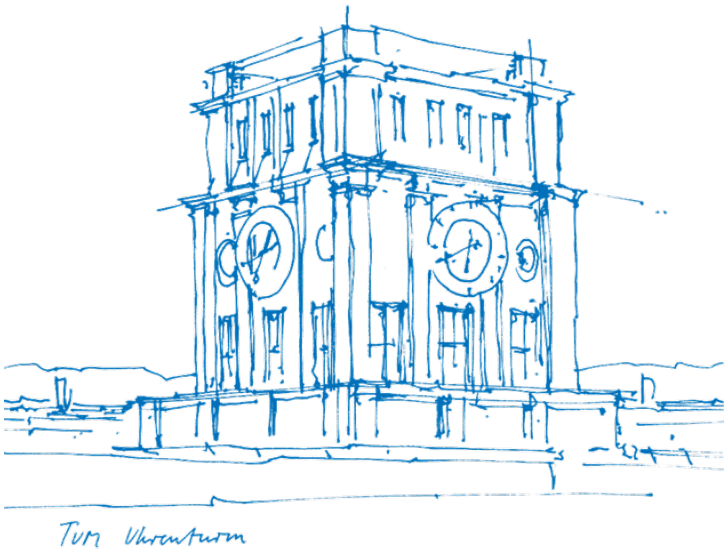
Chair of Mathematical Statistics
Department of Mathematics
Technical University of Munich

Gaussian Graphical Models with Feedback Loops

Sebastian Kaiser



Gaussian Graphical Models with Feedback Loops

Sebastian Kaiser

Zusammenfassung

In dieser Arbeit betrachten wir gerichtete Graphen, die Zyklen enthalten können. Diese Klasse von Graphen ist im Vergleich zu azyklischen Graphen deutlich weniger erforscht. In diesem Zusammenhang wiederholen wir zu Beginn die Grundlagen Graphischer Modelle, wobei wir unter anderem Strukturgleichungsmodelle, das Prinzip von bedingter Unabhängigkeit und separation in einem Graph, sowie die bekannten Markov Eigenschaften wiederholen. Weiterhin nennen wir die Annahmen die in dieser Arbeit getroffen werden. Diese sind unter anderem die Gausssche Verteilung und die Abwesenheit von verborgenen Variablen.

Aufgrund des immensen Wachstums der Anzahl der gerichteten Graphen, eine bekannte Vorgehensweise ist das Zusammenführen von Graphen in Klassen, die eine gewisse Äquivalenz aufweisen. Hier wiederholen wir die am meisten verwendete Definition von Markov Äquivalenz. Im Falle von azyklischen Graphen existiert eine intuitive Charakterisierung von Markov Äquivalenz. Für Graphen die auch Zyklen enthalten können ist so eine lokale Charakterisierung, wie es für azyklische Graphen der Fall ist, nicht verfügbar. Dennoch gibt es auch hierfür eine Charakterisierung von Markov Äquivalenz die wir vorstellen werden. Die Markov Äquivalenzklasse eines zyklischen Graphes besitzt auch einen Vertreter der diese Klasse repräsentiert. Wir wiederholen Vorfahrensgraphen und im speziellen auch Teilweise-Vorfahrensgraphen, die eine Markov Äquivalenzklasse in präsenz von Zyklen repräsentieren können. Als nächstes wiederholen wir auch einen Algorithmus, der diesen Graphen finden kann, vorausgesetzt ein separations Orakel ist gegeben.

Seit längerer Zeit ist bekannt, dass in Anwesenheit von Zyklen die bekannte Charakterisierung von Äquialenz nicht alle Informationen im Graph wiederspiegelt. Dafür präsentieren wir eine kürzlich gefundene neue Charakterisierung von Äquivalenz, die auf Verteilungen basiert. Kurz gesagt sind zwei Strukturen Verteilungsäquivalent genau dann wenn sie die gleiche Menge von Verteilungen generieren können. Um diese Klasse zu bestimmen wurden Abbildungen im Form von Rotationen gefunden. Wir geben bestimmte Eigenschaften von sogenannten einfachen Graphen, die zwischen zwei Knoten nur eine Kante enthalten dürfen. Außerdem implementieren wir den Algorithmus zur Bestimmung der Äquivalenzklasse und visualisieren die Zusammenhänge in dieser Klasse.

Als nächstes berechnen wir diese Verteilungsäquivalenzklasse für jeden Graph mit bis zu fünf Knoten und unterteilen diese Klassen in ihre Markov Äquivalenzklassen. Es zeigt sich, dass der teilweise Vorfahrensgraph in bestimmten Teilen die Verteilungsäquivalenzklasse eindeutig identifizieren kann. Außerdem wenden wir unsere Erkenntnisse über einfache Graphen auf diese Klassen an.

Im letzten Kapitel implementieren wir einen Strukturlernalgorithmus, der mithilfe von einem Datensatz, den zugrundeliegenden Graphen bestimmen soll. Hier verwenden wir bereits bekannte Algorithmen und präsentieren eine Kombination aus diesen.

Contents

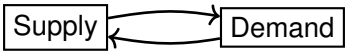
1 Introduction

A graphical model is a statistical model associated to a graph, where the nodes of the graph correspond to random variables and the edges of the graph encode conditional independence statements between the variables [Lauritzen, 1996]. Moreover directed graphical models allow an intuitive causal interpretation and have become essential in causal inference.
Although the roots of graphical modeling can be traced back to path analysis [Wright, 1921], its modern form has been developed in the 1980's and 90's by key publications including [Darroch et al., 1980] and [Lauritzen and Wermuth, 1989]. Furthermore the book of Judea Pearl on probabilistic reasoning [Pearl, 1988] had a substantial impact on promoting graphical models for artificial intelligence. The local computation method proposed by [Lauritzen and Spiegelhalter, 1988] enabled efficient computation of conditional independencies, leading to a theoretical breakthrough. Since then numerous books on graphical modeling have been published, including [Pearl, 2009], [Lauritzen, 1996], [Edwards, 2000],
[Spirtes et al., 1993] and [Maathuis et al., 2018], to name a few of them.

Directed acyclic graphs, also called DAGs, are widely used in modeling causal relationships due to their appealing properties. A DAG admits a factorization property, that is, the joint density is a product of terms that can be efficiently stored and computed. Several Markov properties have been derived that allow one to read conditional independencies directly off the graph through a concept called d-separation. It has been shown that for DAGs these Markov properties are equivalent and under the assumed existence of a density they are equivalent to the factorization property [Pearl, 1988]. Moreover a simple and intuitive characterization of equivalent structures has been derived, called Markov equivalence.

In structure learning one estimates or learns a graph that best describes the dependence structure given a data set. Several algorithms have been developed for this task and applications range from genomics, epidemiology, neuroscience to social sciences and economics [Glymour et al., 2019].
Compare also [Drton and Maathuis, 2017] for an overview of popular approaches in this context.

Nonetheless the well studied class of DAGs face a major limitation, they are not able to represent cycles or feedback loops in the underlying model. Directed cyclic graphs (DCGs) have gained much less attention compared to DAGs. The reasons for this are manifold; they do not always admit the factorization property, the Markov properties might not be equivalent, the characterization of equivalent models is considerably more involved [Spirtes, 1995] [Richardson, 1996b].



Despite the difficulties when working with cyclic models one can not always guarantee that feedback loops are absent. In economics there is a fundamental relationship between the quantity of a product a producer may wish to sell and the quantity that consumers wish to buy. This supply and demand model is already a cyclic model on two variables and determines the price of the product.
So in general there is a need to understand the nature of feedback loops between observed variables. Important work has been done by [Spirtes, 1995], proving conditions under which the factorization and Markov property hold for this type of models. Subsequently, [Richardson, 1996c] presented the cyclic equivalence theorem that characterizes equivalent structures in terms of conditional independence statements and, based on this, developed the first algorithm that discovers a representative of this equivalence class [Richardson, 1996b]. More recently, for more general cyclic models the Markov properties have been

clarified [Forré and Mooij, 2017].
Until now several structure learning algorithms than can deal with feedback loops, and even unobserved variables, exist in the literature [Lacerda et al., 2012] [Strobl, 2019] [Hyttinen et al., 2013]. Statements about the identification of causal effects from data [Foygel et al., 2012] and efficient methods to compute maximum likelihood estimates in cyclic models are available [Drton et al., 2019].
It has been known that when cycles are allowed in the structure, conditional independence may not be a suitable notion of equivalence as it does not reflect all the information that is useful for identifying the underlying structure [Spirtes et al., 1993]. So there might be Markov equivalent structures that can still be distinguished by observational data [Lacerda et al., 2012]. A new notion of equivalence between two structures, based on distributions they can generate, called distribution equivalence, is proposed in [Ghassami et al., 2020]. In this work by Amir Emad Ghassami, graphical operations that transform one directed graph into another distribution equivalent graph, based on rotations on the support matrix, are presented. Given this operations one can compute the distribution equivalence class, that is, the set of structures that can generate the same set of distributions. Furthermore a score-based algorithm for learning a structure from observational data is given.

This thesis is then mainly based on the findings by [Ghassami et al., 2020]. We will use the graphical operations and compute the distribution equivalence class for every directed graph on up to 5 nodes. Here we will connect to Markov equivalence, since two distribution equivalent structures are always Markov equivalent, where the converse does not hold in general when cycles are allowed [Lacerda et al., 2012]. Furthermore statements for a special type of graphs, that only allow one edge between a pair of variables (simple graphs), are derived. In terms of structure learning a hybrid method is proposed that combines the score-based search by [Ghassami et al., 2020] with the discovery algorithm by [Richardson, 1996b].

The work is structures as follows: In *Section 2* we will review some basics about graphical models, including the connection between conditional independence and d-separation through the Markov properties, the converse of the Markov property that allows for identification, called faithfulness, and more broadly some basic concepts of parameter identification. The assumptions we make are that of causal sufficiency and we also restrict ourselves to Gaussian graphical models, which are also shortly reviewed. *Section 3* is concerned with Markov equivalence. Here we present the characterization of Markov equivalence between cyclic structures, and the algorithm for finding this Markov equivalence class, both developed by Thomas Richardson. In *Section 4* we then review the work of Amir Emad Ghassami regarding distribution equivalence. This graphical operations presented in his work allow us to visualize the distribution equivalence class in an undirected graph where every node is a directed graph itself. For simple graphs we give some statements under which conditions the operations on a simple graph lead to another simple graph, or do not so. It is furthermore shown that the definition of distribution equivalence does in some cases only hold up to topological closure of the distribution set. With the algorithm for computing the distribution equivalence class we compute it for every directed graph on up to 5 nodes, call the atlas of classes, and organize them by their corresponding Markov equivalence class. This is presented in *Section 5*. Here we also observe that the representation of a Markov equivalence class in the cyclic setting can in some cases uniquely represent a distribution equivalence class, but also further research is necessary. In *Section 6* we use the score-based algorithm proposed by Amir Emad Ghassami and combine it with the cyclic causal discovery algorithm by Thomas Richardson, resulting in a hybrid structure learning algorithm. In a simulation over cyclic structures up to 15 nodes we see that, given the correct search space, the hybrid method can make better estimates about the data generating structure in less computation time. Finally, *Section 7* concludes this thesis.

2 Preliminaries

We consider graphs $\mathcal{G} = (V, E)$, which consists of a finite set of vertices $V = \{V_1, \ldots, V_p\}$ and a set of edges $E \subseteq V \times V$ between pairs of vertices. We assume there are no self loops, so $E \cap \{(i, i) : i \in V\} = \emptyset$.
Two vertices i, j is a DG are said to be *adjacent* if there is an edge connecting them. If there is an edge $i \to j$ then node i is a direct cause of j, and i is a *parent* of j, and j is a *child* of i. We call j the tail and i the arrowhead of the edge.
A *path* π between two vertices i and j is a sequence of consecutive edges connecting i and j. A *directed path* from i to j is a path consisting of directed edges such that every edge points to j. Then j is a *descendant* of i, and i an *ancestor* of j. Assume that every vertex is a descendant and ancestor of itself. We denote the sets of parents, children, descendants and ancestors of a vertex i in $\mathcal{G}$ as $\mathbf{Pa}_{\mathcal{G}}(i), \mathbf{Ch}_{\mathcal{G}}(i), \mathbf{De}_{\mathcal{G}}(i), \mathbf{An}_{\mathcal{G}}(i)$, respectively. Then for sets $A \subset V$; $\mathbf{An}_{\mathcal{G}}(A) = \bigcup_{i \in A} \mathbf{An}_{\mathcal{G}}(i)$.
A *cycle* occurs when there is a path between vertices i and j, and i and j are adjacent. Moreover we have a *directed cycle* if there is a directed path from i to j and $j \to i$.
A graph $\mathcal{G}$ that only contains directed edges we call a *directed graph* (DG). If $\mathcal{G}$ does not contain directed cycles then it is called *directed acyclic graph* (DAG). Otherwise $\mathcal{G}$ is a *directed cyclic graph* (DCG).
A triple of nodes (i, j, k) is an *unshielded triple*, if i and j, j and k are adjacent, and i and k are not adjacent.

Graphical models can also be defined in terms of their density factorization. Let $\mathbb{P}$ be the joint distribution for $\mathbf{X} = (X_i : i \in V)$ with a density f with respect to a product measure μ.
We say the joint distribution $\mathbb{P}$ factorizes according to the DAG $\mathcal{G}$ if

$$f(\mathbf{X}) = \prod_{i=1}^{p} f(X_i | \mathbf{Pa}_{\mathcal{G}}(X_i)). \tag{2.1}$$

2.1 Linear structural equation models

A linear structural equation model (SEM) captures dependencies between a set of variables $\{X_i : i \in V\}$ by building up a system of linear equations for each variable [Bollen, 1989]. Every equation defines how variable X_i arises from the other variables and an error term ε_i. For a linear SEM we therefore have

$$X_i = \sum_{j \in V \setminus \{i\}} \beta_{ji} X_j + \varepsilon_i, \qquad i \in V. \tag{2.2}$$

This can be written in matrix form as

$$X = BX + \varepsilon, \tag{2.3}$$

where $B = (\beta_{ij})$ is the matrix of coefficients. If variable X_i has no direct effect on X_j, then we have $\beta_{ij} = 0$ (Note that in the literature this is sometimes denoted as $\beta_{ji} = 0$).

The elements of ε are assumed to be Gaussian. Since we can center the data, without loss of generality, assume that ε has zero mean, so

$$\varepsilon \sim \mathcal{N}(0, \Omega), \tag{2.4}$$

where Ω is a symmetric positive definite matrix of parameters. An entry ω_{ji} may capture an unmeasured common cause of X_i and X_j. If there are no latent common causes, constrain $\omega_{ij} = \omega_{ji} = 0$.
We assume that the error terms are jointly independent, so Ω is a diagonal matrix with $\omega_{ii} = Var(\varepsilon_i)$. Write I for the $V \times V$ identity matrix. If $I - B$ is invertible, then the equation system in 2.3 is uniquely solved by $X = (I - B)^{-\top}\varepsilon$, which has covariance matrix

$$Var[X] = \Sigma = (I - B)^{-1}\Omega(I - B)^{-\top}. \tag{2.5}$$

A SEM is called *recursive* if its directed graph is acyclic, otherwise it is *non-recursive*. Corresponding to a set of non-recursive linear equations is a cyclic graph, which is illustrated by the following example [Whittaker, 1990]:

$$\begin{aligned} X_1 &= \varepsilon_1 \\ X_2 &= \varepsilon_2 \\ X_3 &= \beta_{1,3}X_1 + \beta_{4,3}X_4 + \varepsilon_3 \\ X_4 &= \beta_{2,4}X_2 + \beta_{3,4}X_3 + \varepsilon_4 \end{aligned}$$

$\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4$ are jointly independent and normally distributed .

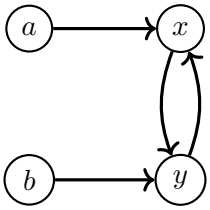


Figure 2.1 The cyclic graph corresponding to the non-recursive SEM.

Gaussian Graphical Models: Since we consider Gaussian graphical models we shortly review some basic properties of this type of model (compare also [Uhler, 2017]). A random vector $\mathbf{X} \in \mathbb{R}^p$ is distributed according to the multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ with mean $\mu \in \mathbb{R}^p$ and covariance matrix $\Sigma \in PD_V$, where PD_V is the cone of positive definite $V \times V$ matrices, if it has density function

$$f_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^p \det \Sigma}} \exp\left\{ -\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right\}, \quad x \in \mathbb{R}^p. \tag{2.6}$$

A random vector $\mathbf{X} \in \mathbb{R}^p$ satisfies the Gaussian graphical model with graph $\mathcal{G}$, if $\mathbf{X}$ has a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, with

$$(\Sigma^{-1})_{i,j} = 0 \quad \text{for all } (i,j) \notin E. \tag{2.7}$$

So $\mathcal{G}$ describes the sparsity pattern of the so called *concentration matrix* $\Theta = \Sigma^{-1}$.
Moreover the following corollary results from the basic properties of the multivariate Gaussian distribution, that is, zeros in the covariance and precision matrix correspond to conditional independence relations.

Corollary 2.1 *[Uhler, 2017, Corollary 2.2]. Let $X \in \mathbb{R}^p$ be distributed as $\mathcal{N}(\mu, \Sigma)$ and let $i, j \in V$ with $i \neq j$. Then*

1. *$X_i \perp\!\!\!\perp X_j$ if and only if $\Sigma_{i,j} = 0$,*
2. *$X_i \perp\!\!\!\perp X_j | X_{V\setminus\{i,j\}}$ if and only if $\Theta_{i,j} = 0$ if and only if $\det(\Sigma_{V\setminus\{i\},V\setminus\{j\}}) = 0$.*

2.2 Conditional Independence and d-separation

For random variables X_i and X_j, we say that X_i and X_j are conditionally independent given $X_S := (X_k : k \in S)$, denoted by $X_i \perp\!\!\!\perp X_j \mid X_S$, if every path between nodes i and j is suitably blocked in $\mathcal{G}$ by the nodes in S. We will sometimes also write $i \perp\!\!\!\perp j \mid S$.

The edge set of a graph encodes conditional independence constraints. This connection is established through the concept of *d-separation* [Pearl, 1988][Lauritzen, 1996].
A non-endpoint node i on a path is called a *collider* if the two adjacent nodes on the path have arrowheads pointed towards i. If a node on a path is not a collider then we say it is a *non-collider*.

Definition 2.1 *For a DG $\mathcal{G} = (V, E)$, let $i, j \in V$ and $S \subseteq V \setminus \{i, j\}$. Then i and j are* d-connected *given S, if there exists a path π from i to j such that,*

- *every collider on π is in $\mathbf{An}_{\mathcal{G}}(S)$, and*
- *every non-collider on π is not is S.*

If there exists no such path then i and j are d-separated *given S. Let $A, B, C \subseteq V$ be pairwise disjoint. If there exists nodes $i \in A$ and $j \in B$ that are d-connected given C, then A and B are d-connected given C. Otherwise A and B are d-separated given C.*

2.2.1 Markov properties

A DG $\mathcal{G} = (V, E)$ and a collection of random variables $(X_i)_{i \in V}$ can satisfy a range of Markov properties. [Lauritzen et al., 1990] stated the following:

Definition 2.2 *A probability measure $\mathbb{P}$ over a set of variables $\mathbf{X}$ satisfies the*

1. local Markov property *relative to a DG $\mathcal{G}$ if for any vertex $i \in V$*

$$i \perp\!\!\!\perp V \setminus (\mathbf{De}_{\mathcal{G}}(i) \cup \mathbf{Pa}_{\mathcal{G}}(i)) \mid \mathbf{Pa}_{\mathcal{G}}(i),$$

2. global Markov property *relative to a DG $\mathcal{G}$ if for any triple (A, B, S) of disjoint subsets of V such that S d-separates A and B,*

$$A \perp\!\!\!\perp B \mid S.$$

For DAGs, the local and global Markov property are equivalent. This is not the case if cycles are allowed. In the DG in *Figure 2.1*, under the local Markov property y is independent of its non-parental non-descendant a, given its parents b and x. Under the global Markov property this is not the case since $a \to x \leftarrow y$ d-connects a and y given b and x. So the both properties are not equivalent.

The global Markov property is also called Markov assumption. It applies to both acyclic and cyclic graphs [Spirtes et al., 1993], but not all directed graphical models satisfy the Markov assumption. However, for a DCG model to satisfy it, the joint distribution should be defined by *generalized factorization*:

Definition 2.3 *[Spirtes, 1995]. The joint distribution of $\mathbf{X}$, $f(\mathbf{X})$, factors according to the directed graph $\mathcal{G}$ with vertices V if and only if for every subset $\mathbf{X}$ of V,*

$$\mathbb{P} = f \cdot \mu \tag{2.8}$$

$$f(\mathbf{An}_{\mathcal{G}}(\mathbf{X})) = \prod_{i \in \mathbf{An}_{\mathcal{G}}(\mathbf{X})} g_i(X_i, \mathbf{Pa}_{\mathcal{G}}(X_i)), \tag{2.9}$$

where g_i is a non-negative function and μ is a product measure.

[Lauritzen et al., 1990] then showed that for a set of random variables $\mathbf{X}$ with a probability measure $\mathbb{P}$, that is absolutely continuous wrt. a product measure μ, $\mathbb{P}$ factors according to a DAG $\mathcal{G}$ if and only if $\mathbb{P}$ satisfies the global Markov property for $\mathcal{G}$. For a cyclic directed graph $\mathcal{G}$ the existence of a factorization according to $\mathcal{G}$ entails that a measure satisfies the global directed Markov property for $\mathcal{G}$ [Spirtes, 1995, Lemma 2]. However, if a probability measure over a set of variables $\mathbf{X}$ satisfies the global Markov property for a cyclic graph $\mathcal{G}$, then it does not follow that $f(\mathbf{X})$ factors according to $\mathcal{G}$. It is then stated by [Spirtes, 1995, Lemma 3] that if a set of random variables $\mathbf{X}$ with probability measure $\mathbb{P}$ that is absolutely continuous wrt. a product measure μ, has positive density $f(\mathbf{X})$ and satisfies the global Markov property for $\mathcal{G}$, then $f(\mathbf{X})$ factors according to $\mathcal{G}$. Furthermore the following assumption on $\mathbb{P}$ is proposed:

Theorem 2.1 *[Spirtes, 1995, Theorem 1]. The probability measure $\mathbb{P}$ of a linear SEM with independent errors satisfies the global Markov property with respect to the SEM's directed (acyclic or cyclic) graph $\mathcal{G}$.*

So we have a concrete example of a cyclic graph that satisfies the factorization in *Definition 2.3*.
Applying the theorem to the graph in *Figure 2.1* only two conditional independence relations are encoded: $a \perp\!\!\!\perp b$ and $a \perp\!\!\!\perp b \mid \{x, y\}$.

This theorem provides a basis for the Cyclic Causal discovery algorithm from [Richardson, 1996b]. In general there may be several directed graphs entailing the same conditional independence statements. These graphs form a *Markov equivalence class*. We will discuss this topic in more detail in *Section 3*.

2.2.2 Faithfulness

Given the distribution $\mathbb{P}$ of $\mathbf{X}$, the problem is to determine the underlying directed graph $\mathcal{G}$. The Markov assumption is fundamental for identifiability of the DG $\mathcal{G}$ based on the distribution $\mathbb{P}$. Still it is not sufficient for inferring the Markov equivalence class for a DG, because there might be many graphs that satisfy the Markov assumption but do not belong to the Markov equivalence class. For example, a complete graph does not satisfy any d-separation relations, hence always satisfies the Markov assumption. Therefore stronger assumptions are needed that encourage the removal of an edge [Park and Raskutti, 2016]. One of the most used assumption is the *Faithfulness assumption*, which states that only those independencies hold true that are implied by the Markov assumption, meaning all conditional independence statements of $\mathbb{P}$ are encode in $\mathcal{G}$.

Definition 2.4 *A directed graph $\mathcal{G}$ is* faithful *to $\mathbb{P}$ if for any disjoint subsets $A, B \subseteq V$ and any subset $S \subset V \setminus \{A, B\}$,*

$$A \text{ d-separated from } B \text{ given } S \iff X_A \perp\!\!\!\perp X_B \mid X_S \text{ according to } \mathbb{P}.$$

One also says that $\mathcal{G}$ is a perfect map *of $\mathbb{P}$.*

While the Faithfulness assumption is sufficient for identifiability and leads to polynomial-time search algorithms [Spirtes et al., 1993], it is also known to be a very strong assumption (compare example of complete DAG on 3 nodes in [Uhler et al., 2013, Section 2]), which is often not satisfied in practice . The assumption is very sensitive to hypothesis testing errors for inferring conditional independence relations from data, where violations often occur in graphs with cycles in the skeleton [Uhler et al., 2013]. Therefore milder assumptions have been considered, all at a computational cost. In the following we give a short review, compare also [Park and Raskutti, 2016] and [Zhang, 2013].
The SGS-minimality condition [Spirtes et al., 1993] states that there exists no proper sub-graph of $\mathcal{G}$ that satisfies the Markov assumption with respect to the distribution $\mathbb{P}$.
The P-minimality condition [Pearl, 2009] states that for directed graphical models statisfying the Markov assumption, models that satisfy more d-separation rules are preferred.
[Zhang, 2013] proved that the SGS-minimality condition is weaker than the P-minimality condition which is weaker than the Faithfulness assumption. Neither P-minimality nor SGS-minimality imply identifiability

of the Markov equivalence class of $\mathcal{G}$. Subsequently [Raskutti and Uhler, 2013] developed the Sparsest Markov representation (SMR) assumption which states that the true DAG model is the DAG satisfying the causal Markov assumption with the fewest edges. The SMR assumption guarantees identifiability of the Markov equivalence class for DAG models.
So the SMR assumption involves counting the number of edges, while the Faithfulness assumption and minimality assumptions involve d-separation relations.
Since these assumptions were originally invented for DAG models, the question is how the connections between the conditions carry over to models that allow for cycles. One difficulty we face in dealing with cyclic models is that of virtual edges between vertices in a graph.

Definition 2.5 *[Richardson, 1996b]. Consider a directed graph $\mathcal{G}$.*

- *For $i, j \in V$, i and j are* really adjacent *in $\mathcal{G}$ if $i \to j$ or $i \leftarrow j$.*
- *For $i, j \in V$, i and j are* virtually adjacent *in $\mathcal{G}$ if i and j have a common child k, such that k is an ancestor of i or j.*

Therefore the maximum d-separation rules (MDR) assumption is proposed by [Park and Raskutti, 2016], which states that the true graph $\mathcal{G}$ entails more d-separation rules than any other graph satisfying the Markov assumption according to the given distribution $\mathbb{P}$. The MDR assumption is strictly weaker than the Faithfulness assumption, so more graphical models satisfy the MDR assumption than the Faithfulness assumption. Moreover the MDR assumption is strictly stronger than the P-minimality assumption. Simulations in [Park and Raskutti, 2016] then show that, in terms of discovering the skeleton of a DG, the MDR assumption outperforms the Faithfulness assumption.
In *Section 6* we will consider the generalized Faithfulness assumption, which is defined in terms of constraints the graph $\mathcal{G}$ imposes on the entries of the precision matrix [Ghassami et al., 2020].

