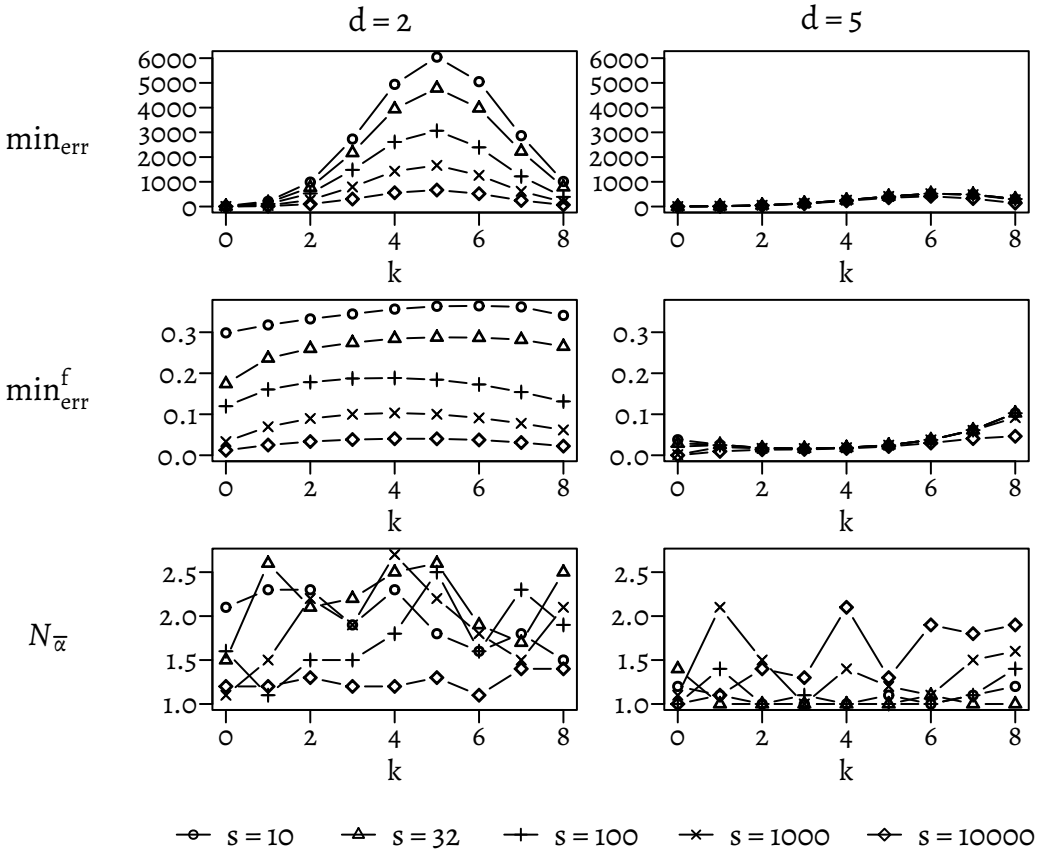


Figure A.2: Behaviour of the minimal number $\min_{\text{err}}$ and fraction $\min^{\text{f}}_{\text{err}}$ of errors, as well as the number of minima $N_{\bar{\alpha}}$ for increasing k, different sample sizes s (see the legend) and expected densities d of the true graph (columns).In all panels, the number of nodes is $n = 12$. For the definition of the measures see the text.