2.2.3 Identification

We assumed the Markov and Faithfulness assumption hold, so all conditional independence relations in the data are also encoded in the underlying graph. Parameter identification is concerned with the question whether the effects of interest, i.e. the coefficients $\beta_{i,j}$ in the linear structural equations, can be recovered from the covariance matrix of the variables (compare [Drton, 2018] for an overview). There are different notions of identifiability, the most strict is if all coefficients $\beta_{i,j}, i \to j \in E$ are identifiable.
Adopting the notation of [Drton et al., 2011], let $\mathbb{R}^E$ be the set of real $V \times V$-matrices $B = (\beta_{i,j})$ with support in E, that is,

$$\mathbb{R}^E = \{B \in \mathbb{R}^{V \times V} : \beta_{i,j} = 0 \text{ if } i \to j \notin E\}. \tag{2.10}$$

Define $\mathbb{R}^E_{reg}$ to be the subset of $\mathbb{R}^E$ for which $I - B$ is invertible. For acyclic graphs the matrix B can be permuted into lower triangular form such that $\det(I - B) = 1$ for all $B \in \mathbb{R}^E$, therefore $\mathbb{R}^E = \mathbb{R}^E_{reg}$.
Define PD_V to be the cone of positive definite symmetric $V \times V$-matrices $\Omega = (\omega_{i,j})$ and define $PD(E)$ to be the subcone of matrices that is supported over E, that is,

$$PD(E) = \{\Omega \in PD_V : \omega_{i,j} = 0 \text{ if } i \neq j \text{ and } i \leftrightarrow j \notin E\}. \tag{2.11}$$

In this work we will assume that there are no bidirected edges in the graphs, i.e. there are no latent common causes (compare *Section 2.3*), so $PD(E)$ will be the set of diagonal matrices with positive entries.
Recall equation (2.5), we can now define the *covariance parametrization* of the model, which is the map

$$\phi_{\mathcal{G}} : \mathbb{R}^E \times PD(E) \to PD_V, \tag{2.12}$$

$$(B, \Omega) \mapsto (I - B)^{-\top} \Omega (I - B)^{-1}. \tag{2.13}$$

In terms of parameter identification the question is now whether the map $\phi_{\mathcal{G}}$ is injective. By [Drton et al., 2011, Theorem 1] if $\phi_{\mathcal{G}}$ is injective, then the graph $\mathcal{G}$ is acyclic. This statement if formulated for mixed graphs, i.e. graphs that contain directed as well as bidirected edges where there can be more than one type of edge between a pair of nodes. Furthermore it is shown that only graphs with at most one edge between a pair of nodes (simple graphs) can have an injective parametrization. Identification in the sense of an injective parametrization might be to strict, so weaker requirements of a model have been considered. If $\phi_{\mathcal{G}}$ is injective for generic choices of $B \in \mathbb{R}^E_{reg}$ and $\Omega \in PD_E$, then one says that the map $\phi_{\mathcal{G}}$ is *generically identifiable*. A condition holds generically if the points for which the condition fails form a lower-dimensional algebraic subset [Drton et al., 2011]. Also cyclic models may also be generically identifiable, consider [Drton, 2009, Example 3.6].
Let

$$\mathcal{M}_{\mathcal{G}} = \{(I - B)^{-\top}\Omega(I - B)^{-1} : B \in \mathbb{R}^E_{reg}, \Omega \in PD(E)\} \tag{2.14}$$

be the set of covariance matrices corresponding to a graph $\mathcal{G}$. An interesting question is to ask for the dimension of $\mathcal{M}_{\mathcal{G}}$, which is crucial for statistical model selection [Amendola et al., 2020]. If for two different graphs $\mathcal{G}$ and $\mathcal{G}'$ we have $\mathcal{M}_{\mathcal{G}} = \mathcal{M}_{\mathcal{G}'}$, meaning the maps $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{G}'}$ have the same image, then we cannot distinguish between $\mathcal{G}$ and $\mathcal{G}'$ when only relying on observational data. $\mathcal{G}$ and $\mathcal{G}'$ then may be called covariance or distributional equivalent. We will cover the topic of distributional equivalence in *Section 4*.

2.3 Causal Sufficiency

Another assumption we make here is that of *Causal sufficiency*, which means that all relevant variables are included in the directed graph $\mathcal{G}$ and there are no hidden (or latent) variables. This makes our models significantly simpler due to some of the following problems of latent variable (LV) models (compare [Richardson and Spirtes, 2002, Introduction]):

- LV models do not form a tractable search space since an arbitrary number of hidden variables results in a class with infinitely many different structures.
- The set of distributions may be difficult to characterize.
- The likelihood may be multi-modal (also in cyclic models).

Moreover DGs with latent variables are not closed under marginalization. Consider the DAG $\mathcal{G}$ in *Figure 2.2*. Assume $\mathcal{G}$ is a perfect map of the distribution of $(X_1, \ldots, X_5)$, i.e. the Markov assumption and Faithfulness hold, and suppose that X_3 is latent. There is no DAG on $\{1, 2, 4, 5\}$ that represents the same d-separation relations as $\mathcal{G}$, therefore no perfect map of the marginalized distribution of (X_1, X_2, X_4, X_5) does exists [Drton and Maathuis, 2017, Section 5.2, Example 16 & 17].

Figure 2.2 A DAG with latent variable X_3 and the corresponding mixed graph that is a perfect map of the distribution of $(X_1, X_2, X_3, X_4, X_5)$.

Mixed graphs have been introduced to address these problems without explicitly modelling the latent variables [Spirtes et al., 1993], so only the observable variables are indexed in the graphs. The edges are directed or bidirected, where a bidirected edge indicates the presence of a latent variables. This allows one to represent more complicated dependencies. For constraint-based structure learning *Maximal ancestral graphs* were introduced by [Richardson and Spirtes, 2002], which extend the class of DG models, but are closed under marginalization. We will cover the topic of ancestral graphs in *Section 3*.

3 Markov Equivalence

In model selection algorithms we seek for a DG model with the highest maximum likelihoods, but there is a problem with this general approach: Different DGs may determine the same statistical model. By putting this collection of possible DGs into a set, we form an *equivalence class*, where elements of this equivalence class determine the same statistical model. Ignoring these classes we would face some difficulties, which can be resolved by treating each class as a single model [Andersson et al., 1997].

The most common way of merging different DGs into one class is given by considering conditional independence constraints.

Definition 3.1 *Two DGs $\mathcal{G}$ and $\mathcal{G}'$ are* Markov equivalent*, denoted by $\mathcal{G} \equiv_M \mathcal{G}'$, if and only if they encode the same set of conditional independence constraints.*

For DAGs there is an intuitive and well-known characterization that specifies Markov equivalent graphs:

Theorem 3.1 *[Verma and Pearl, 1990]. Two DAGs are Markov equivalent if and only if they have the same skeleton and same unshielded colliders.*

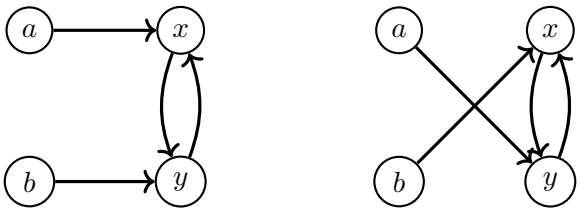


Figure 3.1 Two directed cyclic graphs that entail the same conditional independence statements.

For cyclic DGs this is somewhat more complex, i.e. in *Figure 3.1* the two graphs are Markov equivalent, since we have $a \perp\!\!\!\perp b$ and $a \perp\!\!\!\perp b \mid \{x, y\}$ in both graphs, yet they do not have the same skeleton.

In the following sections we consider Markov equivalent DGs, how we can decide Markov equivalence in the presence of cycles and give a representative of the Markov equivalence class. Furthermore we review an algorithm that discovers this representative, given a d-separation oracle. We base ourselves mainly on the publications by [Richardson, 1996b],[Richardson, 1996c].

3.1 Markov Equivalence Class

Two DGs are Markov equivalent if they share the same conditional independence constraints. We make the following definition

Definition 3.2 *The* Markov equivalence class *(MEC) of a DG $\mathcal{G} = (V, E)$ is the set of DGs*

$$[\mathcal{G}]_M := \{\mathcal{G}' = (V, E') \;\; DG : \mathcal{G}' \equiv_M \mathcal{G}\}. \tag{3.1}$$

We use a subscript M to denote Markov equivalence since we will also consider an equivalence relation based on distributions a structure can generate.

3.2 Deciding Markov Equivalence

Consider we have DAGs $\mathcal{G}_1, \mathcal{G}_2$ and want to check whether $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent. Then one can check if every conditional independence relation in $\mathcal{G}_1$ is also satisfied in $\mathcal{G}_2$, and vice versa. This can be done in exponential time, but by *Theorem 3.1* we only have to check if $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton and same unshielded colliders. Of course $\mathcal{G}_1$ and $\mathcal{G}_2$ must also have the same vertices, otherwise there would be different conditional independence relations in these graphs. An unshielded triple (i, j, k) can either be an unshielded collider or an unshielded non-collider. So two Markov equivalent DAGs must also have the same unshielded non-colliders. The crucial part for efficiency of algorithms based on this theorem by Verma and Pearl is the *local* characterization of Markov equivalence. For structures that include cycles such a local characterization is not available. [Richardson, 1996a] presented the cyclic equivalence theorem which still leads to a polynomial algorithm, though the characterization is considerably more complicated.
Refer to [Richardson, 1996c], see also [Richardson, 1996a] for a more detailed description and all proofs.

Recall *Definition 2.5* of adjacent vertices in a (cyclic) directed graph. We have already seen in *Figure 3.1* that two Markov equivalent DCGs do not need to have the same real adjacencies, but if nodes i and j are really or virtually adjacent in one graph $\mathcal{G}_1$, then i and j are either really or virtually adjacent in $\mathcal{G}_2$ Markov equivalent to $\mathcal{G}_1$. From now on we will use the term adjacent to mean really or virtually adjacent.

In *Figure 3.1* we have the unshielded collider $1 \to 3 \leftarrow 4$ in the left graph, but not in the right one. So this condition is not necessary for Markov equivalence of directed cyclic graphs, hence the set of unshielded non-colliders also does not need to coincide in two Markov equivalent cyclic graphs. The cyclic pendant to an unshielded non-collider is an unshielded conductor.

Definition 3.3 *[Richardson, 1996c]. In a DG $\mathcal{G}$ an unshielded triple of vertices (i, j, k) forms an* unshielded conductor *if j is an ancestor of i or k. If for the unshielded triple (i, j, k), j is <u>not</u> an ancestor of i or k then (i, j, k) is called an* unshielded non-conductor*. Moreover an unshielded non-conductor (i, j, k) is called an* unshielded perfect non-conductor *if j is a descendant of a common child of i and k, otherwise (i, j, k) is an* unshielded imperfect non-conductor*.*

The cyclic pendant to an unshielded collider is therefore an unshielded perfect non-conductor, since they entail the same set of d-connection and d-separation relations. So an unshielded triple is either an unshielded conductor or an unshielded non-conductor, but we also distinguish between unshielded perfect and imperfect non-conductors. The latter can only occur in cyclic graphs, hence can be used as a criterion to detect feedback loops in directed graphs [Richardson, 1996c].
In *Figure 3.1* the unshielded triples (a, x, b) and (a, y, b) form unshielded imperfect non-conductors.

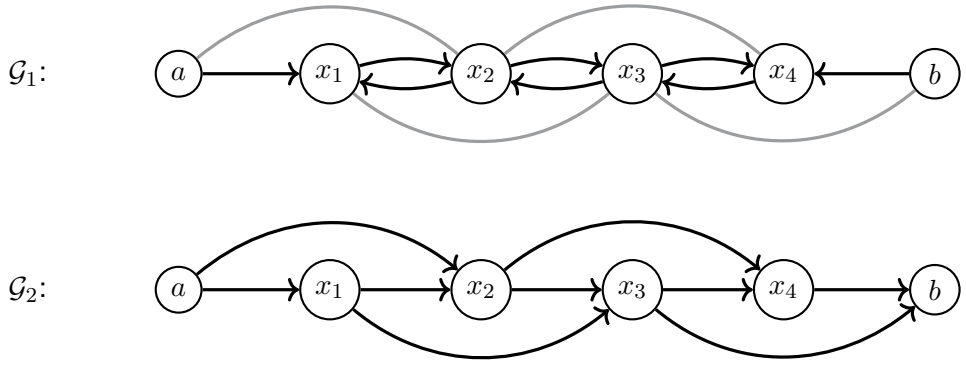


Figure 3.2 Locally Markov equivalent DCGs that are not Markov equivalent [Richardson, 1996c, Figure 6].

As we already mentioned, in acyclic graphs Markov equivalence can be decided only using local information of the graphs. The structures in the previous definition relate to triples of vertices, meaning they are also locally applicable, but these criteria are not sufficient for deciding Markov equivalence of cyclic graphs.

In *Figure 3.2*, $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same vertices and adjacencies, where virtual adjacencies are depicted as gray undirected edges. Moreover they have the same unshielded conductors and non-conductors, but still $\mathcal{G}_1$ and $\mathcal{G}_2$ are not Markov equivalent, since in $\mathcal{G}_1$, a and b are d-separated given the empty set, which is not the case for $\mathcal{G}_2$. Clearly we can extended this graphs by inserting more x_i's between a and b. So for checking Markov equivalence between DCGs we also need to check for certain types of long-range d-separation relations [Richardson, 1996c].

Definition 3.4 *[Richardson, 1996c]. An* itinerary $\varphi = (x_0, x_1, \ldots, x_{n+1})$ *is a sequence of vertices such that for all* i, $0 \leq i \leq n$, x_i *and* x_{i+1} *are adjacent. If for the itinerary* φ*, we have*

(i) for all t, $1 \leq t \leq n$, (x_{t-1}, x_t, x_{t+1}) *is an unshielded conductor,*

(ii) for all k, $1 \leq k \leq n$, x_{k-1} *and* x_{k+1} *are ancestors of* x_k*, and*

(iii) x_0 *is not a descendant of* x_1*, and* x_n *is not an ancestor of* x_{n+1}*,*

then (x_0, x_1, x_2) *and* (x_{n-1}, x_n, x_{n+1}) *are* mutually exclusive (m.e.) unshielded conductors *on the itinerary* $(x_0, \ldots, x_{n+1})$.
An itinerary φ *is an* uncovered itinerary *if the only vertices that are adjacent are those that occur consecutively on the itinerary.*

In *Figure 3.2* we have the uncovered itinerary (a, x_2, x_4, b) in $\mathcal{G}_1$ and $\mathcal{G}_2$, with the (unshielded) conductors (a, x_2, x_4) and (x_2, x_4, b). However, only in $\mathcal{G}_1$ condition *(ii)* and *(iii)* of m.e. conductors on the uncovered itinerary (a, x_2, x_4, b) are fulfilled. Note that for Markov equivalent DCGs the uncovered itineraries do not need to be the same.
We now state the Cyclic Equivalence Theorem of Thomas Richardson:

Theorem 3.2 *[Richardson, 1996c]. Directed cyclic graphs* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are Markov equivalent if and only if the following conditions hold:*

1. $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *have the same adjacencies.*
2. $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *have the same unshielded triples, i.e.*
   a) *the same unshielded conductors,*
   b) *the same unshielded perfect non-conductors, and*
   c) *the same unshielded imperfect non-conductors.*
3. (a, b, c) *and* (x, y, z) *are m.e. conductors on some uncovered itinerary* $\varphi_1 = (a, b, c, \ldots, x, y, z)$ *in* $\mathcal{G}_1$ *if and only if* (a, b, c) *and* (x, y, z) *are m.e. conductors on some uncovered itinerary* $\varphi_2 = (a, b, c, \ldots, x, y, z)$ *in* $\mathcal{G}_2$*.*
4. *If* (a, x, b) *and* (a, y, b) *are unshielded imperfect non-conductors in* $\mathcal{G}_1$ *and* $\mathcal{G}_2$*, then* x *is an ancestor of* y *in* $\mathcal{G}_1$ *if and only if* x *is an ancestor of* y *in* $\mathcal{G}_2$*.*
5. *If* (a, b, c) *and* (x, y, z) *are m.e. conductors on some uncovered itinerary* $\varphi = (a, b, c, \ldots, x, y, z)$ *and* (a, m, z) *is a unshielded imperfect non-conductor in* $\mathcal{G}_1$ *and* $\mathcal{G}_2$*, then* m *is a descendant of* b *in* $\mathcal{G}_1$ *if and only if* m *is a descendant of* b *in* $\mathcal{G}_2$*.*

3.3 Representing the Markov Equivalence Class

It can be convenient to represent the Markov equivalence class by a single mixed graph, i.e. a graph that contains directed as well as undirected edges.

3.3.1 Essential Graphs

We review the well known representation for the MEC of DAGs. In [Andersson et al., 1997] it is shown that there exists a graph $\mathcal{G}^*$, that uniquely represents $[\mathcal{G}]_M$ for $\mathcal{G}$ being a DAG. This graph is called the *essential graph* (also known as CPDAG).

Definition 3.5 *The essential graph $\mathcal{G}^*$ associated with $\mathcal{G}$ is the graph*

$$\mathcal{G}^* := \bigcup \{\mathcal{G}' : \mathcal{G}' \equiv_M \mathcal{G}\}.$$

So $\mathcal{G}^*$ is the smallest graph larger than every $\mathcal{G}' \in [\mathcal{G}]_M$. Therefore $\mathcal{G}^*$ has the same skeleton as $\mathcal{G}$ and an edge $i \to j$ in $\mathcal{G}^*$ is directed if and only if $i \to j$ occurs in every $\mathcal{G}' \in [\mathcal{G}]_M$; all other edges in $\mathcal{G}'$ are undirected. A directed edge in $\mathcal{G}^*$ is called an *essential arrow*, and it is clear by *Theorem 3.1* that every edge forming an unshielded collider in $\mathcal{G}$ must be essential in $\mathcal{G}^*$. So $\mathcal{G}$ and $\mathcal{G}^*$ have the same skeleton and unshielded colliders such that $\mathcal{G}_1 \equiv_M \mathcal{G}_2$ if and only if $\mathcal{G}_1^* = \mathcal{G}_2^*$. [Andersson et al., 1997] also showed how to construct the essential graph $\mathcal{G}^*$ from a DAG $\mathcal{G}$ and gives an atlas of essential graphs with $p \leq 4$ nodes. We will extend this atlas in *Section 5* by considering not only DAGs but also DCGs.

3.3.2 Ancestral Graphs

We now consider a class of graphs that can represent a wider range of causal relationships, namely *ancestral graphs*. Ancestral graphs were originally defined to represent structures that also contain latent variables without explicitly including latent variables in the model [Richardson and Spirtes, 2002].
A directed path from i to j with $i \leftrightarrows j$ in a mixed graph is called an *almost directed cycle*.

Definition 3.6 *A mixed graph $\mathcal{G}^*$ is called an **ancestral graph** if and only if $\mathcal{G}^*$ satisfies the following properties:*

- *There is no directed cycle.*
- *There is no almost directed cycle.*
- *For an undirected edge $i - j$, i and j have no incoming arrowheads.*

From this definition it follows that if $\mathcal{G}$ is an undirected graph or a DAG, then $\mathcal{G}$ is an ancestral graph [Richardson and Spirtes, 2002, Proposition 3.4].
In an ancestral graph a vertex i is said to be a *collider* if two arrowheads point towards i, i.e. $\to i \leftarrow$, $\leftrightarrow i \leftrightarrow$, $\to i \leftrightarrow$, $\leftrightarrow i \leftarrow$. A vertex on a path which is not a collider is a non-collider on the path.
Pearl's d-separation criterion for DGs can then be extended to mixed graphs. A path between vertices i and j in an ancestral graph G is said to be *m-connecting*, given a set $S \subset V \setminus \{i, j\}$, if

- every non-collider on the path is not in S, and
- every collider on the path is an ancestor of a member of S.

If there is no path m-connecting i and j given S, then i and j are said to be *m-separated* given S.

By the pairwise Markov property every missing edge in a DAG corresponds to a conditional independence constraint. This is in general not true for an arbitrary ancestral graph, which motivates the following definition: An ancestral graph is said to be *maximal*, if every missing edge corresponds to a conditional independence relation. So if i and j are not adjacent in a maximal ancestral graph (MAG) $\mathcal{G}$, then there is a set $S \subset V \setminus \{i, j\}$ such that i and j are m-separated given S. So MAGs are maximal in the sense that no edge may be added without changing the independence model [Richardson and Spirtes, 2002].
If now $\mathcal{G}$ is a MAG and $\mathcal{G}'$ is a subgraph of $\mathcal{G}$, then if $\mathcal{G}$ and $\mathcal{G}'$ have the same conditional independence constraints this implies $\mathcal{G} = \mathcal{G}'$. On the other side, by [Richardson and Spirtes, 2002, Theorem 5.1], for an ancestral graph $\mathcal{G}$, there exists a unique maximal ancestral graph $\mathcal{G}^*$ by adding $\leftrightarrow$ edges to $\mathcal{G}$ such that $\mathcal{G}$ and $\mathcal{G}^*$ have the same conditional independence constraints.

Several MAGs can also encode the same conditional independence relations via m-separation, so they are Markov equivalent. The characterization of Markov equivalence for DAGs (*Theorem 3.1*) does not hold for MAGs. By [Ali et al., 2009, Proposition 3.6] having the same adjacencies and unshielded colliders is still a necessary condition for Markov equivalence of MAGs. A sufficient condition is given by [Ali et al., 2009, Theorem 3.7]. Several Markov equivalent MAGs form a Markov equivalence class which is uniquely represented by a partial ancestral graph. [Zhang, 2008a] proposed the following representation of a Markov equivalence graph of MAGs, that contains the edges $\leftrightarrow, \rightarrow, \circ\!\!\rightarrow, \circ\!\!-\!\!\circ, \circ\!\!-$.

Definition 3.7 *[Zhang, 2008a, Definition 3]. Let $[\mathcal{G}]_M$ be the Markov equivalence class of an arbitrary MAG $\mathcal{G}$. The partial ancestral graph (PAG) for $[\mathcal{G}]_M$, Ψ, is a partial mixed graph such that*

- *Ψ has the same adjacencies as $\mathcal{G}$ (and any member of $[\mathcal{G}]_M$),*
- *an arrowhead is in Ψ if and only if it is in every MAG in $[\mathcal{G}]_M$,*
- *a tail is in Ψ if and only if it is in every MAG in $[\mathcal{G}]_M$.*

So basically a PAG represents a Markov equivalence class of MAGs by displaying all common edge marks $(>, <, -)$ shared by all members in the class, and displaying circles for those marks that are not common. So the PAG for a MAG can be seen as the essential graph for the Markov equivalence class of DAGs.

An edge $i \rightarrow j$, with an arrowhead at j then encodes that j is not an ancestor of i in every graph in the MEC, and the tail at i encodes that i is an ancestor of j in every graph in the MEC.
In this definition also bidirected edges can occur, $i \leftrightarrow j$, which then means i is not an ancestor of j, and j is not an ancestor of i in every graph in the MEC, so i and j are confounded by a latent variable.

It has been known that a DG with hidden variables also implies non-parametric constraints which generalize conditional independences, sometimes called 'Verma Constraints' [Richardson and Spirtes, 2002, Section 7.3.1]. Therefore [Shpitser et al., 2014] introduced *nested Markov models*, whereby both conditional independencies and these generalized constraints are used.

In the first Algorithm for discovering cyclic causal structures (CCD) the output of this algorithm is a PAG that represents a MEC. It is assumed that there are no latent confounders, so we do not need bidirected edges. Since we allow for cycles in the structure, the PAG must have some additional information such that it can fulfill the requirements of the cyclic equivalence theorem, i.e. one needs to specify an unshielded conductor and an unshielded imperfect non-conductor. Then the set of unshielded perfect non-conductors is already implied.

Here $*$ is used as a meta-symbol for one of $\{\circ, -, >\}$.
The following definition is due to Thomas Richardson [Richardson, 1996b].

Definition 3.8 *Ψ is a Partial Ancestral Graph (PAG) for the DG $\mathcal{G} = (V, E)$ if and only if:*

1. *There is an edge between i and j in Ψ if and only if i and j are d-connected in $\mathcal{G}$ given all subsets of vertices $S \subseteq V \setminus \{i, j\}$.*
2. *If $i - * j$ in Ψ, marked with a tail at i, then in every graph in $[\mathcal{G}]_M$, i is an ancestor of j.*
3. *If $i * -> j$ in Ψ, marked with an arrowhead at j, then in every graph in $[\mathcal{G}]_M$, j is not an ancestor of i.*
4. *If $i * - * j * - * k$ in Ψ, then j is an ancestor of i or k in every graph in $[\mathcal{G}]_M$.*
5. *If $i -> j <- k$ in Ψ, then in every graph in $[\mathcal{G}]_M$, j is not a descendant of a common child of i and k.*
6. *All endpoints not marked in the above ways is left with a circle, i.e. $\circ - *$.*

Note that only condition 1 above gives necessary and sufficient conditions about features of the PAG; all other conditions are merely necessary conditions. So there can be more that one PAG representing $[\mathcal{G}]_M$ [Richardson, 1996b]. To represent Markov equivalent cyclic structures, compared to the definition of Jiji Zhang, condition 4 and 5 are added. Condition 4 specifies the unshielded conductors and condition 5 the unshielded imperfect non-conductors in every DG in the MEC, hence every DG must also have the same unshielded perfect non-conductors, compare the Cyclic Equivalence Theorem [Richardson, 1996c].

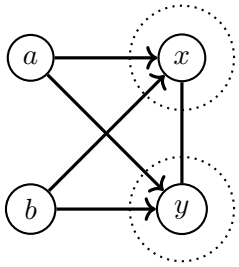


Figure 3.3 A partial ancestral graph representing $[\mathcal{G}]_M$ from *Figure 3.1*.

The PAG from *Figure 3.3* represents the MEC given in *Figure 3.1*. Note that there are only two graphs in $[\mathcal{G}]_M$. We can make following inferences about every graph in $[\mathcal{G}]_M$:

- a and b are ancestors of x and y,
- x and y are no ancestors of a or b,
- x is an ancestor of y, and y is an ancestor of x, so there is cycle between x and y,
- x and y are no descendants of a common child of a and b, so $a \to x \leftarrow b$ does not occur,

in every graph $\mathcal{G}' \in [\mathcal{G}]_M$.

We now present the Cyclic Causal Discovery (CCD) algorithm proposed by Thomas Richardson in 1996. The CCD algorithm was a first approach to discover a representative of the MEC of a possibly cyclic model. It performs conditional independence tests for constructing a partial ancestral graph. Since we will develop a new hybrid algorithm based on the CCD algorithm in *Section 6*, we review it here, for more information refer to the original paper [Richardson, 1996b].

CCD makes use of the following sets:

- $\mathbf{Adj}_{\Psi}(i)$ is the set of vertices adjacent to i in PAG Ψ.
- $i \in \mathbf{Local}_{\Psi}(k) \Leftrightarrow i$ is adjacent to k in Ψ, or there is a vertex j s.t. $i \to j \leftarrow k$ in Ψ.

Algorithm 1 Cyclic Causal Discovery [Richardson, 1996b]

Require: CI oracle for $\mathbb{P}$, satisfying the Markov and Faithfulness assumption w.r.t. a DG $\mathcal{G}$.
Ensure: PAG Ψ for the MEC $[\mathcal{G}]_M$.

Step 1 : Form PAG Ψ with edge $i \circ\!\!-\!\!\circ j$ between every pair of vertices.

$n \leftarrow 0$
repeat
  repeat
    Select i and j, adjacent in Ψ, s.t. $|\mathbf{Adj}_{\Psi}(i) \setminus \{j\}| \geq n$ and $S \subseteq \mathbf{Adj}_{\Psi}(i) \setminus \{j\}$ s.t. $|S| = n$.
    if $i \perp\!\!\!\perp j \mid S$ **then**
      delete $i \circ\!\!-\!\!\circ j$ from Ψ and record S in $\mathbf{Sepset}(i, j)$ and $\mathbf{Sepset}(j, i)$.
    end if
  until all adjacent i, j s.t. $|\mathbf{Adj}_{\Psi}(i) \setminus \{j\}| \geq n$ and all $S \subseteq \mathbf{Adj}_{\Psi}(i) \setminus \{j\}$ s.t. $|S| = n$ tested.
  $n \leftarrow n + 1$
until for all adjacent pairs i, j: $|\mathbf{Adj}_{\Psi}(i) \setminus \{j\}| < n$

Step 2 :

for all (i, j, k) in Ψ s.t i, j and j, k adjacent, but i, k not adjacent (i.e. $i {*\!\!-\!\!*} j {*\!\!-\!\!*} k$) **do**
  if $j \notin \mathbf{Sepset}(i, k)$ **then**
    orient as $i -\!\!> j <\!\!- k$
  else
    orient as $i {*\!\!-\!\!*} \underline{j} {*\!\!-\!\!*} k$
  end if
end for

Step 3 :

for all (a, i, j) in Ψ s.t. a not adjacent to i or j, i and j adjacent, $i \notin \mathbf{Sepset}(a, j)$ **do**
  if $a \not\perp\!\!\!\perp i \mid \mathbf{Sepset}(a, j)$ **then**
    orient $i {*\!\!-\!\!*} j$ as $i <\!\!- j$.
  end if
end for

Step 4 :

$m \leftarrow 0$
repeat
  repeat
    Select (i, j, k) s.t. $i -\!\!> j <\!\!- k$, i and k not adjacent, $|\mathbf{Local}_{\Psi}(i) \setminus \{j, k\}| \geq m$,
    and a set $T \subseteq \mathbf{Local}_{\Psi}(i) \setminus \{j, k\}$ s.t. $|T| = m$
    if $i \perp\!\!\!\perp k \mid T \cup \{j\}$ **then**
      orient as $i -\!\!\!\gtrdot j \lessdot\!\!\!- k$, and record $T \cup \{j\}$ in $\mathbf{Supset}(i, j, k)$
    end if
  until all triples s.t. $i -\!\!> j <\!\!- k$, $|\mathbf{Local}_{\Psi}(i) \setminus \{j\}| \geq m$, $T \subseteq \mathbf{Local}_{\Psi}(i)$ s.t. $|T| = m$ tested.
  $m \leftarrow m + 1$
until for all such triples (i, j, k): $|\mathbf{Local}_{\Psi}(i) \setminus \{j, k\}| < m$

Algorithm 1 Cyclic Causal Discovery (continued)

Step 5 :

if (i, j, k, l) in Ψ exists s.t. $i -\!\!\!\underset{\cdot\cdot}{>} j \underset{\cdot\cdot}{<}\!\!\!- k$, $i -\!\!> j <\!\!- k$ or $i -\!\!\!\underset{\cdot\cdot}{>} j \underset{\cdot\cdot}{<}\!\!\!- k$, j and l adjacent **then**
  if $l \notin \mathbf{Supset}(i, j, k)$ **then**
    orient $j *\!\!- *\, l$ as $j -\!\!> l$
  else
    orient $j *\!\!- *\, l$ as $j *\!\!- l$
  end if
end if

Step 6 :

for all (i, j, k, l) in Ψ s.t. l not adjacent to i and k, and $i -\!\!\!\underset{\cdot\cdot}{>} j \underset{\cdot\cdot}{<}\!\!\!- k$ **do**
  if $i \not\perp\!\!\!\perp l \mid \mathbf{Supset}(i, j, k) \cup \{l\}$ **then**
    orient $j *\!\!- *\, l$ as $j -\!\!> l$
  end if
end for

Given as input a conditional independence oracle for the distribution $\mathbb{P}$, satisfying the Markov and Faithfulness assumption w.r.t. a (cyclic or acyclic) graph $\mathcal{G}$, it has been shown that the CCD algorithm outputs a PAG Ψ for $\mathcal{G}$ (Soundness, [Richardson, 1996a, Theorem 1]).

In practice the conditional independence oracle is replaced by statistical tests for conditional independence in the sample data.
Moreover the CCD algorithm is *d-separation complete*, so given independence oracles for distributions $\mathbb{P}_1,\ \mathbb{P}_2$ satisfying the Markov and Faithfulness assumption w.r.t. graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, CCD outputs PAGs Ψ_1 and Ψ_2. Then $\Psi_1 = \Psi_2$ if and only if $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent [Richardson, 1996a, Theorem 2].
The CCD algorithm is not *complete* in the sense that it gives the most informative PAG for a given DG, so there might be features common to all graphs in the equivalence class that are not captured by the PAG that the CCD algorithm outputs (see also [Zhang, 2008b] for completion of orientation rules in causal discovery).

4 Distribution Equivalence

We now consider a stronger form of equivalence between two directed graphs, but first a short historical review.
In [Richardson, 1996a, Section 2.4] it is already mentioned that there exists a stronger sense of equivalence, called linear statistical equivalence (also called distribution equivalence), which holds when every distribution described by a linear parametrization of one graph can also described by a linear parametrization of the other graph. It is then mentioned that Markov equivalence and distribution equivalence are equivalent for acyclic graphs, but it is left as an open question if it is the case for cyclic structures.
In [Spirtes et al., 1993, Section 12.2] it is stated that if the family of distributions represented by a DAG is multivariate Gaussian, multinomial, or unrestricted, two DAGs without latent variables or selection bias are distribution equivalent if and only if they are Markov equivalent, but this does not hold if the DAGs contain latent variables or if there is sample selection bias. Peter Spirtes also emphasizes the importance of distribution equivalence, because if two structures are distribution equivalent, then it is to be expected that data can not help to discriminate them.
In [Heckerman and Geiger, 1995] distribution equivalence is also called likelihood equivalence. They make the following statement: Two structures that only differ by a single covered arc reversal are distribution equivalent. An arc reversal is a transformation between two structures, where a single arrow is reversed, and an arrow between two nodes is covered if those two nodes would have the same parents if the arrow were removed. We will later call a covered arc reversal a parent exchange in the following sections.
Moreover [Lacerda et al., 2012] mentioned, in a work about the LiNG algorithm, that for cyclic graphs, distribution equivalence entails Markov equivalence. On the other side, Markov equivalence does *not* entail distribution equivalence.
Finally, a characterization of distribution equivalence was released by [Ghassami et al., 2020]. It is mentioned, that in the presence of cycles, d-separation relations do not reflect all information in the distribution, so there might be DGs, with the same conditional independence constraints, that can be distinguished using observational data. In this work it is stated that two (possibly cyclic) structures can be transformed into one another by applying a sequence of rotations on the support matrix. This operations include parent exchanges (covered arc reversals) and cycle reversions. Moreover there is a third operation called a parent reduction, which can only we applied in presence of a 2-cycle $i \leftrightarrows j$ and leads to a graph with one edge less.
Amir Emad Ghassami also presented an algorithm for computing the set of graphs that are distribution equivalent, called distribution equivalence class.

The following section are then mainly based on the publication by [Ghassami et al., 2020].

Let us first introduce the notion of distribution equivalence, that means we consider a class of structures which all can generate the same set of distributions. Following the definition of the original paper, instead of the covariance matrix (2.5), the *precision matrix* $\Theta = \Sigma^{-1}$ contains all the information regarding the distribution.

Definition 4.1 *[Ghassami et al., 2020, Definition 2]. The* distribution set *of $\mathcal{G}$ is defined by*

$$\Theta(\mathcal{G}) := \{\Theta : \Theta = (I - B)\Omega^{-1}(I - B)^T, \textit{for any } (B, \Omega) \textit{ s.t. } \Omega \in \textit{diag}^+ \textit{ and } \operatorname{supp}(B) \subseteq \operatorname{supp}(B_{\mathcal{G}})\}, \tag{4.1}$$

where diag$^+$ is the set of diagonal matrices with positive entries, $B_{\mathcal{G}}$ is the binary adjacency matrix of $\mathcal{G}$, and $\operatorname{supp}(B) = \{(i, j) : B_{ij} \neq 0\}$.

So $\Theta(\mathcal{G})$ contains all the precision matrices (or equivalently, distributions) that can be generated by a structure $\mathcal{G}$.

Definition 4.2 *Two DGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are* distribution equivalent*, denoted by $\mathcal{G}_1 \equiv_D \mathcal{G}_2$, if $\Theta(\mathcal{G}_1) = \Theta(\mathcal{G}_2)$. Moreover the distribution equivalence class (DEC) is defined by*

$$[\mathcal{G}]_D := \{\mathcal{G}' : \Theta(\mathcal{G}') = \Theta(\mathcal{G})\}. \tag{4.2}$$

4.1 Baseline equivalence test

To test distribution equivalence of two structures G_1 and G_2 consider the following procedure [Ghassami et al., 2020]: Let $\Theta \in \Theta(\mathcal{G}_1)$, then we have to check if there exists a choice of exogenous noise variances Ω and edge weights B such that structure $\mathcal{G}_2$ generates Θ. Repeat this by considering $\mathcal{G}_2$ as the given structure and search for a choice of parameters such that $\mathcal{G}_1$ generates $\Theta \in \Theta(\mathcal{G}_2)$.
This can be realized by an equivalence test based on rotations on the matrix. Let v_i be the i-th row of a matrix

$$Q = (I - B)\Omega^{-\frac{1}{2}}. \tag{4.3}$$

Then $\Theta = QQ^T$ is the Gramian matrix of the set of vectors $\{v_1, \dots, v_p\}$.
Given $Q_1 Q_1^T = \Theta$, we have $Q_2 Q_2^T = \Theta$ if and only if $Q_2 = Q_1 U$ for some orthogonal transformation U, since an orthogonal transformation applied to a Gramian matrix results in the same Gramian matrix. Note that a rotation applied to a matrix is an orthogonal transformation.
Let $Q_{\mathcal{G}} := I + B_{\mathcal{G}}$, then for all choices of B and Ω, $\mathrm{supp}(Q) \subseteq \mathrm{supp}(Q_{\mathcal{G}})$.

Proposition 4.1 *[Ghassami et al., 2020, Proposition 2]. We have $\mathcal{G}_1 \equiv_D \mathcal{G}_2$ if and only if, for any Q_1 there exists a rotation $U^{(1)}$ such that $\mathrm{supp}(Q_1 U^{(1)}) \subseteq \mathrm{supp}(Q_{G_2})$, and for any Q_2 there exists a rotation $U^{(2)}$ such that $\mathrm{supp}(Q_2 U^{(2)}) \subseteq \mathrm{supp}(Q_{G_1})$.*

Proof: (see supplementary material of [Ghassami et al., 2020])
"$\Longrightarrow$": If $\mathcal{G}_1$ is distribution equivalent to $\mathcal{G}_2$, then for all Q_1 that generate $\Theta = Q_1 Q_1^\top$, there exists a choice of Q_2 such that $Q_2 Q_2^\top = \Theta$. Since Q_2 is generated by $\mathcal{G}_2$, we have by definition

$$\mathrm{supp}(Q_2) \subseteq \mathrm{supp}(Q_{\mathcal{G}_2}). \tag{4.4}$$

Since $Q_1 Q_1^\top = \Theta$ and $Q_2 Q_2^\top = \Theta$, we have

$$Q_2 = Q_1 U \tag{4.5}$$

for some orthogonal transformation U. This is due to the fact that the generating vectors of a Gramian matrix can be determined up to isometry.
Therefore, by (4.4) and (4.5) it follows that

$$\mathrm{supp}(Q_1 U) \subseteq \mathrm{supp}(Q_{\mathcal{G}_2}). \tag{4.6}$$

It is left to show that there exists a rotation $U^{(1)}$ such that

$$\mathrm{supp}(Q_1 U^{(1)}) \subseteq \mathrm{supp}(Q_{G_2}). \tag{4.7}$$

Since a rotation matrix U is an orthogonal transformation, we have $UU^\top = I$ and $\det(U) = \pm 1$. If $\det(U) = 1$, then U is a proper rotation and we can choose $U^{(1)} = U$. If $\det(U) = -1$, then U is an improper rotation. So we need to find an orthogonal transformation such that $\det(V) = -1$, i.e.

$\det(UV) = 1$. Moreover V should not change the support matrix, i.e. $\text{supp}(Q_1 U) = \text{supp}(Q_1 UV)$. Then UV is a rotation. A matrix V is found by using a diagonal matrix that has diagonal entries ± 1 such that $\det(V) = -1$, then V is a reflection matrix and changes the improper rotation U into a proper rotation UV. We are done by choosing $U^{(1)} = UV$.

"$\Longleftarrow$": If $\text{supp}(Q_1 U^{(1)}) \subseteq \text{supp}(Q_{\mathcal{G}_2})$ then we can choose the entries of $Q_1 U^{(1)}$ as Q_2. Therefore

$$Q_2 Q_2^\top = Q_1 U^{(1)} (U^{(1)})^\top Q_1^\top = Q_1 Q_1^\top. \tag{4.8}$$

So Q_2 can generate the distributions generated by Q_1. This is true for all choices of Q_1, and true if we start with the reverse Q_2. Then $\mathcal{G}_1$ is distribution equivalent to $\mathcal{G}_2$.

□

4.2 Givens rotations

To utilize these orthogonal transformations the use of *Givens rotations* is introduced by [Ghassami et al., 2020], which is a special type of planar rotation spanned by two coordinate axes [Golub and Van Loan, 1996]. For a θ-radian rotation in the (j, k) plane we define the Givens rotation matrix:

Definition 4.3 *The Givens rotation matrix $G(j, k, \theta) = [g]_{p\times p}$ in $\mathbb{R}^p$ is defined as:*

$$G(j,k,\theta) = \begin{cases} g_{i,i} = 1 & \text{for } i \notin j,k \\ g_{i,i} = \cos(\theta) & \text{for } i \in j,k \\ g_{k,j} = -g_{j,k} = -\sin(\theta) & \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

Example 4.1 *To show the effect of a Givens rotation on a matrix Q consider the following example: Let $Q \in \mathbb{R}^{3\times 3}$ and $j = 2, k = 3$, then*

$$G(j,k,\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

is the Givens rotation matrix. If we multiply G from the right side to Q, we get

$$QG = \begin{pmatrix} Q_{1,1} & Q_{1,2}\cos\theta + Q_{1,3}\sin\theta & -Q_{1,2}\sin\theta + Q_{1,3}\cos\theta \\ Q_{2,1} & Q_{2,2}\cos\theta + Q_{2,3}\sin\theta & -Q_{2,2}\sin\theta + Q_{2,3}\cos\theta \\ Q_{3,1} & Q_{3,2}\cos\theta + Q_{3,3}\sin\theta & -Q_{3,2}\sin\theta + Q_{3,3}\cos\theta \end{pmatrix}.$$

Now for a row index $i = 1$ the entry $(QG)_{1,2}$ can be set to zero using a special angle θ^*, i.e.

$$\begin{aligned} Q_{1,2}\cos\theta + Q_{1,3}\sin\theta &= 0 \\ \frac{\sin\theta}{\cos\theta} &= \frac{-Q_{1,2}}{Q_{1,3}} \\ \theta^* &= \tan^{-1}(-Q_{1,2}/Q_{1,3}). \end{aligned}$$

Observe that there is no real-valued solution such that $(QG)_{1,3}$ can also be set to zero using θ^*. So it is not possible to set both entries in the i-th axis of the (j, k) plane to zero.
However there may exist a row index $l \in [p] \setminus \{i\}$ for which we have

$$\frac{-Q_{1,2}}{Q_{1,3}} = \frac{-Q_{l,2}}{Q_{l,3}} \quad \textit{or} \quad \frac{-Q_{1,2}}{Q_{1,3}} = \frac{Q_{l,3}}{Q_{l,2}},$$

meaning that we would also set the $(QG)_{l,2}$ or $(QG)_{l,3}$ entry to zero.

4.3 Support Rotations

Definition 4.4 *For any matrix Q (eq. 4.3) its support matrix ξ is a binary matrix with same size and entries in $\{0, \times\}$, where $\xi_{i,j} = \times$ if $Q_{i,j} \neq 0$ and $\xi_{i,j} = 0$ if $Q_{i,j} = 0$.*

In the following we define the support rotation which operates on the support matrix ξ.

Definition 4.5 *[Ghassami et al., 2020, Definition 5]. A support rotation, $A(i, j, k)$, takes a support matrix ξ as input and sets $\xi_{i,j}$ to zero using a Givens rotation in the (j, k) plane.*
The output is the support matrix of $QG(j, k, \tan^{-1}(-Q_{i,j}/Q_{i,k}))$, where

$$Q \in \arg\max_{Q'} |\operatorname{supp}(Q'G(j, k, \tan^{-1}(-Q_{i,j}/Q_{i,k})))|,$$

such that the support matrix of Q' is ξ.

The support rotation $A(i, j, k)$ only affects the j-th and k-th column. The general effect of a support rotation $A(i, j, k)$ on a support matrix ξ is as follows (compare [Ghassami et al., 2020, Proposition 3]):

- If $\xi_{i,j} = 0$, then $\theta = \tan^{-1}(0) = 0$ and $G(j, k, 0) = Id$, so $A(i, j, k)$ has no effect on ξ.
- If $\xi_{i,j} = \xi_{i,k} = \times$, then $A(i, j, k)$ sets $\xi_{i,j} = 0$ as described above.
  Moreover we distinguish the following subcases:
  - **Reduction:** If for every $l \in [p] \setminus \{i\}$ we have $\xi_{l,j} = \xi_{l,k}$, then there is no other change in support.
  - **Reversible acute rotation:** If there exists $i' \in [p] \setminus \{i\}$ such that $\xi_{i',j} \neq \xi_{i',k}$ only in that row, then $A(i, j, k)$ additionally sets $\xi_{i',j} = \xi_{i',k} = \times$. If $\xi_{i',j} = 0$ (i.e. $\xi'_{i',j} = \times$), then this transformation can be reversed by applying $A(i', j, k)$ on $\xi' = \xi A(i, j, k)$.
  - **Irreversible acute rotation:** It there exist at least two rows such that the j-th and k-th column of ξ differ, then all these entries are set to $\times$.
- **Column swap:** If $\xi_{i,j} = \times$ and $\xi_{i,k} = 0$, then $\theta = \tan^{-1}(\pm\infty) = \pm\pi/2$ and columns j and k are switched.

Note that a reversible acute rotation and column swap does not change the number of "$\times$" of $\operatorname{supp}(\xi)$. A reduction sets only one entry to zero and leaves the rest of the support as it is, thus the name, while a irreversible acute rotation sets one entry to 0 and sets at least two entries to $\times$.

We now state the main theorem of the paper which gives necessary and sufficient conditions of distribution equivalence of two structures using the introduced transformations.
For two support matrices ξ and ξ' we say $\xi \subseteq \xi'$ if $\operatorname{supp}(\xi) \subseteq \operatorname{supp}(\xi')$.

Theorem 4.1 *[Ghassami et al., 2020, Theorem 1]. Let ξ_1 and ξ_2 be support matrices of DGs $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively.*
$\mathcal{G}_1 \equiv_D \mathcal{G}_2$ if and only if there exists a sequence of reductions, reversible acute rotations and column swaps that maps ξ_1 to a subset of ξ_2, and a sequence that maps ξ_2 to a subset of ξ_1.

The theorem shows that irreversible acute rotations are not needed to transform a support ξ into an equivalent support ξ'. Moreover the theorem allows one to implement a search procedure that finds all equivalent structures by first applying all reversible acute rotations and reductions on a support, and then by applying all column swaps.

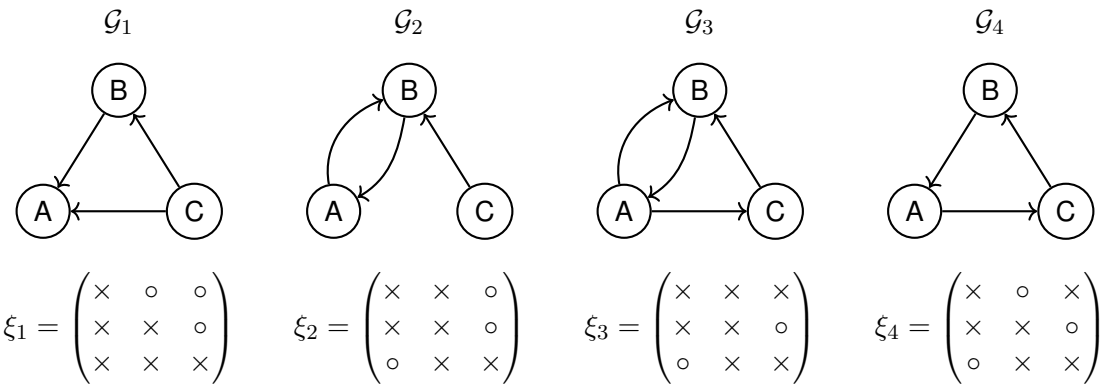


Figure 4.1 Directed graphs on 3 nodes with their corresponding support matrices.

Example 4.2 *In Figure 4.1 consider the four DGs on 3 nodes. We have $\mathcal{G}_1 \equiv_D \mathcal{G}_2 \equiv_D \mathcal{G}_3 \not\equiv_D \mathcal{G}_4$.*
For the first equivalence we just have to change one entry by a reversible acute rotation, setting $(\xi_1)_{3,1} = 0$, to get

$$\xi_1 = \begin{pmatrix} \times & \circ & \circ \\ \times & \times & \circ \\ \times & \times & \times \end{pmatrix} \xrightarrow{A(3,1,2)} \begin{pmatrix} \times & \times & \circ \\ \times & \times & \circ \\ \circ & \times & \times \end{pmatrix} = \xi_2$$

The reverse of this rotation would be $\xi_2 \xrightarrow{A(1,2,1)} \xi_1$.
To show $\mathcal{G}_2 \equiv_D \mathcal{G}_3$ we already have $\xi_2 \subseteq \xi_3$, furthermore

$$\xi_3 = \begin{pmatrix} \times & \times & \times \\ \times & \times & \circ \\ \circ & \times & \times \end{pmatrix} \xrightarrow{A(2,1,2)} \begin{pmatrix} \times & \times & \times \\ \circ & \times & \circ \\ \times & \times & \times \end{pmatrix} \xrightarrow{A(1,3,1)} \begin{pmatrix} \times & \times & \circ \\ \circ & \times & \circ \\ \times & \times & \times \end{pmatrix} \xrightarrow{A(3,1,2)} \begin{pmatrix} \times & \times & \circ \\ \times & \times & \circ \\ \circ & \times & \times \end{pmatrix} = \xi_2,$$

where in the first and third step we used a reversible acute rotation. In the second step we used a reduction, which is necessary since $\mathcal{G}_3$ has an edge more than $\mathcal{G}_2$ (and $\mathcal{G}_1$).
The graph $\mathcal{G}_4$ is not distribution equivalent to any of the other graphs. This can be seen by the fact that we cannot do any reversible acute rotation (or reduction), since for every two columns j, k, there are rows i and i' such that $\xi_{i,j} \neq \xi_{i,k}$ and $\xi_{i',j} \neq \xi_{i',k}$. The only operation is a column swap on the three nodes which leads to a graph with the directed cycle reversed.

Topological closure of the Distribution set. We now use the above example to show that in some cases, two graphs that are distribution equivalent, based on *Theorem 4.1*, do not completely represent the same distribution set.

Consider *Figure 4.1*. For the complete DAG on 3 nodes, $\mathcal{G}_1$, we know that every distribution can be generated. This follows from the Cholesky decomposition and the fact that the adjacency matrix of a DAG can be brought in lower triangular form. If we consider $Q = (I - B)\Omega^{1/2}$, then Q is lower unit triangular and by the LDL decomposition, $\Sigma = QQ^\top$ is positive definite [Golub and Van Loan, 1996].

For the following argumentation compare also [Ghassami et al., 2020] for a more detailed description.
By *Definition 4.2*, $\mathcal{G}_1$ and $\mathcal{G}_2$ are distribution equivalent if every positive definite precision matrix can be represented by $\mathcal{G}_2$. Consider an arbitrary precision matrix Θ generated by $\mathcal{G}_1$. If Θ can be represented by $\mathcal{G}_2$, then it can be decomposed as $\Theta = QQ^\top$. If suffices to show that Q is a matrix of the form

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & e & f \end{pmatrix} \tag{4.10}$$

such that

$$\begin{aligned} a^2 + b^2 = \Theta_{1,1} \quad & ac + bd = \Theta_{1,2} \\ c^2 + d^2 = \Theta_{2,2} \quad & be = \Theta_{1,3} \\ e^2 + f^2 = \Theta_{3,3} \quad & de = \Theta_{2,3}. \end{aligned}$$

Fix the value of e. Then we can uniquely solve for b, d, f and a, c. The obtained values then need to satisfy also $ac + bd = \Theta_{1,2}$, where we arrive at the equality

$$\det \begin{pmatrix} \Theta_{1,1} & \Theta_{1,2} \\ \Theta_{1,2} & \Theta_{2,2} \end{pmatrix} e^2 = \Theta_{1,1}\Theta_{2,3}^2 + \Theta_{2,2}\Theta_{1,3}^2 - 2\Theta_{1,2}\Theta_{1,3}\Theta_{2,3}.$$

Since Θ is positive definite, also the subdeterminant on the left hand side is positive. By plugging in the values of Θ one can also show that the right hand side is also positive, which shows distribution equivalence by [Ghassami et al., 2020].

In the above it is assumed that e is fixed. We show that there are distributions that can not be represented by $\mathcal{G}_2$ if the value of e tends to zero. For this purpose let $\Theta = QQ^\top$ be the precision matrix generated by $\mathcal{G}_2$. Then

$$\Theta = \begin{pmatrix} a^2 + b^2 & ac + bd & be \\ ac + bd & c^2 + d^2 & de \\ be & de & e^2 + f^2 \end{pmatrix}. \tag{4.11}$$

By the above argumentation we have $\Theta(\mathcal{G}_2) \subseteq \Theta(\mathcal{G}_1)$, the reverse implication does not hold:

Proposition 4.2 *The distribution set of $\mathcal{G}_2$ from Figure 4.1 is*

$$\Theta(\mathcal{G}_2) = \{\Theta : \Theta \textit{ positive definite and } \Theta_{32} = 0 \textit{ implies } \Theta_{31} = 0\}.$$

Proof:
Let Θ be a precision matrix in the set $\Theta(\mathcal{G}_2)$. Then Θ positive definite since $\Theta(\mathcal{G}_2) \subset \Theta(\mathcal{G}_1)$, and by equation (4.11), $\Theta_{3,2} = de$. We have $d = 1/\sigma_2$ where σ_2 is the standard deviation of the exogenuous variable ε_2, so σ_2 is bounded. If $\Theta_{3,2} = de \to 0$, then $e \to 0$. But then also $\Theta_{3,1} = be \to 0$.

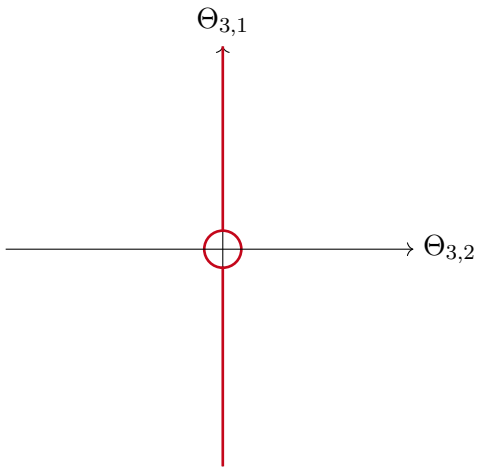


Figure 4.2 The distribution set of $\mathcal{G}_2$ wrt. $\Theta_{3,2}$ and $\Theta_{3,1}$.

Suppose Θ is positive definite such that $\Theta_{3,2} = 0$ implies $\Theta_{3,1} = 0$.
Case 1: $\Theta_{3,2} \neq 0$:
Choose Q to be the matrix as in equation (4.10), then $QQ^\top = \Theta$. It follows that $\Theta \in \Theta(\mathcal{G}_2)$.

Case 2: $\Theta_{3,2} = 0$:
Now take

$$Q = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & \mathbf{0} & f \end{pmatrix}.$$

Then $\Theta = QQ^\top$ is in $\Theta(\mathcal{G}_2)$ with

$$\begin{pmatrix} a^2 + b^2 & ac + bd & 0 \\ ac + bd & c^2 + d^2 & 0 \\ 0 & 0 & f^2 \end{pmatrix}.$$

□

So $\Theta(\mathcal{G}_2)$ does not contain any precision matrices where $\Theta_{3,2} = 0$ and $\Theta_{3,1} \neq 0$, compare *Figure 4.2*. These are limit points of sequences of precision matrices where the edge weight e tends to zero, hence the set $\Theta(\mathcal{G}_2)$ is not closed. However, if we take the topological closure of the distribution set Θ, denoted by $\overline{\Theta}$, then

$$\overline{\Theta(\mathcal{G}_2)} = \overline{\Theta(\mathcal{G}_1)}.$$

Consider now the complete DAG $\mathcal{G}_1$, and the directed cycle on 3 nodes $\mathcal{G}_4$. It is already stated by *Theorem 4.1* that $\mathcal{G}_1 \not\equiv_D \mathcal{G}_3$. We can also show this by the following:
Let Q be the matrix that represents $\mathcal{G}_4$. Then Q takes the form

$$\begin{pmatrix} a & b & 0 \\ 0 & c & d \\ e & 0 & f \end{pmatrix}. \tag{4.12}$$

A distribution Θ is in $\Theta(\mathcal{G}_4)$ if and only if there exist $a, b, c, d, e, f \in \mathbb{R}$ such that $QQ^\top = \Theta$. Now define

$$\tilde{Q} = \begin{pmatrix} a & -b & 0 \\ 0 & c & d \\ e & 0 & f \end{pmatrix} \tag{4.13}$$

as Q but the entry $Q_{1,2}$ is negated. Then

$$\tilde{Q}\tilde{Q}^\top = \begin{pmatrix} \Theta_{1,1} & -\Theta_{1,2} & \Theta_{1,3} \\ -\Theta_{1,2} & \Theta_{2,2} & \Theta_{2,3} \\ \Theta_{1,3} & \Theta_{2,3} & \Theta_{3,3} \end{pmatrix} =: \Theta^-. \tag{4.14}$$

We have $\Theta^- \in \Theta(\mathcal{G}_4)$ and Θ^- is positive definite.
By [Drton and Yu, 2010, Lemma 5.1] a precision matrix Θ is in $\Theta(\mathcal{G})$ if and only if Θ^- is in $\Theta(\mathcal{G})$, where in Θ^- one entry is negated.
The necessary condition on Θ is that $\det(\Theta^-) > 0$.