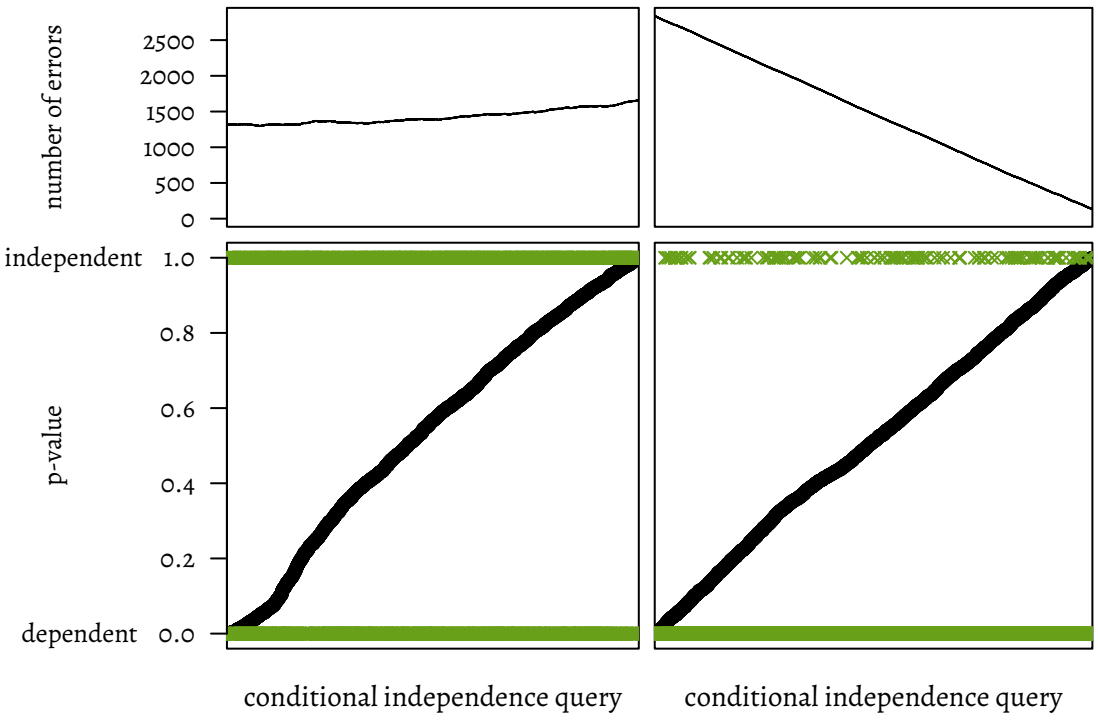


Figure A.3: Example of p-values obtained from two graphs with $n = 12$ nodes and an expected density of $d = 2$ (left) and $d = 5$ (right), with $s = 10$ samples and for conditional independence queries of order $k = 7$ (lower panels). Above, the error function is shown, which describes for each conditional independence query the number of errors that would result from the classification of the complete set of p-values if α is chosen from the interval between the p-value of this conditional independence query and the one next in order.

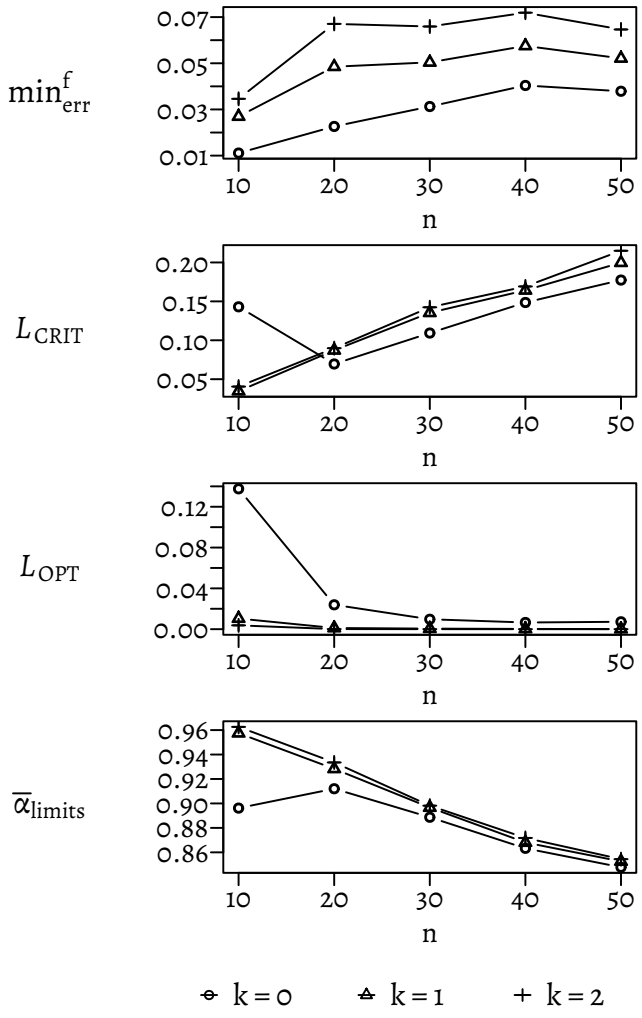


Figure A.4: Behaviour of different measures with increasing n for $s = 10000$ samples and expected densities $d = 2$ of the true graph. For the definition of the measures see the text.