Now let

$$\Theta_\rho = \begin{pmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{pmatrix}. \tag{4.15}$$

If $\Theta_\rho \in \Theta(\mathcal{G}_4)$, then we must have

$$\det \begin{pmatrix} 1 & -\rho & \rho \\ -\rho & 1 & \rho \\ \rho & \rho & 1 \end{pmatrix} > 0. \tag{4.16}$$

Choose $\rho = 1$, then the determinant (4.16) equals -2. So Θ_ρ^- is not positive definite, hence Θ_ρ is not in $\Theta(\mathcal{G}_4)$ (compare also [Drton and Yu, 2010, Example 5.2]).
So not every positive definite matrix remains positive definite after negating one entry. In general the directed cycle on 3 nodes can not represent high correlations between the variables. This shows again that the complete DAG and the directed cycle are not distribution equivalent.

In structure learning we usually only have one distribution at hand and the task is to find the data-generating structure. Clearly, there are distributions generated by the complete DAG that can also be generated by the directed cycle. So in general it is not possible to discover the ground truth structure given only one distribution. [Ghassami et al., 2020] therefore proposed a weaker notion of equivalence, called *quasi equivalence* (compare *Definition 6.1*), which states that two structures are quasi equivalent if the distributions they can both generate has non-zero Lebesgue measure.
For graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ this is clearly the case. One can also show that the complete DAG, $\mathcal{G}_1$, and the directed cycle, $\mathcal{G}_4$, are quasi equivalent. Take again the matrix Q (4.12), then

$$QQ^\top = \begin{pmatrix} a^2+b^2 & bc & ae \\ bc & c^2+d^2 & df \\ ae & df & e^2+f^2 \end{pmatrix}. \tag{4.17}$$

Fix the value of e, solve for a, b, c, d, f, then we arrive at

$$(\Theta_{2,2}\Theta_{1,1} - \Theta_{1,2}^2)e^4 + (-\Theta_{1,1}\Theta_{2,2}\Theta_{3,3} - \Theta_{2,2}\Theta_{1,3}^2 + \Theta_{1,1}\Theta_{2,3}^2 + \Theta_{3,3}\Theta_{1,2}^2)e^2 + (\Theta_{2,2}\Theta_{3,3}\Theta_{1,3}^2 - \Theta_{2,3}^2\Theta_{1,3}^2) = 0.$$

The solution to this equation is not always real, and only for a non-measure zero subset of the distributions satisfied [Ng et al., 2020].

4.4 Graphical Characterization of Distribution Equivalence

In the previous section we presented *Theorem 4.1* that gives necessary and sufficient conditions for distribution equivalence of two DGs based on Givens rotations applied to the support of the DGs [Ghassami et al., 2020, Theorem 1]. The following section transfers this results over to graphical operations, so we present the graphical counterpart of the theorem (compare [Ghassami et al., 2020, Section 4].

Definition 4.6 *[Ghassami et al., 2020, Definition 8]. Let $\mathcal{G}$ be a DG. For a vertex i, let $P_i = \mathbf{Pa}_{\mathcal{G}}(i) \cup \{i\}$. Then i and j are* parent reducible *if $P_i = P_j$, and* parent exchangeable *if $|P_i \Delta P_j| = 1$, where Δ is the symmetric difference operator.*

The three support rotations on the support matrix ξ of a DG $\mathcal{G}$ then lead to the following (graphical) operations:

- **Parent Reduction:** If i and j are parent reducible, then a support rotation of reduction rotation type on the i-th and j-th column of ξ removes a parent of i or j.
- **Parent Exchange:** If i and j are parent exchangeable, then there exists a $P_i \Delta P_j = \{k\}$. The support rotation of reversible acute rotation type removes the parent of either i or j. Additionally an edge from k is added to either i or j.
- **Cycle Reversion:** A cycle reversion swaps the columns of each node of a cycle C with the column corresponding to the predecessor in the cycle. This operation reverses the direction of the cycle and every edge pointing to a node in C will be pointing to the predecessor of the node.

So for parent exchange between two vertices i and j we need $|P_i \ \Delta \ P_j| = 1$, which means that i and j must be adjacent. Assume for a DG $\mathcal{G}$ there is only one edge between i and j, i.e. no 2-cycle $i \leftrightarrows j$. Then i and j must have the same parents. We can either reverse this edge between i and j, or remove a parent from i and j (if there are some), which adds the edge that leads to $i \leftrightarrows j$. This means for parent exchangeable nodes i and j there is one operation that keeps the number of 2-cycles in $\mathcal{G}$, and $|P_i \cap P_j| - 1$ operations that add a 2-cycle $i \leftrightarrows j$.
For a parent reduction on i and j we must have $i \leftrightarrows j$, since otherwise $|P_i \ \Delta \ P_j| \neq 0$.

Theorem 4.2 *[Ghassami et al., 2020, Theorem 2]. For DGs $\mathcal{G}_1$ and $\mathcal{G}_2$ we have $\mathcal{G}_1 \equiv_D \mathcal{G}_2$ if and only if there exists a sequence of parent reductions, parent exchanges and cycle reversions that maps $\mathcal{G}_1$ to a subgraph of $\mathcal{G}_2$, and a sequence that maps $\mathcal{G}_2$ to a subgraph of $\mathcal{G}_1$.*

4.5 Example of a Distribution Equivalence Class

We now give an example of a specific DEC and show how every element in this DEC is connected to each other.
Consider the DGs on 4 nodes in *Figure 4.3* which forms a DEC (compare [Ghassami et al., 2020, Figure 5]).

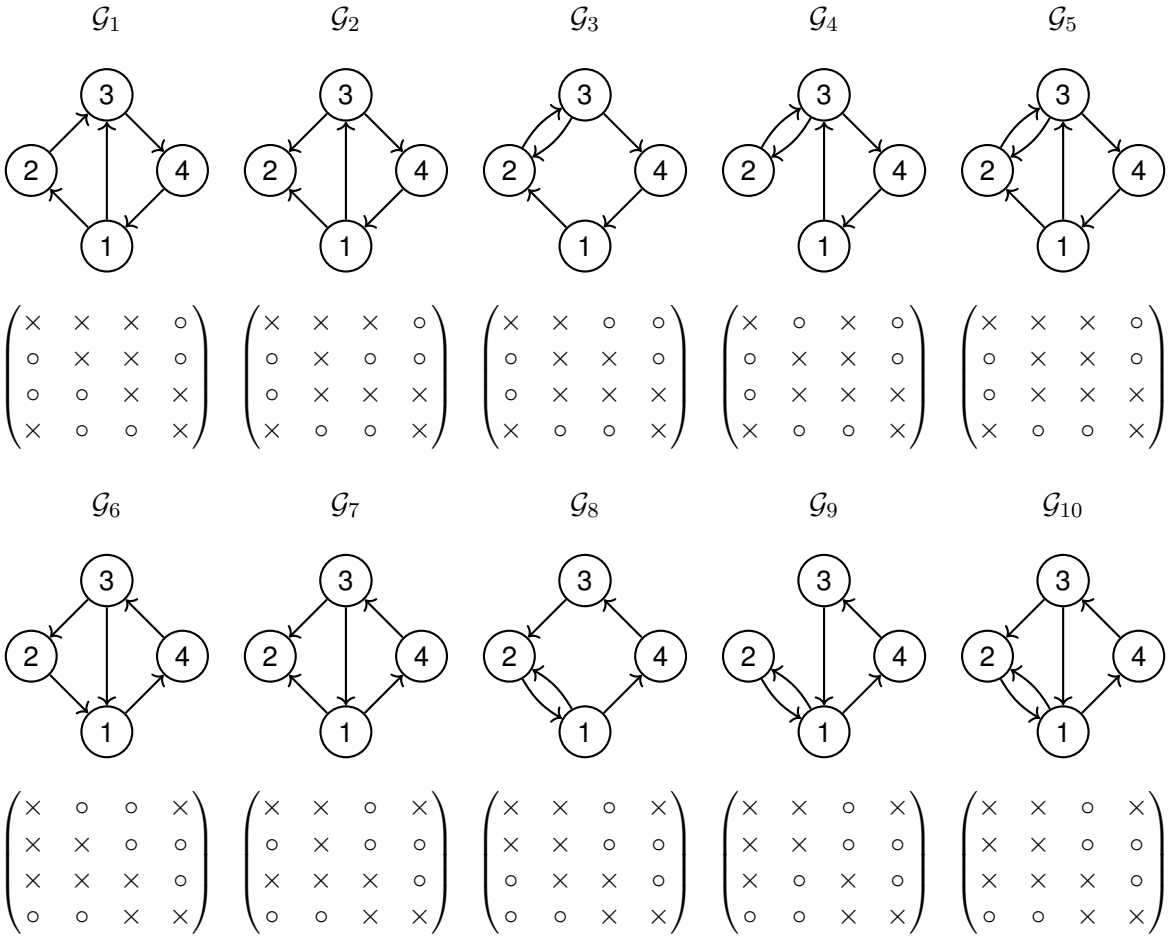


Figure 4.3 All elements of a distribution equivalence class with their corresponding support matrices.

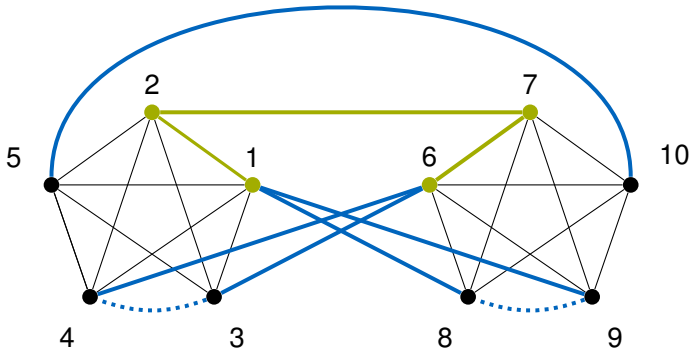
In this example we have 10 elements in the DEC, where 4 of them do not contain a directed cycle of length 2 between a pair of nodes.

Definition 4.7 *A directed graph $\mathcal{G}$ is called* simple *if it does not contain any 2-cycles. Denote the subset of $[\mathcal{G}]_D$ that only contains simple graphs as*

$$[\mathcal{G}]_D^{sim} := \{\mathcal{G}' \in [\mathcal{G}]_D : \mathcal{G}' \textit{ simple }\}.$$

For a simple graph an edge $i \to j$ implies the absence of $i \leftarrow j$. In *Figure 4.3* this is the case for $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_6$, and $\mathcal{G}_7$.
The DGs $\mathcal{G}_1, \ldots, \mathcal{G}_5$ only differ in the triple of nodes $(1, 2, 3)$, where $\mathcal{G}_5$ has one edge more and is a supergraph of the other graphs in the row. Moreover $\mathcal{G}_1, \ldots, \mathcal{G}_4$ can be transformed into one another by a parent exchange on nodes 2 and 3. From $\mathcal{G}_5$ a parent reduction leads to the other graphs in the row. So the support matrices differ only in their second and third column. The same holds for $\mathcal{G}_6, \ldots, \mathcal{G}_{10}$ with $\mathcal{G}_{10}$ being the one graph with an edge more and the parent exchanges are between nodes 1 and 2.
Observe that the graphs in the first row contain cycles $(1, 2, 3, 4)$ and/or $(1, 3, 4)$, which occur in reversed order in the second row, meaning a cycle reversion has to be performed to transform a graph from the first row to the second row, and vice versa.

If we now consider every graph as a node in (another) graph and every support rotation represents an edge in this graph, we can visualize how the elements of a DEC are connected. To not confuse with our directed graphs we will denoted the visualization of a DEC by DEC-plot, edges (operations) in this plot are called DEC-edges and nodes (elements of the DEC) DEC-nodes.



The green DEC-nodes depict simple graphs and green DEC-edges simple rotations. Cycle reversions are drawn in blue and cycle reversions on a 2-cycle as dotted blue lines. The remaining black DEC-edges are parent exchanges/reductions.

In our example for every graph only two vertices are parent exchangeable, leading to this two 'subclasses' in the DEC-plot above, which are connected by cycle reversions. Between the simple graphs $\mathcal{G}_1$ and $\mathcal{G}_6$ there is only one sequence of operations that transforms simple graphs to simple graphs, that is,

$$\mathcal{G}_1 - \mathcal{G}_2 - \mathcal{G}_7 - \mathcal{G}_6.$$

Observe however that it would not be necessary to perform every transformation to get every element in the DEC.

4.6 Simple Graphs

In the previous section we considered a particular DEC and depicted the operations between these graphs in a DEC-plot. We observe that between the simple graphs in this DEC there is a sequence of operations only on simple graphs, and therefore it is naturally to ask whether such a sequence exists for two simple DCGs $\mathcal{G}$ and $\mathcal{G}'$ such that $\mathcal{G} \equiv_D \mathcal{G}'$.

Recall the set of covariance matrices $\mathcal{M}_\mathcal{G}$ (2.14) a structure can generate. For statistical model selection knowing the dimension of $\mathcal{M}_\mathcal{G}$ is crucial. Since $\mathcal{M}_\mathcal{G}$ is defined by the pair of matrices (B, Ω) we expect that the $\dim(\mathcal{M}_\mathcal{G})$ equals the number of unknown parameters $|V| + |E|$, but this might not be true for general graphs. $\mathcal{M}_\mathcal{G}$ is a subset of the space of $V \times V$ matrix and has expected dimension [Drton, 2018]

$$\min\{|V| + |E|, \frac{|V|(|V|+1)}{2}\}. \tag{4.18}$$

For simple graph it is indeed shown by [Amendola et al., 2020, Theorem 3.1] that the expected dimension of $\mathcal{M}_\mathcal{G}$ equals the number of unknown parameters, i.e.

$$\dim(\mathcal{M}_\mathcal{G}) = |V| + |E|. \tag{4.19}$$

This theorem also includes mixed graphs, i.e. graphs with directed an bidirected edges. Then by equation (4.19) it follows that distribution equivalent simple cyclic graphs must have at least the same number of edges. This also follows by the graphical operation presented by [Ghassami et al., 2020], since the only operation that changes the number of edges is a parent reduction, but this can only be applied on non-simple graphs [Ghassami et al., 2020, Proposition 7].

Another interesting result that states a sufficient condition for distribution equivalence of simple graphs is given by [Amendola et al., 2020, Theorem 4.1]: Two simple graphs are distribution equivalent up to closure, if they have the same skeleton and collider triples. A collider triple is a triple of nodes (i, j, k) such that i and j, and j and k are adjacent and j is a collider.

Definition 4.8 *An operation from a simple graph to another simple graph is called a* ***simple operation****. Similarly we call a simple operation of type parent exchange and cycle reversion a simple parent exchange and simple cycle reversion, respectively.*

Notation 4.1 *Let $\mathcal{G}$ be a DCG that contains a cycle $\mathcal{C}$. Denote the length of $\mathcal{C}$ by c. We will write $i \in \mathcal{C}$ to say node i is a element of the cycle $\mathcal{C}$, moreover $i + 1 \in \mathcal{C}$ is the subsequent node of i in $\mathcal{C}$.*

In the following we will examine how the different operations presented in *Section 4.4* affect the creation of a 2-cycle. It is important to mention here that i rather focus on the graphical operations instead of directly considering the support matrix. In some cases the presented proofs might be a bit cumbersome and a more concise result can be achieved by examining the corresponding support matrix and Givens rotations.

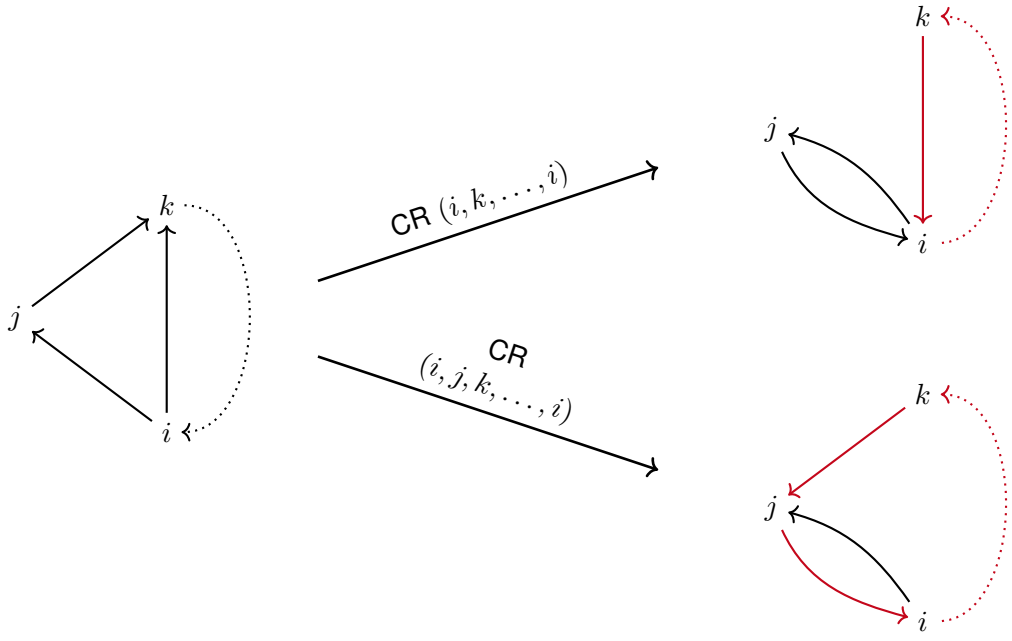


Figure 4.4 Effect of cycle reversions (drawn in red).

Observe how a cycle reversion on a cycle $\mathcal{C}$ in a (simple) DCG $\mathcal{G}$ effects the edges not in $\mathcal{C}$. In *Figure 4.4* we consider the triple of nodes (i, j, k), where there is a directed path from k to i (dotted line), and directed paths $i \to j \to k$ and $i \to k$. So $(i, j, k, \ldots, i)$ and $(i, k, \ldots, i)$ form cycles in $\mathcal{G}$. If we reverse cycle $(i, k, \ldots, i)$, leading to the upper right graph, then there occurs a two cycle $i \leftrightarrows j$. This comes from the fact that with a cycle reversion also every edge pointing to a node in the cycle points to the preceder of this node after the cycle reversion (compare [Ghassami et al., 2020, Proposition 4]). Similarly, reversing $(i, j, k, \ldots, i)$ leads to the 2-cycle $i \leftrightarrows j$. We have the following sufficient condition for a cycle reversion being simple.

Proposition 4.3 *Let $\mathcal{G}$ be a DCG that contains a cycle $\mathcal{C}$. If there is no other cycle $\mathcal{C}'$ in $\mathcal{G}$ such that*

$$|\mathcal{C} \,\Delta\, \mathcal{C}'| = 1, \tag{4.20}$$

then the cycle reversion on $\mathcal{C}$ is simple.

Proof: Let $\tilde{\mathcal{G}}$ be the DCG that results from reversing cycle $\mathcal{C}$ in the simple DCG $\mathcal{G}$, and let $\tilde{\mathcal{C}}$ in $\tilde{\mathcal{G}}$ be the reversed cycle of $\mathcal{C}$.
By [Ghassami et al., 2020, Proposition 4] reversing a cycle $\mathcal{C}$ in $\mathcal{G}$ also changes an edge, which is not in $\mathcal{C}$, and points to a node in $\mathcal{C}$, such that in $\tilde{\mathcal{G}}$ is pointing to the preceder of this node.
Assume the cycle reversion on $\mathcal{C}$ is not simple, so $\tilde{\mathcal{G}}$ contains a 2-cycle. At least one of the two nodes of the 2-cycle must be contained in $\tilde{\mathcal{C}}$ in $\tilde{\mathcal{G}}$, since otherwise transforming $\tilde{\mathcal{G}}$ back to $\mathcal{G}$ by reversing $\tilde{\mathcal{C}}$ results in $\mathcal{G}$ not simple.
Let $i \leftrightarrows j$ be the 2-cycle in $\tilde{\mathcal{G}}$, compare *Figure 4.4*. Note that a cycle in a simple directed graph contains at least 3 nodes.

Case 1: $i, j \in \tilde{\mathcal{C}}$.

Let $\tilde{\mathcal{C}} = (k, j, i, \dots, k)$ be the cycle in $\tilde{\mathcal{G}}$, that results from reversing $\mathcal{C}$ in $\mathcal{G}$. The edge $i \to j$ is not in $\tilde{\mathcal{C}}$ but points to $j \in \tilde{\mathcal{C}}$. The preceder of j in $\tilde{\mathcal{C}}$ is k, hence after reversing $\tilde{\mathcal{C}}$ we get the edge $i \to k$ in $\mathcal{G}$. Moreover reversing $\tilde{\mathcal{C}}$ leads to $\mathcal{C} = (i, j, k, \dots, i)$ in $\mathcal{G}$. Then $\mathcal{C}' = (i, k, \dots, i)$ forms a cycle in $\mathcal{G}$ with $|\mathcal{C} \,\Delta\, \mathcal{C}'| = 1$.

Case 2: $i \in \tilde{\mathcal{C}},\ j \notin \tilde{\mathcal{C}}$.

Wlog. let $\tilde{\mathcal{C}} = (k, i, \dots, k)$ be the cycle in $\tilde{\mathcal{G}}$, that results from reversing $\mathcal{C}$ in $\mathcal{G}$. The edge $j \to i$ is not in $\tilde{\mathcal{C}}$ but points to $i \in \tilde{\mathcal{C}}$, hence will be pointing to k after reversing $\tilde{\mathcal{C}}$. The edge $i \to j$ is not in $\tilde{\mathcal{C}}$ and is not pointing to a node in $\tilde{\mathcal{C}}$, therefore remains unchanged. Moreover reversing $\tilde{\mathcal{C}}$ leads to $\mathcal{C} = (i, k, \dots, i)$ in $\mathcal{G}$. Then $\mathcal{C}' = (i, j, k, \dots, i)$ forms a cycle in $\mathcal{G}$ with $|\mathcal{C} \,\Delta\, \mathcal{C}'| = 1$.

□

This means that if the cycle reversion on $\mathcal{C}$ is not simple, then there must be a cycle $\mathcal{C}'$ in $\mathcal{G}$ such that $|\mathcal{C} \,\Delta\, \mathcal{C}'| = 1$ and we either have $\mathcal{C}' \subset \mathcal{C}$ or $\mathcal{C} \subset \mathcal{C}'$.
The following two statements give necessary conditions such that a cycle reversion is simple.

Proposition 4.4 *Let $\mathcal{G}$ be a DCG with a cycle $\mathcal{C}$. If there is a node $i \in \mathcal{C}$ such that*

$$i + 2 \in \mathbf{Ch}_{\mathcal{G}}(i), \tag{4.21}$$

then the cycle reversion on $\mathcal{C}$ is not simple.

Proof: Let $\mathcal{C} = (i, i+1, i+2, \dots, i)$ in $\mathcal{G}$. Since $i+2 \in \mathbf{Ch}(i)$ there is an edge $i \to i+2$, which is not in $\mathcal{C}$ but points to a node $i+2$ in $\mathcal{C}$. Reversing $\mathcal{C}$ leads to a DCG $\tilde{\mathcal{G}}$ that contains a cycle $\tilde{\mathcal{C}} = (i+2, i+1, i, \dots, i+2)$ and the edge $i \to i+1$, hence $\tilde{\mathcal{G}}$ contains a 2-cycle $i \leftrightarrows i+1$.

□

Proposition 4.5 *Let $\mathcal{G}$ be a DCG with cycle $\mathcal{C}$. If for a node $i \in \mathcal{C}$ there exists a node $j \notin \mathcal{C}$ such that*

$$j \in \mathbf{Pa}(i+1) \quad \textit{and} \quad j \in \mathbf{Ch}(i), \tag{4.22}$$

then the cycle reversion on $\mathcal{C}$ is not simple.

Proof: Let $\mathcal{C} = (i, i+1, \dots, i)$ in $\mathcal{G}$ and j a node in $\mathcal{G}$ but not in $\mathcal{C}$ with $j \in \mathbf{Pa}(i+1)$. Then $j \to i+1$ with $i+1$ in $\mathcal{C}$, therefore we have $j \to i$ after reversing $\mathcal{C}$. The edge $i \to j$ remains unchanged after reversing $\mathcal{C}$ since $j \notin \mathcal{C}$, hence the cycle reversion on $\mathcal{C}$ creates the 2-cycle $i \leftrightarrows j$.

□

Note that in the previous two proposition the assumptions imply that there are cycles $\mathcal{C}$ and $\mathcal{C}'$ such that $|\mathcal{C} \,\Delta\, \mathcal{C}'| = 1$. By *Proposition 4.3* the non-existence of such two cycles is sufficient for a cycle reversion to be simple. The important part in the proof of *Proposition 4.5* is that the node j satisfying equation (4.22) is not in the cycle that gets reversed, meaning the edge $i \to j$, for node i in the cycle, remains unchanged. This is not the case if j is in the cycle itself.

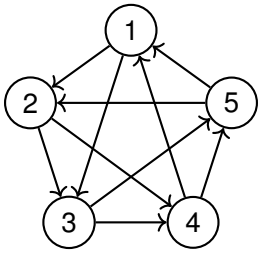


Figure 4.5 A simple DCG.

Consider the simple (and complete) DCG $\mathcal{G}$ in *Figure 4.5* on 5 nodes. $\mathcal{G}$ contains two cycles of length 5, five cycles of length 4, and five cycles of length 3. Only one of them does not fulfill one of the conditions in *Proposition 4.4 or 4.5*, that is, the cycle $\mathcal{C} = (1, 3, 5, 2, 4)$. For every node $i \in \mathcal{C}$ there is a node j such that equation 4.22 is fulfilled. Nevertheless in the DEC of $\mathcal{G}$ there exists exactly one another simple graph $\mathcal{G}'$, obtained by a cycle reversion on $\mathcal{C}$ (compare *Table 5.4: DEC 168.43*).
Let $i = 2$ and $j = 3$, then $i + 1 = 4$. Reversing $\mathcal{C}$ also changes the edge $3 \to 4$ to $3 \to 2$. Still there will be no 2-cycle $2 \leftrightarrows 3$ since 3 is a node of the cycle with $2 \to 3$, so we have $2 \to 1$ after reversing $\mathcal{C}$ since 1 is the preceder of node 3 in $\mathcal{C}$. The same holds for all other nodes in $\mathcal{C}$. Reversing the cycle $\mathcal{C}$ then also reverses every edge not in the cycle, so $\mathcal{G}' = (\mathcal{G})^\top$ which is clearly simple.
Note that this is the only DCG on at most 5 nodes where equation 4.22 is fulfilled for two nodes in a simple DCG but the cycle reversion is still simple. Moreover $\mathcal{C}$ is also the only cycle in $\mathcal{G}$ such that there is no $\mathcal{C}'$ with $|\mathcal{C} \,\Delta\, \mathcal{C}'| = 1$, so the condition in *Proposition 4.3* might also be necessary. Since i am not aware if this is true for every compilation of cycles i leave it as an open question.

Until now we only considered cycle reversions on simple graphs, and sufficient and necessary conditions for a simple cycle reversion. We therefore might ask whether there are operations on a graph $\mathcal{G}$, resulting in a distribution equivalent structure $\mathcal{G}'$, such that the cycle reversion on $\mathcal{C}$ in $\mathcal{G}'$ is simple.

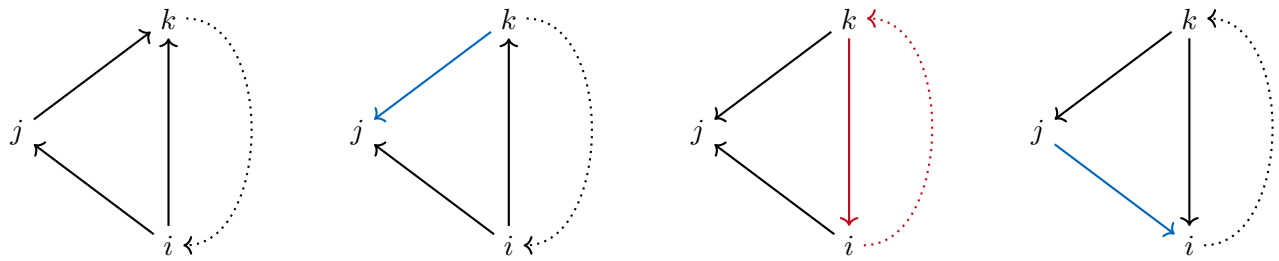


Figure 4.6 Sequence of simple operations.

Consider the triple of nodes (i, j, k) in *Figure 4.6*, denote the graphs from left to right as $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4$. We perform a sequence of simple operations from $\mathcal{G}_1$ to $\mathcal{G}_4$. By *Proposition 4.5 and 4.4* every cycle reversion in $\mathcal{G}_1$ is non-simple. Nevertheless for nodes j and k we have $\mathbf{Pa}_{\mathcal{G}_1}(j) = \{i, j\}$ and $\mathbf{Pa}_{\mathcal{G}_1}(k) = \{i, j, k\}$, respectively. So $|\mathbf{Pa}_{\mathcal{G}_1}(j) \,\Delta\, \mathbf{Pa}_{\mathcal{G}_1}(k)| = 1$, i.e. j and k are parent exchangeable, compare *Definition 4.6*. A simple parent exchange on j and k leads to $\mathcal{G}_2$. Now there is only the cycle $(i, k, \ldots, i)$ left and no edge is pointing towards a node in the cycle. The cycle reversion is simple, leading to $\mathcal{G}_3$. In a last support rotation we can reverse the edge between i and j by a simple parent exchange, resulting in $\mathcal{G}_4$ with both cycles, $(i, j, k, \ldots, i)$ and $(i, k, \ldots, i)$ in $\mathcal{G}_1$, reversed.

So in some cases there exists a sequence of simple operations such that the resulting distribution equivalent graph admits a simple cycle reversion. Here we only have to consider parent exchanges, since a parent reduction can never be simple.

Definition 4.9 *[Ghassami et al., 2020, Definition 7]. A DG $\mathcal{G}$ is **reducible** if there exists $\mathcal{G}'$ such that $\mathcal{G} \equiv_D \mathcal{G}'$ and $E(\mathcal{G}') \subset E(\mathcal{G})$.*

Proposition 4.6 *[Ghassami et al., 2020, Proposition 7]. A DG with no 2-cycles is irreducible.*

This leads to the following result about equivalence of DAGs.

Corollary 4.1 *[Ghassami et al., 2020, Corollary 2]. DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are equivalent if and only if there exists a sequence of parent exchanges that maps $\mathcal{G}_1$ to $\mathcal{G}_2$, and one that maps $\mathcal{G}_2$ to $\mathcal{G}_1$.*

Clearly all parent exchanges between two DAGs are simple. This result can be extended to simple DGs:

Corollary 4.2 *For two simple DGs $\mathcal{G}$ and $\mathcal{G}'$ assume $\mathcal{G} \equiv_D \mathcal{G}'$. There exists a sequence of parent exchanges that maps $\mathcal{G}$ to $\mathcal{G}'$, and one that maps $\mathcal{G}'$ to $\mathcal{G}$ if and only if there exists a sequence of simple parent exchanges that maps $\mathcal{G}$ to $\mathcal{G}'$, and one that maps $\mathcal{G}'$ to $\mathcal{G}$.*

Proof: For simple DGs without a cycle, i.e. DAGs, this is already shown in *Corollary 4.1*. It remains to show for simple DGs that contain a cycle. If there exists a sequence of simple parent exchanges between $\mathcal{G}$ and $\mathcal{G}'$ then there also exists a sequence of not necessarily simple parent exchanges.
Assume there exists a sequence of parent exchanges. For a parent exchange between nodes i and j in a simple DG, i and j must be adjacent. We then either reverse this edge between i and j, or remove a parent from i and j which results in a 2-cycle $i \leftrightarrows j$. This means we only change column i and j of the support matrix, therefore consider the subgraph $\mathcal{G}_{\mathbf{Pa}(i,j)}$ of $\mathcal{G}$ that only involves nodes i and j and their common parents. Note that in a simple DCG every cycle contains at least 3 nodes. Since i or j are never ancestors of their common parents in $\mathcal{G}_{\mathbf{Pa}(i,j)}$ due to simplicity of $\mathcal{G}$, the subgraph $\mathcal{G}_{\mathbf{Pa}(i,j)}$ is acyclic. The result then follows from *Corollary 4.1*.

□

So a sequence of parent exchanges can always be formulated as a sequence of simple parent exchanges. To resolve the problem of an existing sequence of simple operations between simple graphs we therefore only have to consider cycle reversions. The following corollary shows that in the case of *Proposition 4.4* there does not exist a simple cycle reversion on $\mathcal{C}$ for a graph in the DEC.

Corollary 4.3 *Let $\mathcal{G}$ be a simple DCG with a cycle $\mathcal{C}$. Assume there is a node $i \in \mathcal{C}$ such that $i+2 \in \mathbf{Ch}_{\mathcal{G}}(i)$, i.e. equation (4.21) is fulfilled. Then there exists no sequence of simple operations, transforming $\mathcal{G}$ into $\mathcal{G}'$, that is $\mathcal{G} \equiv_D \mathcal{G}'$, such that the cycle reversion on $\mathcal{C}$ in $\mathcal{G}'$ is simple.*

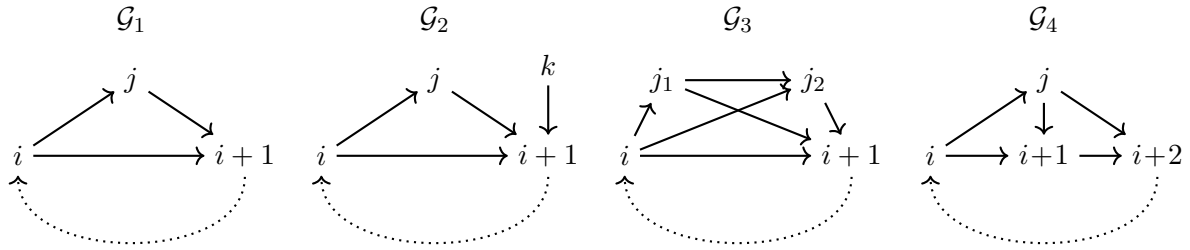
Proof: In the proof of *Proposition 4.4* we have shown that reversing $\mathcal{C} = (i, i+1, i+2, \dots, i)$ results in a 2-cycle $i \leftrightarrows i+1$, since $i \to i+2$ changes to $i \to i+1$. To do a simple cycle reversion on $\mathcal{C}$ in a DCG $\mathcal{G}'$, such that $\mathcal{G} \equiv_D \mathcal{G}'$, we therefore have to prevent the edge $i \to i+2$ from being present in $\mathcal{G}'$. The edge cannot be removed since this would create a 2-cycle, so the only operation left is a simple parent exchange, reversing $i \to i+2$ in $\mathcal{G}$. The node $i+2$ has parent $i+1$, so for a simple parent exchange between i and $i+2$, $i+1$ must also be a parent of i. But since we already have the cycle $\mathcal{C} = (i, i+1, i+2, \dots, i)$ in $\mathcal{G}$, i can not be a parent of $i+1$, as this would contradict the simplicity of $\mathcal{G}$. A simple parent exchange between i and $i+2$ is not possible, hence there is no simple operation that transforms $\mathcal{G}$ into a distribution equivalent $\mathcal{G}'$ such that *Proposition 4.4* does not hold.

□

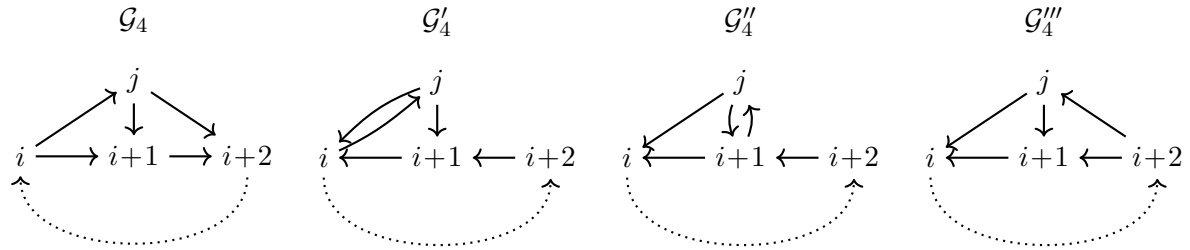
For a DCG $\mathcal{G}$ with a cycle $\mathcal{C}$, such that for a node $i \in \mathcal{C}$ and a node $j \notin \mathcal{C}$ equation (4.22) is fulfilled, this is considerably more involved. Let $\mathcal{G}'$ a DCG distribution equivalent to $\mathcal{G}$ such that cycle $\mathcal{C}$ is reversed in $\mathcal{G}'$. We distinguish the following cases:

1. $\mathcal{G}'$ can not be simple.
2. $\mathcal{G}'$ is simple and there exists a sequence of simple operations between $\mathcal{G}$ and $\mathcal{G}'$.
3. $\mathcal{G}'$ is simple and there does not exist a sequence of simple operations between $\mathcal{G}$ and $\mathcal{G}'$.

Since it was not possible to formulate a concise result regarding the existence of a simple cycle reversion, i will give some examples where each of the above cases occurs.
In every of the following structures there will be a cycle $\mathcal{C} = (i, i+1, i+2, \dots, i)$ and a node $j \notin \mathcal{C}$ such that $j \in \mathbf{Pa}_{\mathcal{G}}(i+1)$ and $j \in \mathbf{Ch}_{\mathcal{G}}(i)$.



Structure $\mathcal{G}_1$ occurs in two of the graphs in the example of a DEC in *Section 4.5*. Since $|P_j \,\Delta\, P_{i+1}| = 1$, we can perform a parent exchange, then only the cycle $\mathcal{C}$ is left and by *Proposition 4.3* the cycle reversion is simple.
In $\mathcal{G}_2$, node $i+1$ has parent k that is not a parent of j. So a parent exchange as in $\mathcal{G}_1$ is not possible. In fact there does not even exist a simple graph such that the cycle $\mathcal{C}$ is reversed. Assume $\mathcal{G}_2'$ results from reversing $\mathcal{C}$ in $\mathcal{G}_2$, then $\mathcal{G}_2'$ contains the 2-cycle $i \leftrightarrows j$. Moreover k is a parent of i in $\mathcal{G}_2'$ but not of j. Then $\{k, i+1\} \subseteq (\mathbf{Pa}_{\mathcal{G}_2'}(i) \,\Delta\, \mathbf{Pa}_{\mathcal{G}_2'}(j))$, so i and j are not parent exchangeable in $\mathcal{G}_2'$. Therefore the 2-cycle $i \leftrightarrows j$ can not be removed in $\mathcal{G}_2'$. The same argumentation holds if k is only a parent of node j.
In $\mathcal{G}_3$ there are j_1, j_2 such that equation (4.22) is fulfilled. However, a sequence of parent exchanges leads to a graph such that the cycle reversion on $\mathcal{C}$ is simple. First, reverse $j_1 \to j_2$, then $j_1 \to i+1$, and finally $j_2 \to i+1$.
In $\mathcal{G}_4$, node j is also a parent of $i+2$. A parent exchange leads to a graph with the same structure but relabeled nodes. So the cycle reversion is non-simple. Nevertheless, there exists a simple graph in the DEC of $\mathcal{G}_4$, by conducting 2 parent exchanges after the cycle reversion:



Until now this is the only known edge configuration where no simple cycle reversion exists, yet there is a simple DG with this cycle reversed. In the atlas of DECs in *Section 5* the graphs with this property are marked with an **X**.

4.7 Algorithm for computing the DEC

In this section the algorithm for, given a directed graph $\mathcal{G}$, computing the DEC $[\mathcal{G}]_D$ is presented. We mainly orient ourselves by the algorithm given by [Ghassami et al., 2020] (see supplementary material section O), and make some small modifications for our purposes.

Algorithm 2 Reduction, Acute Rotation and Cycle Operations [Ghassami et al., 2020]

function REDUCTION(ξ, i, j)
  $\xi' \leftarrow \xi$
  $\xi'_{i,j} \leftarrow 0$
  return ξ'
end function

function ACUTEROTATION(ξ, i, j, k, l)
  $\xi' \leftarrow \xi$
  $\xi'_{i,j} \leftarrow 0$
  $\xi'_{l,j} \leftarrow \xi'_{l,k} \leftarrow 1$
  return ξ'
end function

function FINDROTATIONS(ξ)
  $Rotations \leftarrow \emptyset$
  for j, k in $[p]$ s.t. $\|\xi_{\cdot,j} - \xi_{\cdot,k}\| = 0$ **do** ▷ Find Reductions
    for i in $[p]$ s.t. $\xi_{i,j} = 1$ **do**
      if $i \neq j$ **then**
        $Rotations \leftarrow Rotations \cup \{\text{REDUCTION}(\xi, i, j)\}$
      end if
      if $i \neq k$ **then**
        $Rotations \leftarrow Rotations \cup \{\text{REDUCTION}(\xi, i, k)\}$
      end if
    end for
  end for

  for j, k in $[p]$ s.t. $\|\xi_{\cdot,j} - \xi_{\cdot,k}\| = 1$ **do** ▷ Find Acute Rotations
    $l \leftarrow$ row-index s.t. $\xi_{l,j} \neq \xi_{l,k}$
    for $i \neq l$ s.t. $\xi_{i,j} = 1$ **do**
      if $i \neq j$ **then**
        $Rotations \leftarrow Rotations \cup \{\text{ACUTEROTATION}(\xi, i, j, k, l)\}$
      end if
      if $i \neq k$ **then**
        $Rotations \leftarrow Rotations \cup \{\text{ACUTEROTATION}(\xi, i, k, j, l)\}$
      end if
    end for
  end for
  return $Rotations$
end function

function CYCLEREVERSION(ξ)
  $Reversed \leftarrow \emptyset$
  $\mathcal{C} \leftarrow$ list of cycles in ξ
  for C in $\mathcal{C}$ **do**
    $\xi' \leftarrow \xi$ with cycle C reversed
    $Reversed \leftarrow Reversed \cup \{\xi'\}$
  end for
  return $Reversed$
end function

The functions reduction, acute rotation and cycle reversion implement the types of rotations given in *Section 4.3*. Moreover FINDROTATIONS, given the support matrix ξ, computes the symmetric difference between every pair of columns and then applies a reduction or an acute rotation. The output is a list of support matrices.

Algorithm 3 Compute the distribution equivalence class

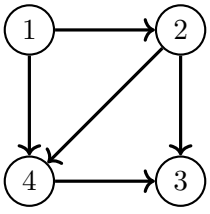
```
1: procedure DEC(ξ*)
2:     Equiv ← {ξ*}
3:     stack ← ∅
4:     stack.push(ξ*)
5:     log ← ∅
6:     while stack not empty do
7:         ξ ← stack.pop()
8:         Rotations ← FINDROTATIONS(ξ)
9:         if |Rotations| = 0 then
10:            next
11:        else
12:            for ξ̃ in Rotations do
13:                if ξ̃ not in Equiv then
14:                    Equiv ← Equiv ∪ {ξ̃}
15:                    stack.push(ξ̃)
16:                    if ξ and ξ̃ are simple then
17:                        log ← (ξ, ξ̃)          ▷ keep track of simple rotations
18:                    end if
19:                end if
20:            end for
21:        end if
22:    end while

23:    stack ← Equiv
24:    while stack not empty do
25:        ξ ← stack.pop()
26:        Reversals ← CYCLEREVERSION(ξ)
27:        if |Reversals| = 0 then
28:            next
29:        else
30:            for ξ̃ in Reversals do
31:                if ξ̃ not in Equiv then
32:                    Equiv ← Equiv ∪ {ξ̃}
33:                    stack.push(ξ̃)
34:                end if
35:                if ξ and ξ̃ are simple then
36:                    log ← (ξ, ξ̃)
37:                end if
38:            end for
39:        end if
40:    end while
41:    return (Equiv, log)
42: end procedure
```

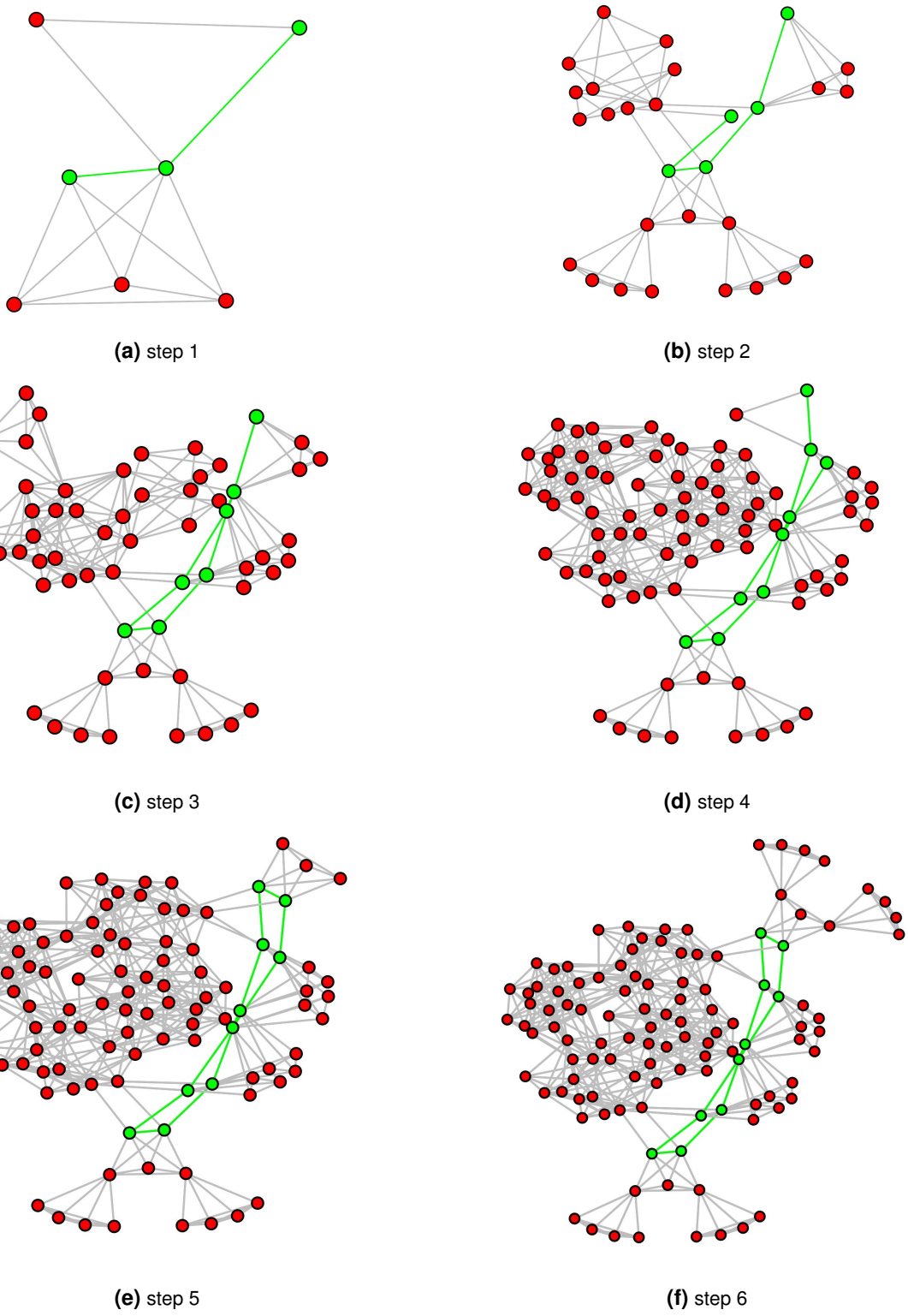
Recall *Theorem 4.1* of distribution equivalent structures. We need to perform a sequence of rotations on the support matrix to find every distribution equivalent structure. So the algorithm by [Ghassami et al., 2020] first conducts every parent exchange on the initial support ξ^*, where every reduction and acute rotation is computed until no new support occurs. In the second step for every support matrix from step one we compute the cycle reversions until no new support is created. This procedure then computes every support matrix for which the corresponding graphs are distribution equivalent.
Since we are also interested whether the rotation is simple we keep track of all rotations in a log-file. Then after the algorithm terminates we consider this log file as an edgelist of an undirected graph an compute the number of connected components. If we only have one connected component then every simple graph can be transformed into every other simple graph in this DEC. In fact the algorithm presented in [Ghassami et al., 2020] would not compute every possible rotation between the elements of the DEC. To see this compare the DEC-plot in *Section 4.5*; the algorithm does not find the black edges in the right part of this DEC-plot, i.e. the parent exchanges and reductions after the cycle reversion. So in practice another step of parent reductions and parent exchanges after the cycle reversion is added to find every possible rotation. In *Section 4.8* this is again visualized by some more DEC-plots, compare *Figure 4.11*.

4.8 Visualization of the DEC

In this section we visualize some DECs and the connections between their elements as in *Section 4.5* for our example DEC. So for a given graph $\mathcal{G}$, we compute the DEC using *Algorithm 3*, while keeping track of all rotations performed.
We start with the following DAG $\mathcal{G}_1$:



The DEC of $\mathcal{G}_1$ consists of 113 DGs (with 10 simple DGs), while there are in total 6 permutations of nodes that have the same structure (see *Table 5.2: DEC 16.1*).
We show the DEC-plot of $[\mathcal{G}_1]_D$, where simple graphs are depicted as green DEC-nodes and graphs that contain 2-cycles as red DEC-nodes. Operations (DEC-edges) between simple graphs are also colored green. Since $\mathcal{G}_1$ is a DAG it is enough to only use parent exchanges and parent reductions to compute the whole DEC [Ghassami et al., 2020, Corollary 2].



(a) step 1 **(b)** step 2

(c) step 3 **(d)** step 4

(e) step 5 **(f)** step 6

Figure 4.9 Trace of *algorithm 3*.

In *Algorithm 3* we apply rotations to the support matrices until no new support is created in a while loop. To get every structure in the DEC of $\mathcal{G}_1$ we need to repeat these steps 6 times. The starting point of the algorithm, i.e. the DAG $\mathcal{G}_1$, is the DEC-node in *Figure 4.11a*, that is connected to every other DEC-node. After step 1 we have 7 graphs in total. Starting from $\mathcal{G}_1$, we can perform a parent exchange between nodes 1 and 2, where there is a simple and a non-simple parent exchange. These lead to the DEC-nodes at the top. Moreover nodes 2 and 3 are also parent exchangeable, with one simple and three non-simple parent

exchanges, since 2 and 3 have the common parent 1.
Observe that there are some sorts of 'clusters' that occur in this DEC-plot, which we have already seen in the example DEC in *Section 4.5*. Interestingly, there are graphs in this DEC that only can be transformed into one another by rotating over a simple graph, so there are disjoint subsets of $[\mathcal{G}_1]_D$, that are separated by the set of simple graphs.

In this DEC there are now DAGs, but also structures with cycles, so we compute the second phase of *Algorithm 3*, that is, all cycle reversions.

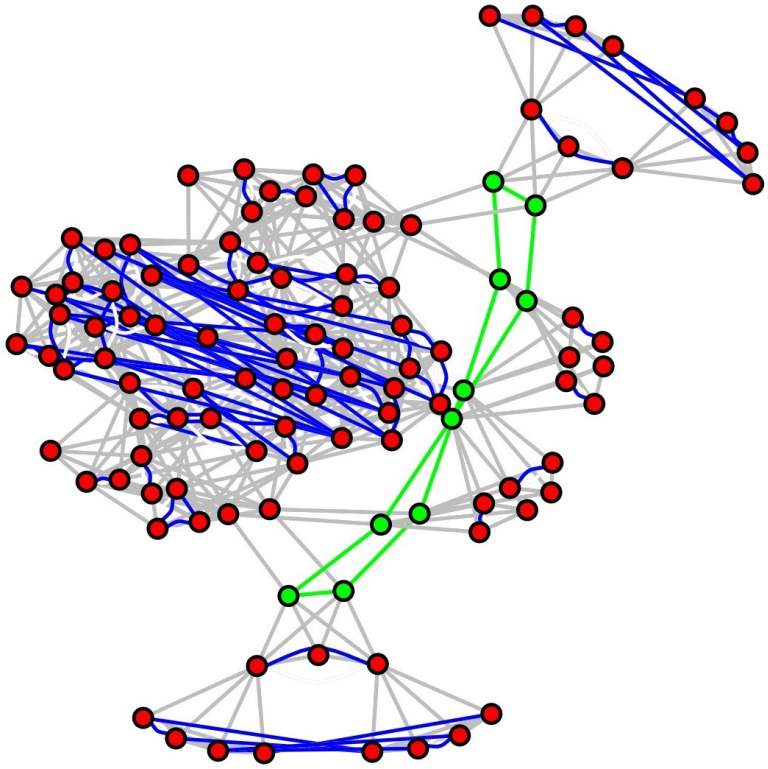
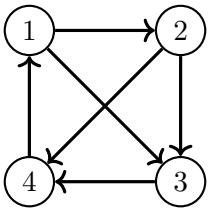


Figure 4.10 Additional cycle reversions after step 6.

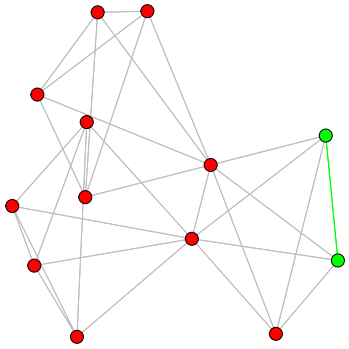
If we now additionally compute all cycle reversions, depicted as blue DEC-edges in *Figure 4.10*, then no graph is added to the DEC, but there are more connections between the DEC-nodes. Moreover some of the 'clusters' are now connected by a cycle reversion.

By [Ghassami et al., 2020, Corollary 2], two DAGs are distribution equivalent if and only if there exists a sequence of parent exchanges that maps one graph to another and vice versa. Considering also cyclic graphs, we have already seen in *Section 4.5* that there are distribution equivalent DGs with different numbers of edges, so a parent reduction is needed. Given our numerical results on up to 5 nodes we might assume that *every* DG in the DEC of a DAG can be found by only applying parent exchanges and parent reductions. This in turn means for *Algorithm 3*, we can check after the first phase if there is a DAG in the DEC and then skip the second phase.

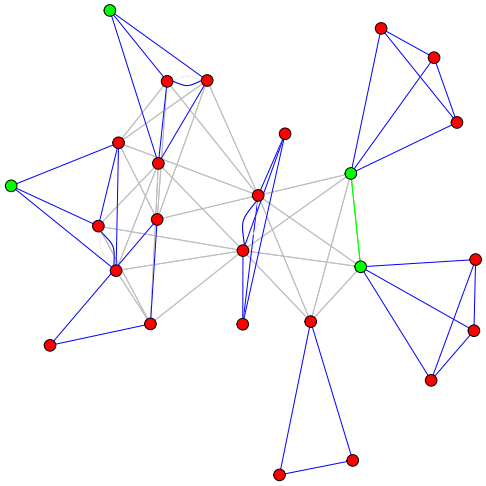
The next DEC we explore contains graphs with directed cycles and simple graphs that can not be transformed into one another by only simple operations. We use the graph $\mathcal{G}_2$ as starting point:



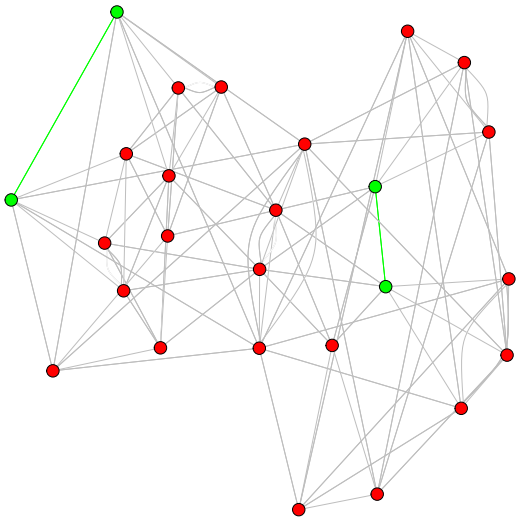
The DEC of $\mathcal{G}_2$ consists of 26 DGs where 4 of them are simple (see *Table 5.2: DEC 22.2*). The simple graphs in this DEC contain the structure we have considered at the very end of *Section 4.6*.
After all parent exchanges/reductions on $\mathcal{G}_2$, we have computed 13 DGs, the cycle reversions then compute the remaining 13 DGs. Again, we added an additional phase of parent exchanges/reductions to show all connections in this DEC.



(a) parent exchanges/reductions



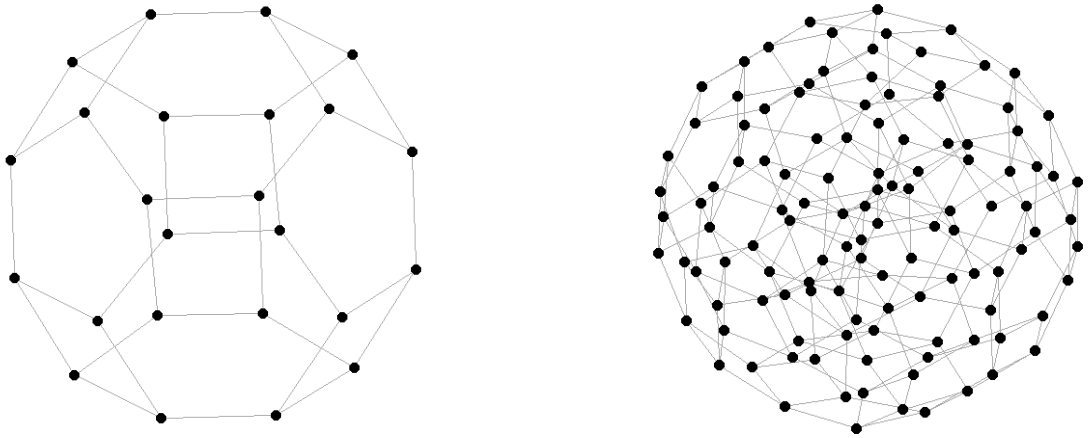
(b) cycle reversions



(c) $[\mathcal{G}_2]_D$ with all connections

Figure 4.11 Phases of *Algorithm 3*.

The last DECs we consider are them of the complete DAGs on 4 and 5 nodes, denote by $\mathcal{G}_4$ and $\mathcal{G}_5$, respectively. Since the whole DEC is far too big to visualize properly (1397 and 202516 elements, respectively), we only focus on the simple graphs in this classes.



(a) $[\mathcal{G}_3]_D^{sim}$.

(b) $[\mathcal{G}_4]_D^{sim}$.

Figure 4.12 DEC-plots for the simple graphs in the DEC of the complete DAG on 4 and 5 nodes.

Clearly every DEC-node has degree $p - 1$. This comes from the fact that we can bring the support of the complete DAG in lower triangular form, then the two neighboring columns have symmetric difference of 1, so these nodes are parent exchangeable. The maximum of all shortest paths between two DEC-nodes equals the number of edges in the complete DAGs, since we can only reverse every edge.

This visualizations might help future research to better understand the nature of a distribution equivalence class, and give more efficient implementations for the computation of it.
In the case of a DEC that contains simple graphs, that can not be transformed into one another by only simple rotations as in $[\mathcal{G}_2]_D$ above, it might be interesting to know how many non-simple rotations we actually need to reach the next simple graph. In this DEC we have also seen that not every rotation would be needed to compute the whole DEC. So in *Figure 4.11b*, the output of the original algorithm by [Ghassami et al., 2020], every DEC-node has degree of at least 2.

5 Atlas for Equivalence Classes

Equipped with the algorithm for computing the distribution equivalence class for a given graph $\mathcal{G}$, we can now compute every DEC for all directed graphs on p nodes. Due to computational issues we focus on DGs with $p \in \{3, 4, 5\}$.
We extend the results in [Andersson et al., 1997], where the Markov equivalence classes for DAGs are computed and consider also cyclic graphs.

For every pair of nodes 1 and 2, we have 4 possible edge configurations, that is $1 \to 2,\ 1 \leftarrow 2,\ 1 \leftrightarrows 2$ or there is no edge between 1 and 2.
On p nodes there are $\binom{p}{p-1}$ nodepairs, leading to the following number of DGs in total:

$$\begin{aligned} p = 3: &\quad 4^3 = 64 \\ p = 4: &\quad 4^6 = 4096 \\ p = 5: &\quad 4^{10} = 1048576 \end{aligned}$$

Algorithm 4 Compute DEC for every directed graph

```
1:  procedure ATLAS(p)
2:      blacklist ← ∅
3:      DG ← directed graphs on p nodes
4:      for G in DG do
5:          if G not in blacklist then
6:              Equiv ← DEC(G)                          ▷ Algorithm 3
7:              for G̃ in Equiv do
8:                  blacklist.push(G̃)
9:              end for
10:         end if
11:     end for
12: end procedure
```

The above algorithm takes p as input and first computes the set of DGs. Then for every DG we compute the DEC, using *Algorithm 3*, and add the elements to a blacklist so that we do not compute the same DEC multiple times. In practice, after step 6 of the above algorithm, we compute various additional information regarding the size of the DEC, the number of simple graphs in the DEC, if there is a simple rotation between every pair of simple graphs. This will be stored in a list and is the output of this algorithm.
We now give a detailed description of the equivalence classes. Remember that distribution equivalence entails Markov equivalence, so if two structures are not Markov equivalent, this implies that they are not distribution equivalent [Lacerda et al., 2012]. This in turn means that different DECs can only occur within a MEC, so after computing every DEC we also check whether two DECs are Markov equivalent. This allows us to organize the DECs by their corresponding MEC, where we use the partial ancestral graph as a representation for the MEC. Since until now their is no unique representation of a DEC, we provide one graph for every DEC in the cases where the MEC and DEC do not coincide. This only occurs if there are cycles included since in the acyclic case we know that Markov equivalence and distribution equivalence are equivalent [Ghassami et al., 2020, Proposition 1].

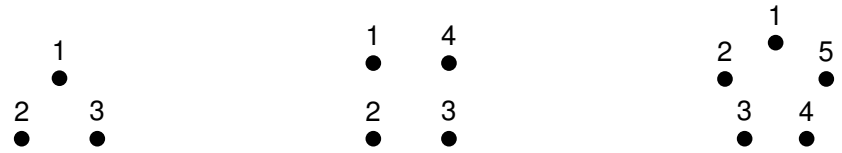


Figure 5.1 Labeling of the nodes.

The atlas are mainly constructed as in [Andersson et al., 1997]. In the first column we depict the representative of the unlabeled MECs, the PAG, computed with the CCD algorithm [Richardson, 1996b]. The second column shows the conditional independence constraints for one MEC, for which we label the nodes as in *Figure 5.1*. The third column gives the total number of labeled MECs that are represented by this PAG; and in the fourth column the total number of DGs among the labeled MECs. The remaining three columns correspond to the DECs, where in the fifth column we depict a DG if the MEC and DEC do not coincide. This is not optimal since there may be a lot of structures in the DEC and a single graph is not as expressive as the PAG for the MEC. The sixth and seventh column then give the number of labeled DECs and the number of DGs, respectively. (In the case of $p = 4$ and $p = 5$ we also add an index so that we can refer to a specific class.)

We start with the atlas on $p = 3$ nodes. The set of DGs consists of 64 structures in total, which are divided into 11 MECs and described by 5 unlabeled MECs. Here we have the empty graph, the structures with an isolated node and a single edge, the unshielded collider, the unshielded non-collider, and the complete graph. For the complete graph we have already shown that the complete DAG and the directed cycle on 3 nodes are not distribution equivalent. So in total there are 12 DECs and 6 unlabeled DECs.

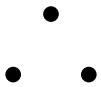
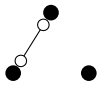
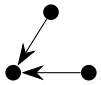
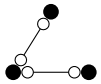
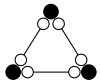
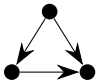
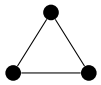
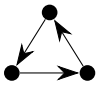
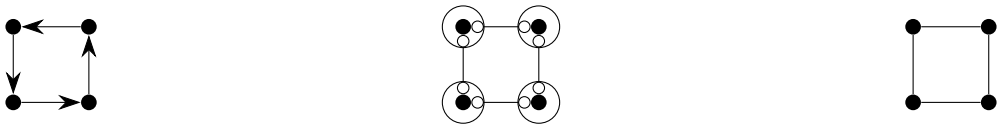
PAG	MP	#MEC	#DG	G	#DEC	#DG
	$1 \perp\!\!\!\perp 2 \perp\!\!\!\perp 3$	1	1		1	1
	$\{1,2\} \perp\!\!\!\perp 3$	3	9		3	9
	$1 \perp\!\!\!\perp 3$	3	3		3	3
	$1 \perp\!\!\!\perp 3 \mid 2$	3	15		3	15
	(None)	1	36		1	34
	(None)				1	2
Total:		11	64		12	64

Table 5.1 Atlas: $p = 3$

Note that the CCD algorithm only performs orientation rules on unshielded triples (or quadtruples where at least two nodes are not adjacent). So in the MEC containing the DEC of the complete DAG and of the simple directed cycle, after the first step the algorithm is done. In the case of 4 nodes the same occurs in the MECs where there is a DEC of a DCG and a DEC of a DAG. Especially when there is a complete subgraph on 3 nodes in a DG where none of the nodes of the subgraph is an unshielded collider, then by *Theorem 3.1* we can reverse every edge. Thus in this MEC there is a DEC that contains a graph where this subgraph is acyclic and there is a DEC with a graph where this subgraph is the directed cycle.
In [Richardson, 1996a] it is already mentioned that the CCD algorithm does not always give the most informative PAG for a given graph $\mathcal{G}$, in that there may be features which are not captured by the PAG that the CCD algorithm outputs, so it is not *complete* in this sense. Nevertheless it is *d-separation complete*, so if the d-separation oracle for two graphs causes the algorithm to produce the same PAGs, then they are also d-separation equivalent.
Therefore we added all possible ancestral relations that can be made. This brings some sort of redundancy in the representation of the equivalence class.

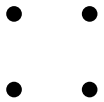
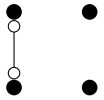
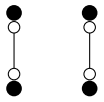
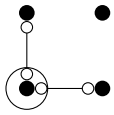
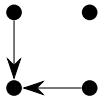
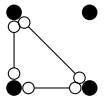
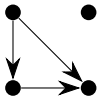
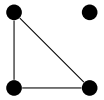
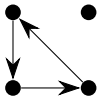
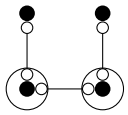
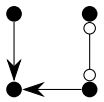
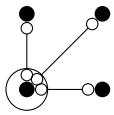
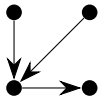
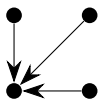


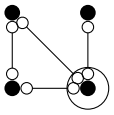
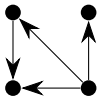
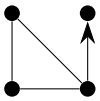
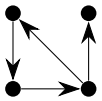
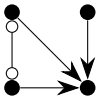
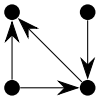
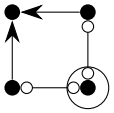
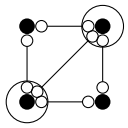
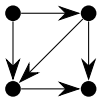
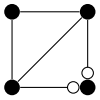
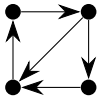
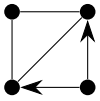
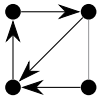
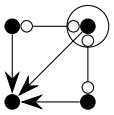
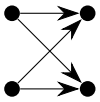
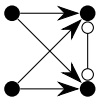
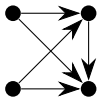
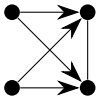
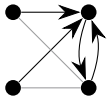
Consider the directed cycle on 4 nodes on the left hand side. The graph in the middle is the output of the CCD algorithm, where every node is circled (underlining in *Definition 3.8*), meaning that for every unshielded triple (i, j, k), j is either an ancestor of i or k, so this encodes the presence of an unshielded conductor. The edges are not oriented. It is clear that the PAG only represents the directed cycle (there is also no DAG with the same conditional independence statements), but the PAG on the right hand side also represents only the directed cycle. The tails at every node mean ancestorship of the adjacent nodes. This comes from the fact that only the first condition in *Definition 3.8* is sufficient, all other conditions are merely necessary [Richardson, 1996a].

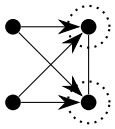
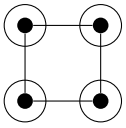
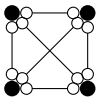
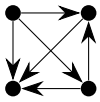
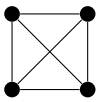
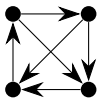
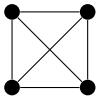
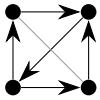
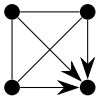
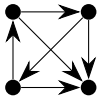
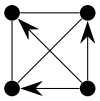
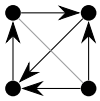
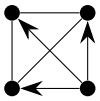
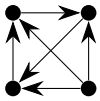
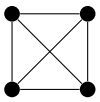
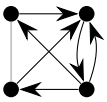
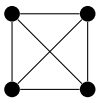
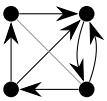
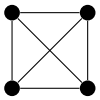
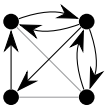
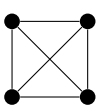
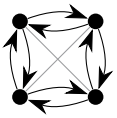
We will see that the PAG is not equal for every markov equivalent DEC, it can therefore show features common to every element of a DEC. In some cases the PAG can even uniquely determine a specific DEC; as we have already seen for the directed cycle above.

Consider now the set of DGs on $p = 4$ nodes. We have in total 4096 DGs, which are divided into 194 MECs and 22 unlabeled MECs. If we consider the set of DAGs there are 185 MECs and 20 unlabeled MECs [Andersson et al., 1997]. So there are 2 structures with conditional independence constraints that can not be represented by a DAG, that are the directed cycle above and the DCG we also have seen in previous examples (*Figure 2.1*, compare MEC 20 & 21 in *Table 5.2*. Moreover there are in total 327 DECs that are described by 40 unlabeled DECs.

To connect to our previous findings we add an **X** to the index if there are two simple graphs that can not be transformed into one another only by simple operations. For every DEC that contains a simple DCG, we add the total number of simple graphs in brackets, where (S) denotes that every graph in this DEC is simple. This occurs if there is no possible parent exchange/reduction and *Proposition 4.3* is fulfilled. If there is a simple graph in a DEC, then this graph is depicted, meaning if there is a non-simple graph as example graph, one knows that this class only consists of non-simple graphs. Virtual edges (compare *Definition 2.5*) are depicted as gray undirected edges.
For easier reference we enumerate every MEC and every DEC in a specific MEC. For example, MEC 22 below has no conditional independence constraints, and there are 22.1 - 22.14 different DECs in this MEC. The number of unlabeled MECs and total number of DGs is then given in the first row of every MEC.

index	PAG	MP	#MEC	#DG	G	#DEC	#DG
1		$1 \perp\!\!\!\perp 2 \perp\!\!\!\perp 3 \perp\!\!\!\perp 4$	1	1		1	1
2		$\{1,2\} \perp\!\!\!\perp 3 \perp\!\!\!\perp 4$	6	18		6	18
3		$\{1,2\} \perp\!\!\!\perp \{3,4\}$	3	27		3	27
4		$1 \perp\!\!\!\perp 3 \mid 2$ $\{1,2,3\} \perp\!\!\!\perp 4$	12	60		12	60
5		$1 \perp\!\!\!\perp 3$ $\{1,2,3\} \perp\!\!\!\perp 4$	12	12		12	12
6.1		$\{1,2,3\} \perp\!\!\!\perp 4$	4	144		4	136
6.2						4	8 (S)
7		$1 \perp\!\!\!\perp 3 \mid 2$ $\{1,2\} \perp\!\!\!\perp 4 \mid 3$	12	84		12	84
8		$1 \perp\!\!\!\perp \{3,4\}$ $2 \perp\!\!\!\perp 4 \mid 3$	24	72		24	72
9		$1 \perp\!\!\!\perp \{3,4\} \mid 2$ $\{1,4\} \perp\!\!\!\perp 3 \mid 2$	4	28		4	28
10		$1 \perp\!\!\!\perp 4$ $\{1,4\} \perp\!\!\!\perp 3 \mid 2$	12	12		12	12
11		$1 \perp\!\!\!\perp 3 \perp\!\!\!\perp 4$	4	4		4	4

12.1		$\{1,2\} \perp\!\!\!\perp 4 \mid 3$	12	552		12	528
12.2						12	24 (S)
13		$\{1,2\} \perp\!\!\!\perp 4$	12	36		12	36
14		$2 \perp\!\!\!\perp 4$ $1 \perp\!\!\!\perp 4 \mid \{2,3\}$	24	24		24	24
15		$2 \perp\!\!\!\perp 4 \mid 3$ $1 \perp\!\!\!\perp 3 \mid \{2,4\}$	12	60		12	60
16.1		$1 \perp\!\!\!\perp 3 \mid \{2,4\}$	6	822		6	678
16.2						12	120 (48)
16.3						12	24 (S)
17		$1 \perp\!\!\!\perp 3 \mid 4$	12	60		12	60
18		$1 \perp\!\!\!\perp 2$ $3 \perp\!\!\!\perp 4 \mid \{1,2\}$	6	6		6	6
19.1		$1 \perp\!\!\!\perp 2$	6	54		6	42
19.2						6	12

20		$1 \perp\!\!\!\perp 2$ $1 \perp\!\!\!\perp 2 \mid \{3,4\}$	6	12		6	12
21		$1 \perp\!\!\!\perp 4 \mid \{2,3\}$ $2 \perp\!\!\!\perp 3 \mid \{1,4\}$	3	6		3	6
22.1		(None)	1	2002		1	1397
22.2 X						6	156 (24)
22.3						4	12 (S)
22.4						4	8 (S)
22.5						12	24 (S)
22.6						4	8 (S)
22.7						4	16
22.8						12	48
22.9						4	180
22.10						1	9

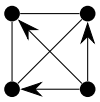
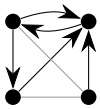
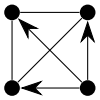
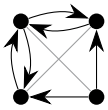
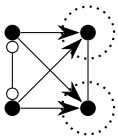
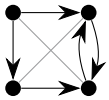
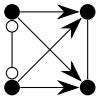
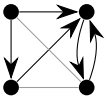
22.11				12	36
22.12				12	36
22.13				6	36
22.14				6	36
Total:		194	4096	327	4096

Table 5.2 Atlas: $p = 4$

We now review some of the intrinsic structures that lead to different distribution equivalence classes. This might help further research in this area since there are a lot of open questions, in particular, is there are supergraph representing a unique DEC, as the essential graph represents the MEC for DAGs and the PAG for MAGs. Moreover there might be graphical conditions, besides rotating the support matrix, that lead to a easier applicable characterization of distribution equivalence.

We focus on the MECs that contain several DECs, since those are mainly the classes where cycles occur. Moreover the focus lies more on DECs that involve at least one simple graph.

Recall the labeling of the nodes in *Figure 5.1*.
Consider MEC 12, which consists of 2 DECs. The triples $(1, 3, 4)$ and $(2, 3, 4)$ are unshielded conductors, depicted by a circle around 3. Moreover there are neither unshielded non-conductors, nor other long-range d-separations. DEC 12.1 contains all DAGs that have the conditional independence $\{1, 2\} \perp\!\!\!\perp 4 \mid 3$. Clearly there are not only DAGs in this class, but also graphs with 2-cycle. In the classes that contains DAGs there are no DCGs with a simple directed cycle. The DCGs with the given CI constraint $\{1, 2\} \perp\!\!\!\perp 4 \mid 3$ are contained in DEC 12.2. This class has a directed cycle on $(1, 2, 3)$, meaning there is always an arrowhead pointing towards 3. But then the edge between 3 and 4 has to be directed towards 4, so that we fulfill the properties of the cyclic equivalence theorem (*Theorem 3.2*, i.e. Markov equivalent DCGs need to have the same unshielded conductors). This in turn means that in every DG in DEC 12.2, the edge between 3 and 4 has to be directed as $3 \to 4$. Call such a node a **sink of the cycle**. The DCGs in DEC 12.2 contain only one cycle, so by *Proposition 4.3* the cycle reversion is simple.
If the edge is directed as $4 \to 3$, then by *Definition 2.5* of adjacent vertices there is a virtual adjacency between 4 and the predecessor of 3, compare also [Richardson, 1996c, Section 3.6] and [Ghassami et al., 2020, Proposition 4] about the direction of cycles. The corresponding DEC is then 16.3, where in the PAG we have undirected edges that encodes the directed cycle, and a node, 3 in this case, that is an ancestor of 2 and 4. Clearly 3 is also an ancestor of 1 in every DG in this DEC, but we have the CI constraint $1 \perp\!\!\!\perp 3 \mid \{2, 4\}$. Call such a node a **source of a cycle**. This structure also reveals a weakness of PAGs; they do not represent if two vertices are really or virtually adjacent like the essential arrows in essential graphs. In DEC 12.2 we have already seen that there is an essential arrow $3 \to 4$, which is not essential

in the DEC that contains the DAGs.
MEC 16 consists of two other DECs; DEC 16.1 contains all DAGs with the given CI constraint (compare DEC-plot in *Figure 4.9*). For DEC 16.2 the PAG encodes that there must be a directed cycle, and the nodes 2 and 4 are ancestors of 3 in every graph in the DEC, and 3 is an ancestor of 2 and 4 in some but not all graphs in the DEC. 3 is then neither a source node nor a sink node of the cycle. The example graph of DEC 16.2 contains two cycles where each of them fulfills one of the conditions in *Proposition 4.4* and *Proposition 4.5*, so not every graph in this DEC is simple. Nevertheless by reversing $3 \to 2$, the cycle reversion on $(1, 4, 2)$ will be simple by *Proposition 4.3*. This DEC we have already studied in detail in *Section 4.5* and depicted the support rotations by a DEC-plot.
MEC 19 comprises two DECs where DEC 19.1 involves the DAGs with the CI constraint $1 \perp\!\!\!\perp 2$. DEC 19.2 consists of two non-simple DCGs, where there is a 2-cycle $3 \leftrightarrows 4$. Note that in the example graph in DEC 19.1 we can do a non-simple parent exchange, removing on parent of 3 and 4, resulting in a 2-cycle $3 \leftrightarrows 4$. If we compare DEC 19.2 and DEC 20 then in the PAG of DEC 20 the nodes 3 and 4 are circled by dots. This characterizes unshielded imperfect non-conductors, so node 4 for example is not a descendant of a common child of 1 and 2 in every graph in the MEC. This is clearly not the case for DEC 19.2, where node 4 is a unshielded perfect non-conductor (compare *Theorem 3.2*).
MEC 20 represents the graphs in *Figure 3.1*.
MEC 21 contains the 4-cycles. Every node is an ancestor of the adjacent nodes in every graph in the MEC, so every node is also an unshielded conductor.
Now consider MEC 22 where there is no CI constraint between any of the variables. This MEC contains 14 DECs, where 6 of them involve at least one simple graph. Again there is a DEC (22.1) with the (complete) DAGs. The PAGs for DEC 22.2 and 22.3 both encode that every node is an ancestor of every other node. DEC 22.2 is a special case compared to all other DECs on 4 nodes in the sense that there exists two simple graphs that cannot be transformed to one another by only simple rotations (compare DEC-plot in *Figure 4.11*).
This is not the case for DEC 22.3 where we have two directed cycles, $\mathcal{C}$ and $\mathcal{C}'$, of length 3. But since $|\mathcal{C} \,\Delta\, \mathcal{C}'| = 2$, by *Proposition 4.3* every cycle reversion is simple, so this DEC consists of only simple graphs.
For DEC 22.4 the PAG encodes that there is a directed cycle on 3 nodes and a sink node of the cycle. Since the edges from a node of a cycle to a sink node of the cycle are all really adjacent, the PAG uniquely determines this DEC. This is not the case for a source node of a cycle, consider DECs 22.5 and 22.6, which both have node 3 as a source. Only in DEC 22.6 the edges of the source node are really adjacent. Since the PAG does not differentiate between real and virtual adjacencies, it coincides for these two classes.

We now consider the set of directed graphs on 5 nodes, where there are in total $1.048.576$ directed graphs from which 29.281 DGs are DAGs [Andersson et al., 1997] and 59049 DGs are simple graphs.
We use again *Algorithm 4* to compute every distribution equivalence class, where we get a total of 40.602 DECs with 683 different permutations of nodes on this classes.
The size of the classes is distributed as follows (in boldface the size of the DEC $|[G]_D|$ and below the number of DECs that have this size):

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
2616	4777	5860	4335	3755	3880	1710	1320	1740	2070	240	600	122	600
15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**25**	**26**	**27**	**28**	**30**	**34**
460	75	120	105	130	710	380	150	270	300	90	75	340	150
35	**36**	**38**	**40**	**42**	**44**	**45**	**48**	**49**	**50**	**51**	**53**	**54**	**55**
180	105	60	30	210	571	180	120	60	150	60	60	210	50
57	**60**	**63**	**65**	**68**	**69**	**70**	**76**	**84**	**89**	**90**	**93**	**97**	**100**
60	10	50	60	160	80	120	35	60	10	60	30	60	15
102	**108**	**113**	**125**	**130**	**131**	**139**	**161**	**200**	**223**	**240**	**264**	**279**	**315**
10	10	150	15	30	60	60	30	30	10	10	60	10	15
377	**402**	**575**	**666**	**1142**	**1397**	**1597**	**1689**	**2656**	**3027**	**9126**	**202516**		
60	5	60	15	10	5	20	30	30	10	10	1		

Observe that a lot of the DECs occur 30 or 60 times. This arises from the fact that in a graph with 5 nodes we often have a complete subgraph on 3 nodes and then connect the over two nodes to either one node of this subgraph or two different nodes:

So we have $\binom{5}{3} \cdot 3 \cdot 2 = 60$ and $\binom{5}{3} \cdot 3 = 30$ possible configurations, respectively.
As in the case of 4 nodes we are rather interested in the intrinsic structure of the graphs, therefore we omit the labeling of the nodes.
Since every graph on 4 nodes is also a subgraph of a graph on 5 nodes, where the additional node has degree 0, we omit these graphs. The PAG on this 4 nodes is then the same as in *Table 5.2*. We therefore keep the order for the classes **1 - 22.14**. For this 22 Markov equivalence classes the Markov properties stay the same, only $\{1, 2, 3, 4\} \perp\!\!\!\perp 5$ is added which we also leave out until class **22.14**. The size of the equivalence classes also remain equal compared to the graphs on 4 nodes, but the number of unlabeled equivalence classes changes.

The rest of the unlabeled classes is structured as follows: We take the skeleton of the PAG as ordering, since ancestral graphs with the same adjacencies share similar Markov properties [Richardson and Spirtes, 2002] [Ali et al., 2009].
Then for every MEC with more than one DEC we first depict the classes with DAGs, which are followed by simple graphs with cycles and then classes that only contain graphs with 2-cycles. If a class contains a simple graph then this graph is depicted. Again we complete the PAG outputted from the CCD algorithm to depict the most informative representative.

We give a short overview of the different skeletons of the PAG. So given a DG and one wants to find the different DECs that share the same conditional independence statements, only the first step of the CCD algorithm has to be computed since this already determines the skeleton. Steps 2-6 in the CCD are then only ancestral relationships and orientation rules. Said differently, we categorize the classes by their conditional independence graph, that is an undirected graph where two nodes i and j are not adjacent if and only if there is a subset of nodes $S \subset V \setminus \{i, j\}$ such that $i \perp\!\!\!\perp j \mid S$. In total we then have 22 different CIGs (leaving out classes 1-22 from the atlas on 4 nodes). Note that two graphs that do not have the same CIG also have different conditional independence constraints and are not Markov equivalent, hence not distribution equivalent [Lacerda et al., 2012].

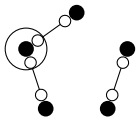
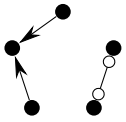
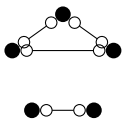
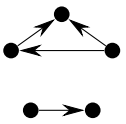
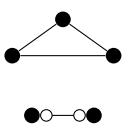
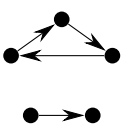
The undirected edges in a CIG are not to be confused with an undirected edge in an ancestral graph, where it means that two nodes are an ancestor of one another.

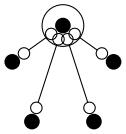
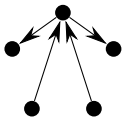
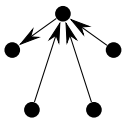
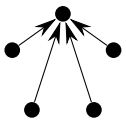
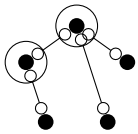
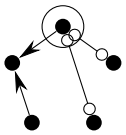
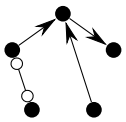
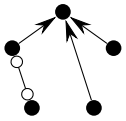
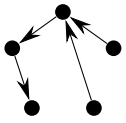
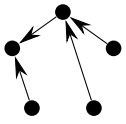
Note that we have in total 683 different DECs, but due to restrictions on the length of this work we simply cannot depict every class. For that reason we leave out the DECs that only contain non-simple graphs for MECs with more than 20 different DECs and rather focus on the DECs that involve at least one simple graph.

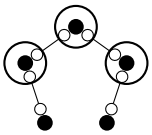
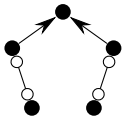
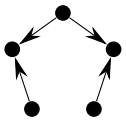
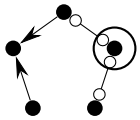
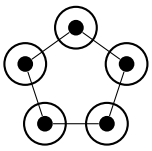
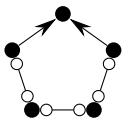
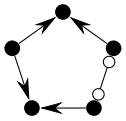
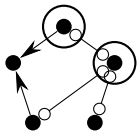
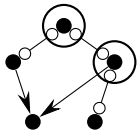
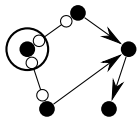
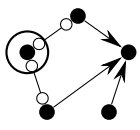
Consider the CIG of section **XIX**; there is a MEC (139) with CI constraints $1 \perp\!\!\!\perp \{3,4,5\} \mid 2$. This is the only MEC where we will deviate a bit from the depiction of our DECs, since in this case we will get different DECs dependent on where node 1 is adjacent to the remaining nodes.

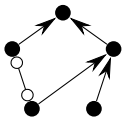
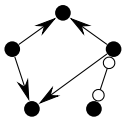
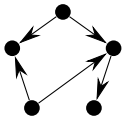
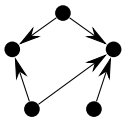
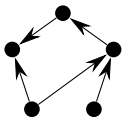
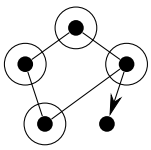
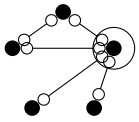
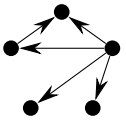
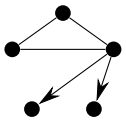
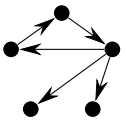
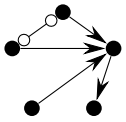
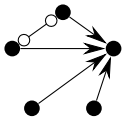
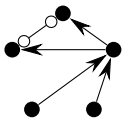
Recall the labeling of the nodes in *Figure 5.1* for the CI constraints. The atlas for DGs on $p = 5$ is shown in the following table:

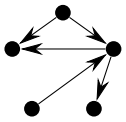
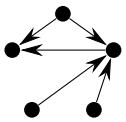
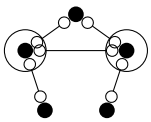
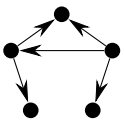
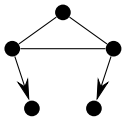
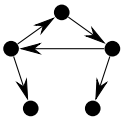
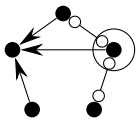
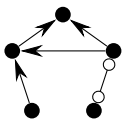
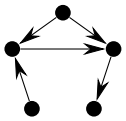
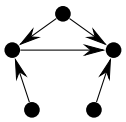
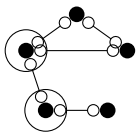
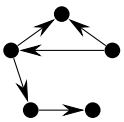
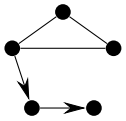
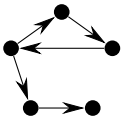
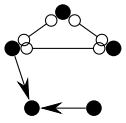
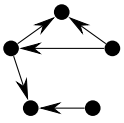
index	PAG	MP	#MEC	#DG	G	#DEC	#DG
1			1	1		1	1
2			10	30		10	30
3			15	135		15	135
4			30	150		30	150
5			30	30		30	30
6.1			10	360		10	340
6.2						10	20
7			60	420		60	420
8			120	360		120	360
9			20	140		20	140
10			60	60		60	60
11			20	20		20	20
12.1			60	2760		60	2640
12.2						60	120
13			60	180		60	180
14			120	120		120	120
15			60	300		60	300
16.1			30	4110		30	3390
16.2						60	600
16.3						60	120
17			60	300		60	300
18.1			30	270		30	210

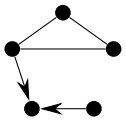
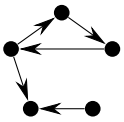
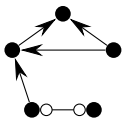
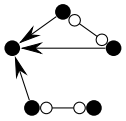
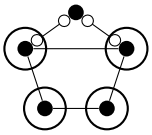
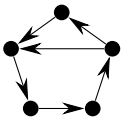
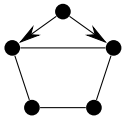
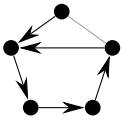
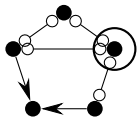
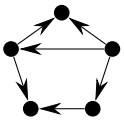
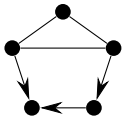
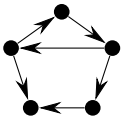
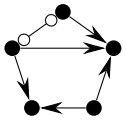
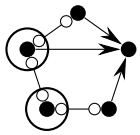
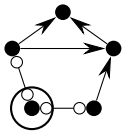
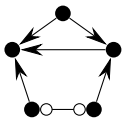
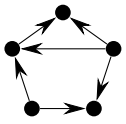
18.2						30	60
19			30	30		30	30
20			30	60		30	60
21			15	30		15	30
22.1			5	10010		5	6985
22.2						30	780
22.3						20	60
22.4						20	40
22.5						60	120
22.6						20	40
22.7						20	80
22.8						60	240
22.9						20	900
22.10						5	45
22.11						60	180
22.12						60	180
22.13						30	180
22.14						30	180
23		$1 \perp\!\!\!\perp 3 \mid 2$ $\{1,2,3\} \perp\!\!\!\perp \{4,5\}$	30	450		30	450
24		$1 \perp\!\!\!\perp 3$ $\{1,2,3\} \perp\!\!\!\perp \{4,5\}$	30	90		30	90
		Total **I** :	60	540		60	540
25.1		$\{1,2,5\} \perp\!\!\!\perp \{3,4\}$	10	1080		10	1020
25.2						10	60 (40)

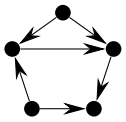
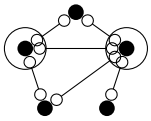
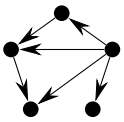
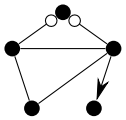
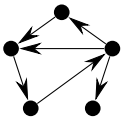
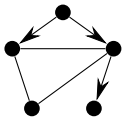
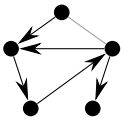
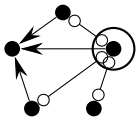
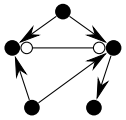
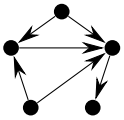
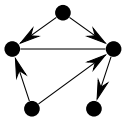
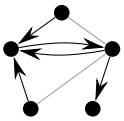
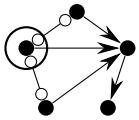
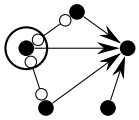
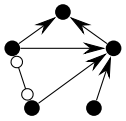
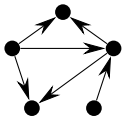
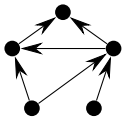
		Total **II** :	10	1080	20	1080
26		$2 \perp\!\!\!\perp \{3,4,5\} \mid 1$ $3 \perp\!\!\!\perp \{4,5\} \mid 1$ $4 \perp\!\!\!\perp 5 \mid 1$	5	45	5	45
27		$2 \perp\!\!\!\perp \{3,4,5\} \mid 1$ $\{3,4\} \perp\!\!\!\perp 5 \mid 1$ $3 \perp\!\!\!\perp 4$	30	30	30	30
28		$2 \perp\!\!\!\perp \{3,4,5\} \mid 1$ $3 \perp\!\!\!\perp 4 \perp\!\!\!\perp 5$	20	20	20	20
29		$2 \perp\!\!\!\perp 3 \perp\!\!\!\perp 4 \perp\!\!\!\perp 5$	5	5	5	5
		Total **III** :	60	100	60	100
30		$\{2,3\} \perp\!\!\!\perp \{4,5\} \mid 1$ $1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp 5 \mid 1$	60	540	60	540
31		$\{2,3\} \perp\!\!\!\perp \{4,5\} \mid 1$ $1 \perp\!\!\!\perp 3$ $4 \perp\!\!\!\perp 5 \mid 1$	60	300	60	300
32		$\{2,3\} \perp\!\!\!\perp 5 \mid 1$ $\{2,3\} \perp\!\!\!\perp 4$ $1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp 5 \mid 1$	120	360	120	360
33		$\{2,3\} \perp\!\!\!\perp \{4,5\}$ $1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp 5$	60	180	60	180
34		$\{2,3\} \perp\!\!\!\perp \{4,5\} \mid 1$ $1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp 5$	60	60	60	60
35		$\{2,3\} \perp\!\!\!\perp \{4,5\} \mid 1$ $1 \perp\!\!\!\perp 3$ $4 \perp\!\!\!\perp 5$	60	60	60	60
		Total **IV** :	420	1500	420	1500

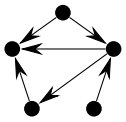
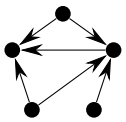
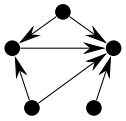
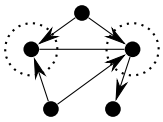
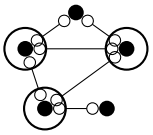
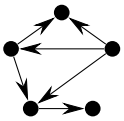
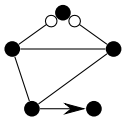
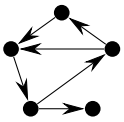
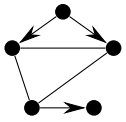
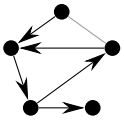
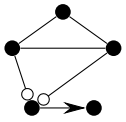
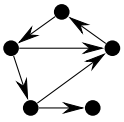
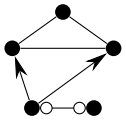
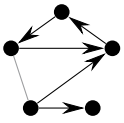
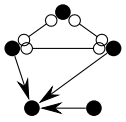
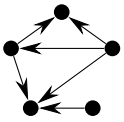
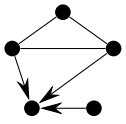
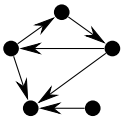
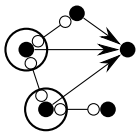
36		$\{2,3\} \perp\!\!\!\perp \{4,5\} \mid 1$ $1 \perp\!\!\!\perp 3 \mid 2$ $1 \perp\!\!\!\perp 4 \mid 5$	60	540	60	540
37		$\{2,3\} \perp\!\!\!\perp \{4,5\}$ $1 \perp\!\!\!\perp 3 \mid 2$ $1 \perp\!\!\!\perp 4 \mid 5$	60	540	60	540
38		$\{2,3\} \perp\!\!\!\perp \{4,5\} \mid 1$ $1 \perp\!\!\!\perp \{3,4\}$	60	60	60	60
39		$\{2,3\} \perp\!\!\!\perp \{4,5\} \mid 1$ $1 \perp\!\!\!\perp 3$ $1 \perp\!\!\!\perp 4 \mid 5$	120	600	120	600
		Total **V** :	300	1740	300	1740
40		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $2 \perp\!\!\!\perp \{4,5\} \mid \{1,3\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	12	24	12	24 (S)
41		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $\{2,3\} \perp\!\!\!\perp 5 \mid 4$ $2 \perp\!\!\!\perp \{4,5\} \mid 3$	60	420	60	420
42		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $2 \perp\!\!\!\perp \{4,5\}$ $3 \perp\!\!\!\perp 5 \mid 4$	60	180	60	180
		Total **VI** :	132	624	132	624
43		$4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3 \mid 4$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$	60	420	60	420
44		$4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $2 \perp\!\!\!\perp 5 \mid 1$	120	840	120	840
45		$4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3 \mid 2$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$	60	300	60	300
46		$4 \perp\!\!\!\perp \{1,2,3\}$ $1 \perp\!\!\!\perp 3 \mid 2$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$	60	300	60	300

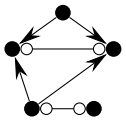
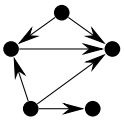
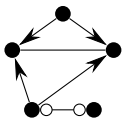
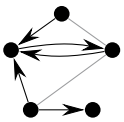
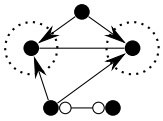
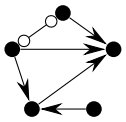
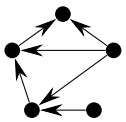
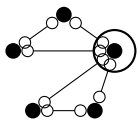
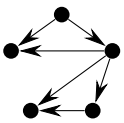
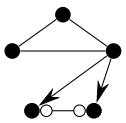
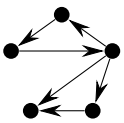
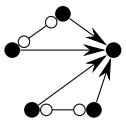
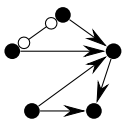
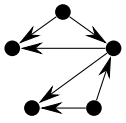
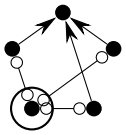
47		$4 \perp\!\!\!\perp \{2,3\}$ $4 \perp\!\!\!\perp 1 \mid \{2,3,5\}$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $2 \perp\!\!\!\perp 5 \mid 3$	120	360		120	360
48		$4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $2 \perp\!\!\!\perp 5$	60	180		60	180
49		$4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$	60	60		60	60
50		$4 \perp\!\!\!\perp \{1,2,3\}$ $1 \perp\!\!\!\perp 3$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$	60	60		60	60
51		$4 \perp\!\!\!\perp \{2,3\} \mid 1$ $4 \perp\!\!\!\perp 1 \mid \{3,5\}$ $1 \perp\!\!\!\perp 3 \mid 5$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$	120	120		120	120
52		$4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$	60	120		60	120 (S)
		Total **VII** :	780	2760		780	2760
53.1		$\{1,2\} \perp\!\!\!\perp \{3,4\} \mid 5$ $3 \perp\!\!\!\perp 4 \mid 5$	30	1680		30	1620
53.2						30	60 (S)
54		$\{1,2\} \perp\!\!\!\perp 3$ $\{1,2\} \perp\!\!\!\perp 4 \mid 5$ $3 \perp\!\!\!\perp 4 \mid 5$	60	180		60	180
55		$\{1,2\} \perp\!\!\!\perp \{3,4\}$ $3 \perp\!\!\!\perp 4$	30	90		30	90
56		$\{1,2\} \perp\!\!\!\perp \{3,4\} \mid 5$ $3 \perp\!\!\!\perp 4$	30	150		30	150

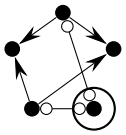
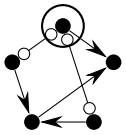
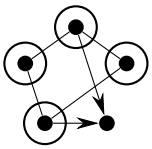
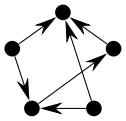
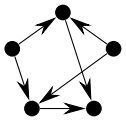
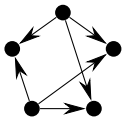
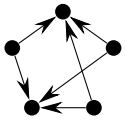
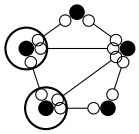
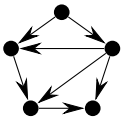
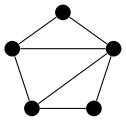
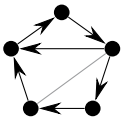
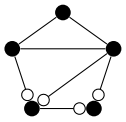
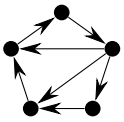
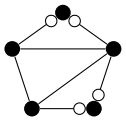
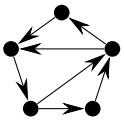
57		$1 \perp\!\!\!\perp 3$ $\{1,2\} \perp\!\!\!\perp 4 \mid 5$ $3 \perp\!\!\!\perp \{2,4\} \mid 5$	120	120		120	120
58		$1 \perp\!\!\!\perp \{3,4\}$ $2 \perp\!\!\!\perp \{3,4\} \mid \{1,5\}$ $3 \perp\!\!\!\perp 4$	60	60		60	60
		Total **VIII** :	330	2280		360	2280
59.1		$3 \perp\!\!\!\perp \{1,4,5\} \mid 2$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$	60	3360		60	3240
59.2						60	120 (S)
60		$3 \perp\!\!\!\perp \{1,4,5\}$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$	120	600		120	600
61		$3 \perp\!\!\!\perp \{4,5\}$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3 \mid 2$	120	360		120	360
62		$3 \perp\!\!\!\perp \{4,5\} \mid \{1,2\}$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$ $1 \perp\!\!\!\perp 3$	120	120		120	120
63		$3 \perp\!\!\!\perp \{4,5\} \mid \{1,2\}$ $4 \perp\!\!\!\perp \{1,2,3\}$ $1 \perp\!\!\!\perp 3$	120	120		120	120
		Total **IX** :	540	4560		600	4560
64.1		$4 \perp\!\!\!\perp \{1,2,5\} \mid 3$ $3 \perp\!\!\!\perp \{1,5\} \mid 2$	60	3360		60	3240
64.2						60	120 (S)
65.1		$4 \perp\!\!\!\perp \{1,2,5\}$ $3 \perp\!\!\!\perp \{1,5\} \mid 2$	60	2160		60	2040

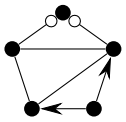
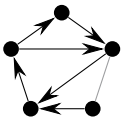
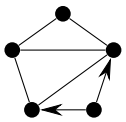
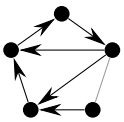
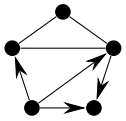
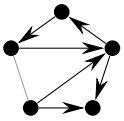
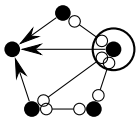
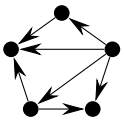
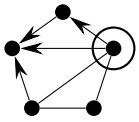
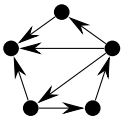
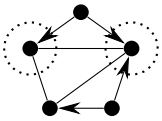
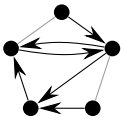
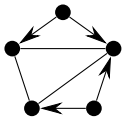
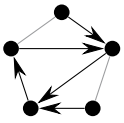
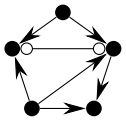
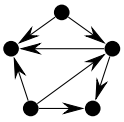
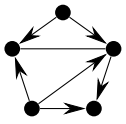
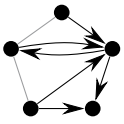
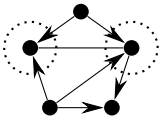
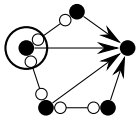
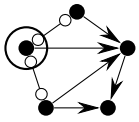
65.2						60	120 (S)
66		$4 \perp\!\!\!\perp \{1,2,5\} \mid 3$ $3 \perp\!\!\!\perp 5$ $3 \perp\!\!\!\perp 1 \mid \{2,5\}$	120	360		120	260
67		$\{3,4\} \perp\!\!\!\perp \{1,5\}$ $2 \perp\!\!\!\perp 4 \mid 3$	60	540		60	540
		Total **X** :	300	6420		420	6420
68.1		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $2 \perp\!\!\!\perp 4 \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	60	720		60	600 (240)
68.2						60	120 (S)
69.1		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $2 \perp\!\!\!\perp 4 \mid 5$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	120	5520		120	5280
69.2						120	240 (S)
70		$1 \perp\!\!\!\perp \{3,4\} \mid 5$ $2 \perp\!\!\!\perp 4 \mid \{3,5\}$ $3 \perp\!\!\!\perp 5$	120	360		120	360
71		$1 \perp\!\!\!\perp \{3,4\} \mid 2$ $2 \perp\!\!\!\perp 4 \mid 3$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	120	840		120	840
72		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $2 \perp\!\!\!\perp 4 \mid 3$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	120	600		120	600
73		$1 \perp\!\!\!\perp \{3,4\}$ $2 \perp\!\!\!\perp 4 \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid 4$	120	360		120	360
74		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $2 \perp\!\!\!\perp 4 \mid \{3,5\}$ $3 \perp\!\!\!\perp 5$	120	120		120	120

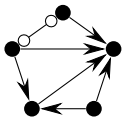
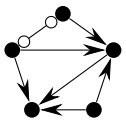
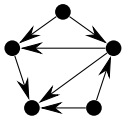
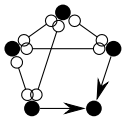
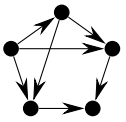
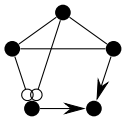
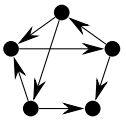
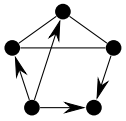
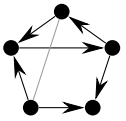
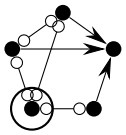
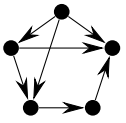
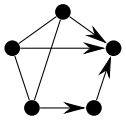
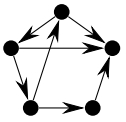
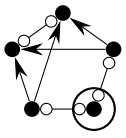
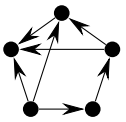
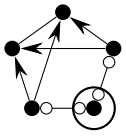
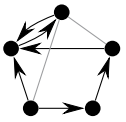
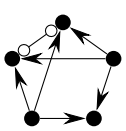
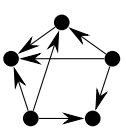
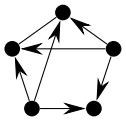
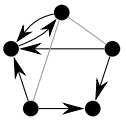
75		$1 \perp\!\!\!\perp 3$ $\{1,2\} \perp\!\!\!\perp 4 \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2\}$	120	120		120	120
		Total **XI** :	900	8640		1080	8640
76.1		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$	60	9300		60	7860
76.2						120	1200 (480)
76.3						120	240 (S)
77		$1 \perp\!\!\!\perp 3 \mid 5$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$	60	420		60	420
78.1		$1 \perp\!\!\!\perp 3$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$	60	540		60	420
78.2						60	120
79		$1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$	60	300		60	300
80		$1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp \{1,2,3\}$	60	300		60	300
81		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp \{2,3\}$ $4 \perp\!\!\!\perp 1 \mid \{2,5\}$	120	360		120	360
82		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp \{1,3\} \mid \{2,5\}$ $4 \perp\!\!\!\perp 2$	60	60		60	60
83		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $4 \perp\!\!\!\perp 3$	120	120		120	120

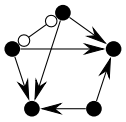
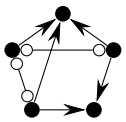
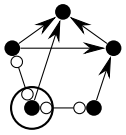
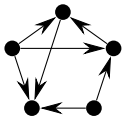
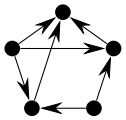
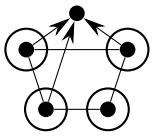
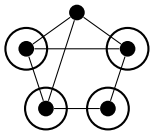
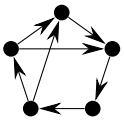
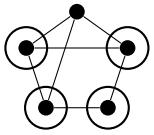
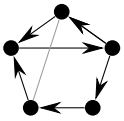
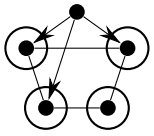
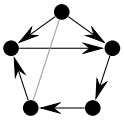
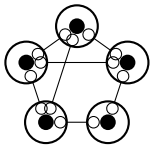
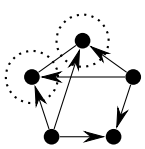
84		$1 \perp\!\!\!\perp 3 \mid 5$ $4 \perp\!\!\!\perp \{2,3\} \mid \{1,5\}$ $4 \perp\!\!\!\perp 1$	120	120		120	120
85		$1 \perp\!\!\!\perp 3$ $4 \perp\!\!\!\perp \{1,3\}$ $4 \perp\!\!\!\perp 2 \mid \{1,3,5\}$	60	60		60	60
86		$1 \perp\!\!\!\perp 3$ $4 \perp\!\!\!\perp \{1,2,3\}$	60	60		60	60
87		$1 \perp\!\!\!\perp 3$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp \{1,2,3\} \mid 5$	60	120		60	120
		Total **XII** :	900	11760		1200	11760
88.1		$4 \perp\!\!\!\perp \{1,2,5\} \mid 3$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$	60	10020		60	8340
88.2						60	600 (240)
88.3						60	120 (S)
88.4						60	600 (240)
88.5						60	360 (240)
89.1		$4 \perp\!\!\!\perp \{1,2,5\}$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$	60	2220		60	2100
89.2						60	120 (S)
90		$4 \perp\!\!\!\perp \{1,2,5\} \mid 3$ $1 \perp\!\!\!\perp 3 \mid 2$	120	840		120	840

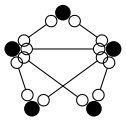
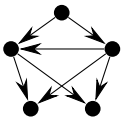
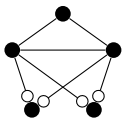
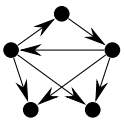
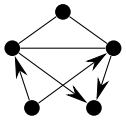
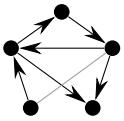
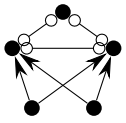
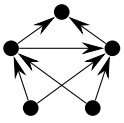
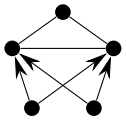
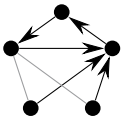
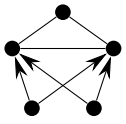
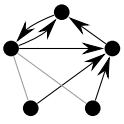
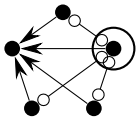
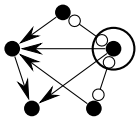
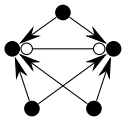
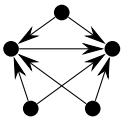
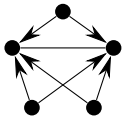
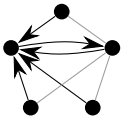
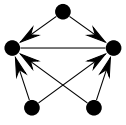
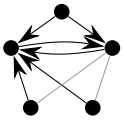
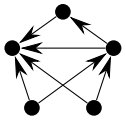
91.1		$4 \perp\!\!\!\perp \{1,2,5\} \mid 3$ $1 \perp\!\!\!\perp 3$	60	1620		60	1260
91.2						60	360
92		$4 \perp\!\!\!\perp \{1,2,5\} \mid 3$ $1 \perp\!\!\!\perp 3$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$	60	360		60	360
93		$4 \perp\!\!\!\perp \{1,2\}$ $4 \perp\!\!\!\perp 5 \mid \{2,3\}$ $1 \perp\!\!\!\perp 3 \mid 2$	120	360		120	360
94		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $4 \perp\!\!\!\perp 5$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$	120	120		120	120
		Total **XIII** :	600	15540		960	15540
95.1		$\{1,2\} \perp\!\!\!\perp \{3,4\} \mid 5$	15	5025		15	4725
95.2						30	300 (120)
96		$\{1,2\} \perp\!\!\!\perp \{3,4\}$	15	135		15	135
97		$\{1,2\} \perp\!\!\!\perp 3$ $\{1,2\} \perp\!\!\!\perp 4 \mid \{3,5\}$	60	180		60	180
98		$1 \perp\!\!\!\perp \{3,4\}$ $2 \perp\!\!\!\perp \{3,4\} \mid \{1,5\}$	60	60		60	60
		Total **XIV** :	150	5400		180	5400
99		$1 \perp\!\!\!\perp 3 \mid \{2,4,5\}$ $2 \perp\!\!\!\perp \{4,5\} \mid 3$ $4 \perp\!\!\!\perp 5 \mid 3$	20	140		20	140

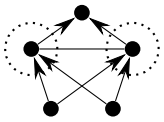
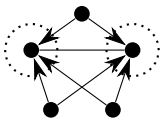
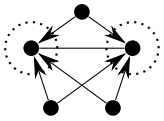
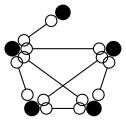
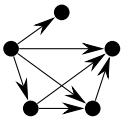
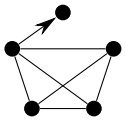
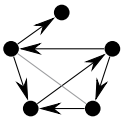
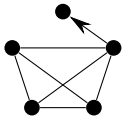
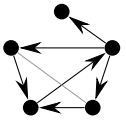
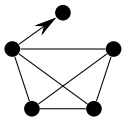
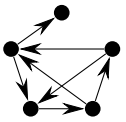
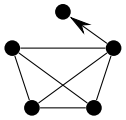
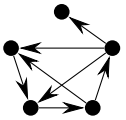
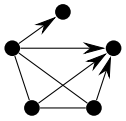
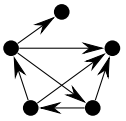
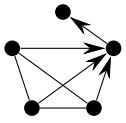
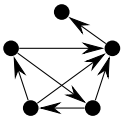
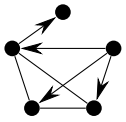
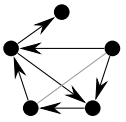
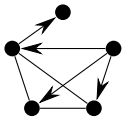
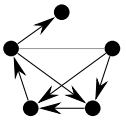
100		$1 \perp\!\!\!\perp 3 \mid 4$ $2 \perp\!\!\!\perp \{4,5\} \mid \{1,3\}$ $4 \perp\!\!\!\perp 5 \mid \{1,3\}$	30	150		30	150
101		$1 \perp\!\!\!\perp 3 \mid \{2,4\}$ $2 \perp\!\!\!\perp 4 \mid 1$ $\{2,4\} \perp\!\!\!\perp 5 \mid \{1,3\}$	60	300		60	300
102		$1 \perp\!\!\!\perp 3 \mid \{2,4\}$ $2 \perp\!\!\!\perp 5 \mid \{1,3\}$ $4 \perp\!\!\!\perp \{2,5\} \mid \{1,3\}$	30	60		30	60 (S)
103		$1 \perp\!\!\!\perp 3 \mid \{2,4,5\}$ $2 \perp\!\!\!\perp 4$ $\{2,4\} \perp\!\!\!\perp 5 \mid 3$	60	60		60	60
104		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $\{2,5\} \perp\!\!\!\perp 4 \mid \{1,3\}$ $2 \perp\!\!\!\perp 5$	30	30		30	30
105		$1 \perp\!\!\!\perp 3$ $2 \perp\!\!\!\perp \{4,5\} \mid \{1,3\}$ $4 \perp\!\!\!\perp 5 \mid \{1,3\}$	10	10		10	10
106		$1 \perp\!\!\!\perp 3 \mid \{2,4,5\}$ $2 \perp\!\!\!\perp 4 \perp\!\!\!\perp 5$	10	10		10	10
		Total **XV** :	250	760		250	760
107.1		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$	60	23460		60	15840
107.2						60	180 (S)
107.3						120	2640 (600)
107.4						60	3000 (480)

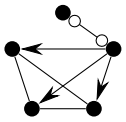
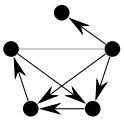
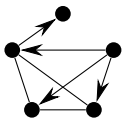
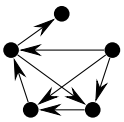
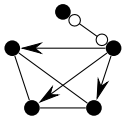
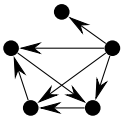
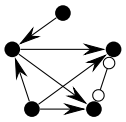
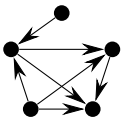
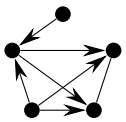
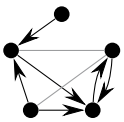
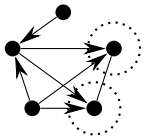
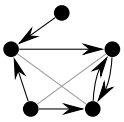
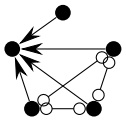
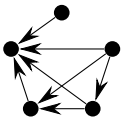
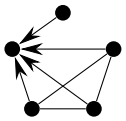
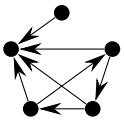
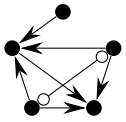
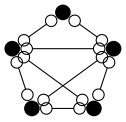
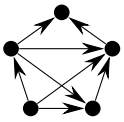
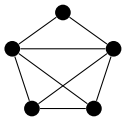
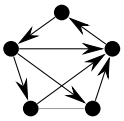
107.5						120	1200 (480)
107.6						120	360 (120)
107.7						120	240 (S)
108.1		$1 \perp\!\!\!\perp 3 \mid 5$ $4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$	120	5520		120	5280
108.2						120	240 (S)
109.1		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp 1$ $4 \perp\!\!\!\perp 2 \mid \{3,5\}$	60	300		60	180
109.2						60	120 (S)
110.1		$1 \perp\!\!\!\perp 3$ $4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$	120	1060		120	840
110.2						120	240
111		$1 \perp\!\!\!\perp 3$ $1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$	120	240		120	240
112		$1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp \{1,2\} \mid 3$	60	420		60	420
113		$1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$	120	600		120	600

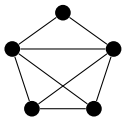
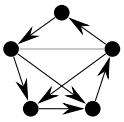
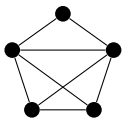
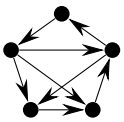
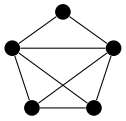
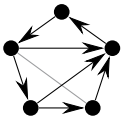
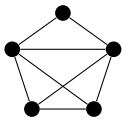
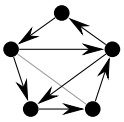
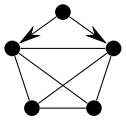
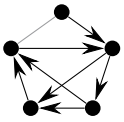
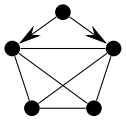
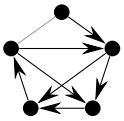
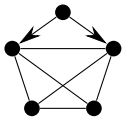
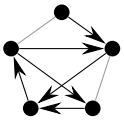
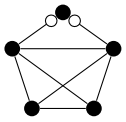
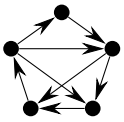
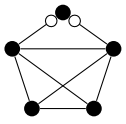
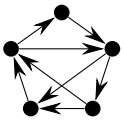
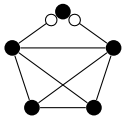
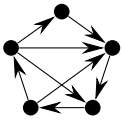
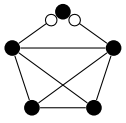
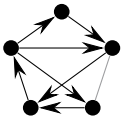
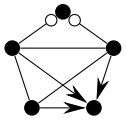
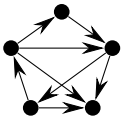
114		$1 \perp\!\!\!\perp 3 \mid 2$ $4 \perp\!\!\!\perp \{1,2\}$	120	360			120	360
115		$1 \perp\!\!\!\perp 3 \mid \{2,4,5\}$ $4 \perp\!\!\!\perp \{1,2\}$	120	360			120	360
116		$1 \perp\!\!\!\perp 3 \mid \{2,4,5\}$ $4 \perp\!\!\!\perp 1$ $4 \perp\!\!\!\perp 2 \mid \{1,5\}$	120	120			120	120
		Total **XVI** :	1020	32440			1920	32440
117.1		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2\}$	30	4110			30	3390
117.2							60	600 (240)
117.3							60	120 (S)
118.1		$4 \perp\!\!\!\perp \{1,2\} \mid 3$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	60	2760			60	2640
118.2							60	120 (S)
119.1		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid 4$	30	1350			30	1050
119.2							30	300
120.1		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $3 \perp\!\!\!\perp 5$	30	270			30	210
120.2							30	60

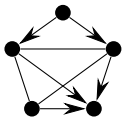
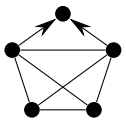
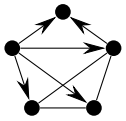
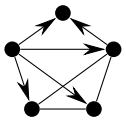
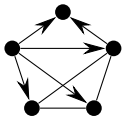
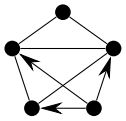
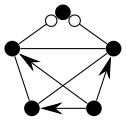
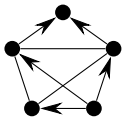
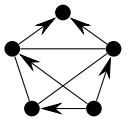
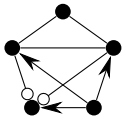
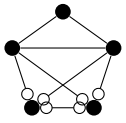
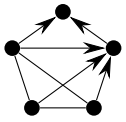
121		$4 \perp\!\!\!\perp \{1,2\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	30	90			30	90
122		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid 2$	60	300			60	300
123		$4 \perp\!\!\!\perp 1 \mid \{2,3,5\}$ $4 \perp\!\!\!\perp 2 \mid 3$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	120	600			120	600
124		$4 \perp\!\!\!\perp 1 \mid \{2,5\}$ $4 \perp\!\!\!\perp 2$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	120	120			120	120
125		$4 \perp\!\!\!\perp 1 \{2,3,5\}$ $4 \perp\!\!\!\perp 2$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	60	60			60	60
126		$4 \perp\!\!\!\perp 1 \mid \{2,3,5\}$ $4 \perp\!\!\!\perp 2 \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	60	120			60	120 (S)
127.1 X		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	30	1080			30	780 (240)
127.2							60	180 (S)
127.3							60	120 (S)
128		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid 4$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	30	300			30	300 (0)
129		$4 \perp\!\!\!\perp \{1,2\} \mid \{3,5\}$ $3 \perp\!\!\!\perp 5$ $3 \perp\!\!\!\perp 5 \mid \{1,2\}$	30	60			30	60
		Total **XVII** :	?	11220	?		11220	

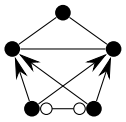
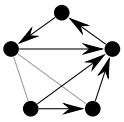
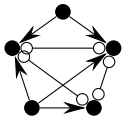
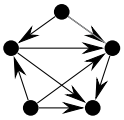
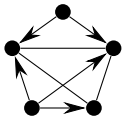
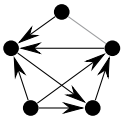
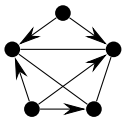
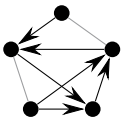
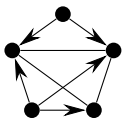
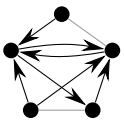
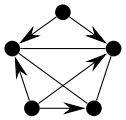
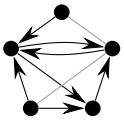
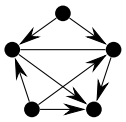
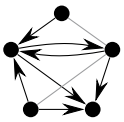
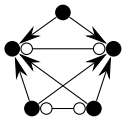
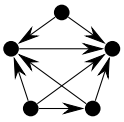
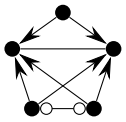
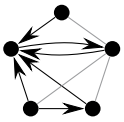
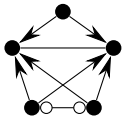
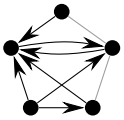
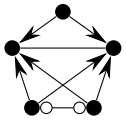
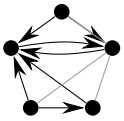
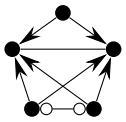
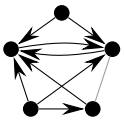
130.1		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $3 \perp\!\!\!\perp 4 \mid \{2,5\}$	10	3060		10	2400
130.2						30	540 (180)
130.3						60	120 (S)
131.1		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $3 \perp\!\!\!\perp 4$	30	540		30	210
131.2						30	60 (S)
131.3						30	300
132		$1 \perp\!\!\!\perp \{3,4\} \mid 5$ $3 \perp\!\!\!\perp 4 \mid 5$	20	140		20	140
133		$1 \perp\!\!\!\perp 3 \mid \{2,5\}$ $1 \perp\!\!\!\perp 4 \mid 5$ $3 \perp\!\!\!\perp 4 \mid \{2,5\}$	60	300		60	300
134.1		$1 \perp\!\!\!\perp \{3,4\}$ $3 \perp\!\!\!\perp 4$	10	?		10	90
134.2						10	20
134.3						30	60
135		$1 \perp\!\!\!\perp \{3,4\} \mid 5$ $3 \perp\!\!\!\perp 4$	60	60		60	60

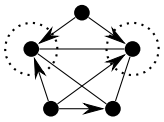
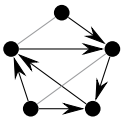
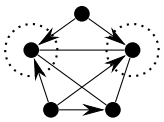
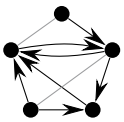
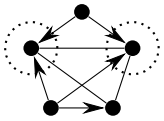
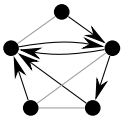
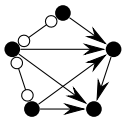
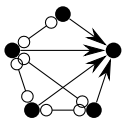
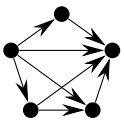
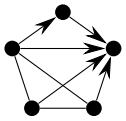
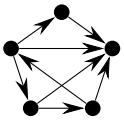
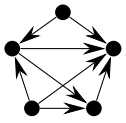
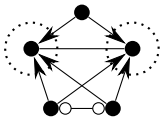
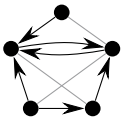
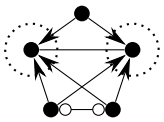
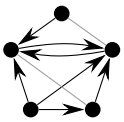
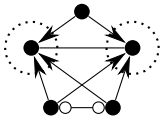
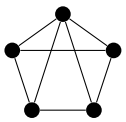
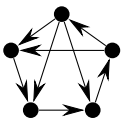
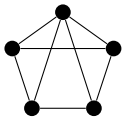
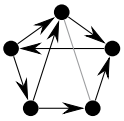
136		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$ $3 \perp\!\!\!\perp 4$ $3 \perp\!\!\!\perp 4 \mid \{2,5\}$	30	60		30	60
137		$1 \perp\!\!\!\perp 3 \perp\!\!\!\perp 4$ $1 \perp\!\!\!\perp 4 \mid \{2,5\}$ $3 \perp\!\!\!\perp 4 \mid \{2,5\}$	30	60		30	60
138		$1 \perp\!\!\!\perp 3 \perp\!\!\!\perp 4$ $3 \perp\!\!\!\perp 4 \mid \{2,5\}$	30	60		30	60
		Total **XVIII** :	?	4450	?	4450	
139.1		$1 \perp\!\!\!\perp \{3,4,5\} \mid 2$	20	45560		20	31940
139.2						60	180 (S)
139.3						20	60 (S)
139.4 X						60	1560 (240)
139.5 X						60	1560 (240)
139.6						60	120 (S)
139.7						20	40 (S)
139.8						60	120 (S)
139.9						120	240 (S)

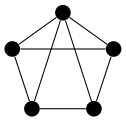
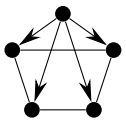
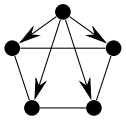
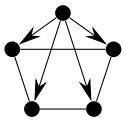
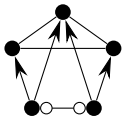
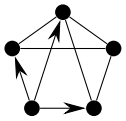
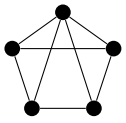
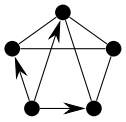
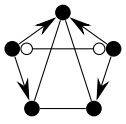
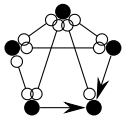
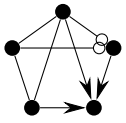
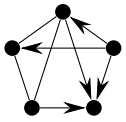
139.10						60	360 (240)
139.11						60	120 (S)
139.12						20	120 (80)
	non-simple DEC **139.13**-**139.29** omitted					1140	9140
140.1		$1 \perp\!\!\!\perp \{4,5\} \mid \{2,3\}$ $1 \perp\!\!\!\perp 3$	60	660		60	420
140.2						60	120
140.3						60	120
141.1		$1 \perp\!\!\!\perp \{3,4,5\}$	20	720		20	680
141.2						20	40 (S)
142		$1 \perp\!\!\!\perp \{3,5\}$ $1 \perp\!\!\!\perp 4 \mid \{2,3,5\}$	60	180		60	180
		Total **XIX** :	180	47120		2040	47120
143.1		$1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$	15	130710		30	79680
143.2						60	240 (60)

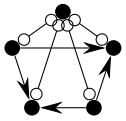
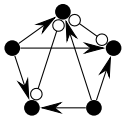
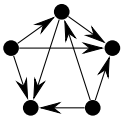
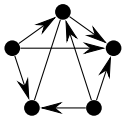
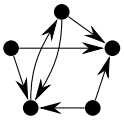
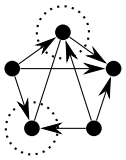
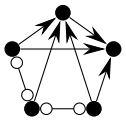
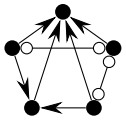
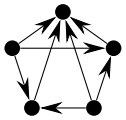
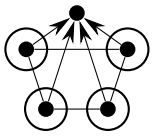
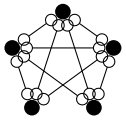
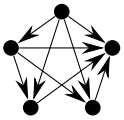
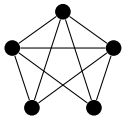
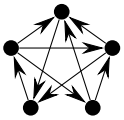
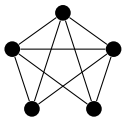
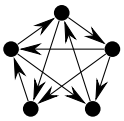
143.3			120	480 (240)
143.4			60	300 (120)
143.5			120	480 (120)
143.6			30	120 (S)
143.7 X			30	780 (120)
143.8			120	480 (120)
143.9			60	180 (S)
143.10 X			120	5040 (600)
143.11 X			30	3900 (240)
143.12 X			30	1260 (120)
143.13			60	900 (360)
143.14			60	600 (240)

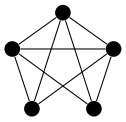
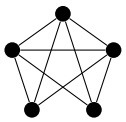
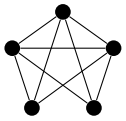
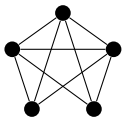
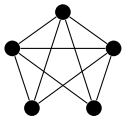
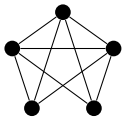
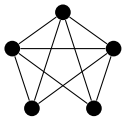
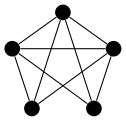
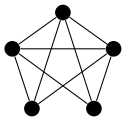
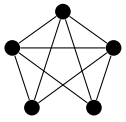
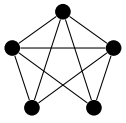
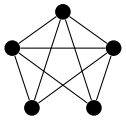
143.15		60	120 (S)
143.16		60	180 (S)
143.17		60	120 (S)
143.18		120	240 (S)
143.19		60	120 (S)
143.20		120	360 (120)
143.21		60	600 (240)
143.22		60	120 (S)
143.23		120	240 (S)
143.24		60	840 (240)
143.25		30	6000 (360)
143.26		60	120 (S)

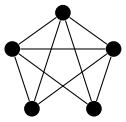
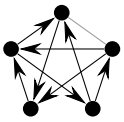
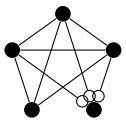
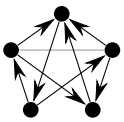
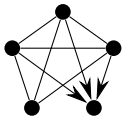
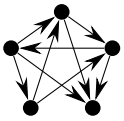
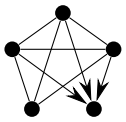
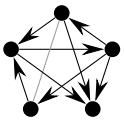
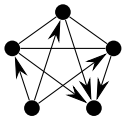
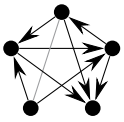
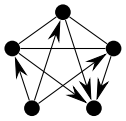
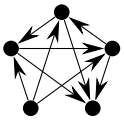
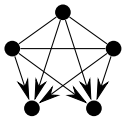
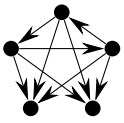
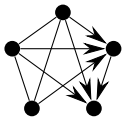
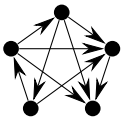
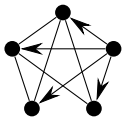
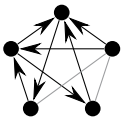
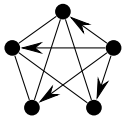
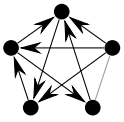
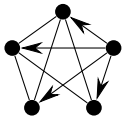
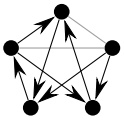
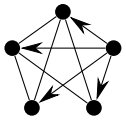
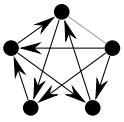
143.27						30	180 (120)
	non-simple DEC **143.28**-**143.60** omitted					2280	27030
144.1		$1 \perp\!\!\!\perp 3$ $1 \perp\!\!\!\perp 4 \mid \{2,3,5\}$	60	2340		60	1140
144.2						60	120 (S)
144.3						120	240 (S)
144.4						120	360
144.5						120	360
144.6						60	120
145.1		$1 \perp\!\!\!\perp \{3,4\}$	30	1710		30	810
145.2						30	180
145.3						60	360
145.4						30	180
145.5						30	180

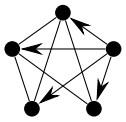
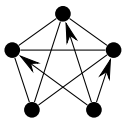
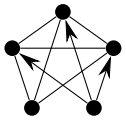
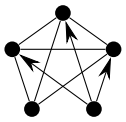
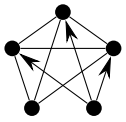
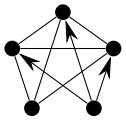
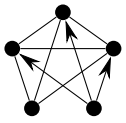
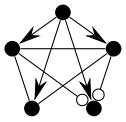
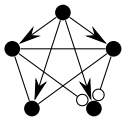
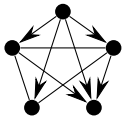
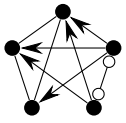
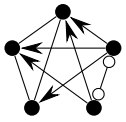
146.1		$1 \perp\!\!\!\perp 3$ $1 \perp\!\!\!\perp 4 \mid \{2,5\}$	60	1320		60	120 (S)
146.2						60	840
146.3						120	360
147		$1 \perp\!\!\!\perp 3 \mid 2$ $1 \perp\!\!\!\perp 4 \mid \{2,3,5\}$	120	600		120	600
148.1		$1 \perp\!\!\!\perp \{3,4\} \mid 2$	60	2760		60	2640
148.2						60	120 (S)
149		$1 \perp\!\!\!\perp 3$ $1 \perp\!\!\!\perp 4 \mid \{2,3\}$	120	120		120	120
150.1		$1 \perp\!\!\!\perp \{3,4\}$ $1 \perp\!\!\!\perp 4 \mid \{2,3,5\}$	60	720		60	360
150.2						60	360
151		$1 \perp\!\!\!\perp \{3,4\}$ $1 \perp\!\!\!\perp \{3,4\} \mid \{2,5\}$	30	180		30	180
		Total **XX** :	555	140460		5580	140460
152.1 X		$2 \perp\!\!\!\perp 4 \mid \{1,3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	15	6990		60	3000 (240)
152.2						60	900 (360)

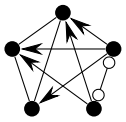
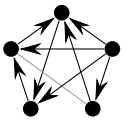
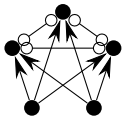
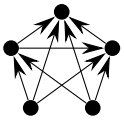
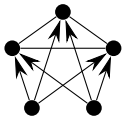
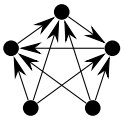
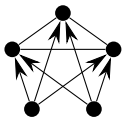
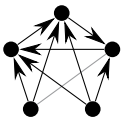
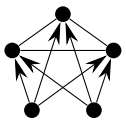
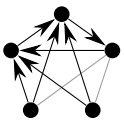
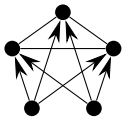
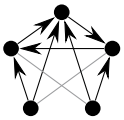
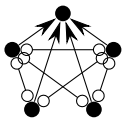
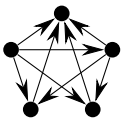
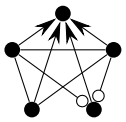
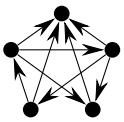
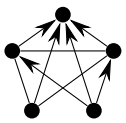
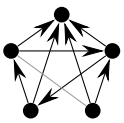
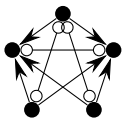
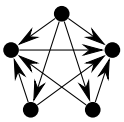
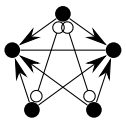
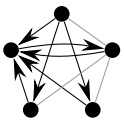
152.3					60	240 (120)
152.4					15	30 (S)
152.5					60	120 (S)
152.6					30	60 (S)
152.7					60	360 (240)
152.8					120	360 (120)
152.9					15	1140
152.10					60	240
152.11					60	540
153.1		$2 \perp\!\!\!\perp 4 \mid \{1,3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2\}$	60	8220	60	6780
153.2					120	1200 (480)
153.3					120	240 (S)

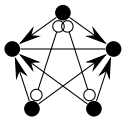
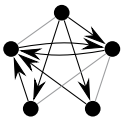
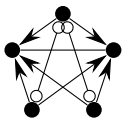
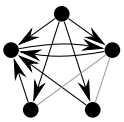
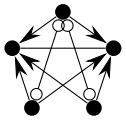
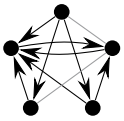
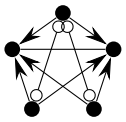
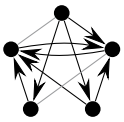
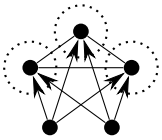
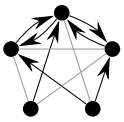
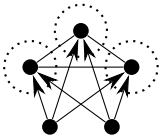
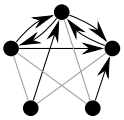
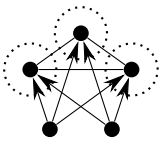
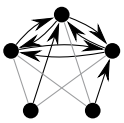
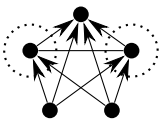
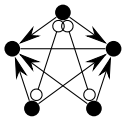
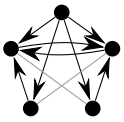
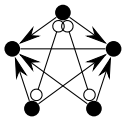
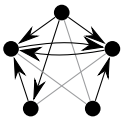
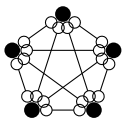
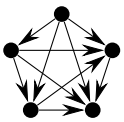
154		$2 \perp\!\!\!\perp 4 \mid 1$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	30	150		30	150
155.1		$2 \perp\!\!\!\perp 4$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	30	510		30	390
155.2						60	120
156		$2 \perp\!\!\!\perp 4$ $2 \perp\!\!\!\perp 4 \mid \{1,3\}$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	60	120		60	120
157		$2 \perp\!\!\!\perp 4 \mid 3$ $3 \perp\!\!\!\perp 5 \mid \{1,2,4\}$	60	300		60	300
158		$2 \perp\!\!\!\perp 4 \mid 5$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	60	300		60	300
159		$2 \perp\!\!\!\perp 4$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	30	30		30	30
160		$2 \perp\!\!\!\perp 4 \mid \{3,5\}$ $3 \perp\!\!\!\perp 5 \mid \{2,4\}$	15	30		15	30 (S)
		Total **XXI** :	360	16650		1245	16650
161.1		$3 \perp\!\!\!\perp 4 \mid \{1,2,5\}$	10	221300		10	91260
161.2X						60	22620 (600)
161.3X						60	5400 (240)

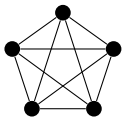
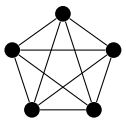
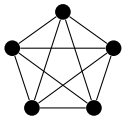
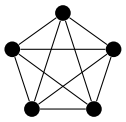
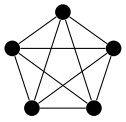
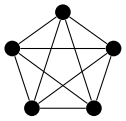
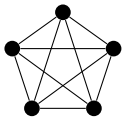
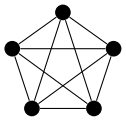
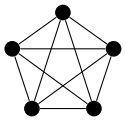
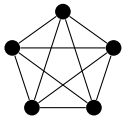
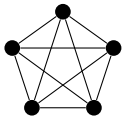
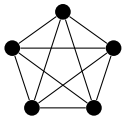
161.4 X		60	2100 (420)
161.5		120	600 (120)
161.6		60	240 (S)
161.7		120	720 (120)
161.8		60	240 (120)
161.9		120	2400 (240)
161.10		60	360 (180)
161.11		60	480 (120)
161.12		120	600 (120)
161.13		120	600 (120)
161.14		60	300 (120)
161.15		120	720 (120)

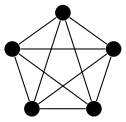
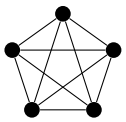
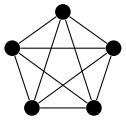
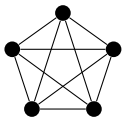
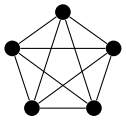
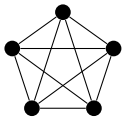
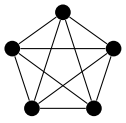
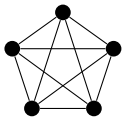
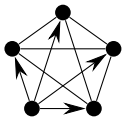
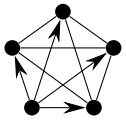
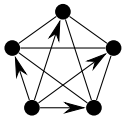
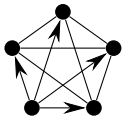
161.16			60	1200 (240)
161.17			20	420 (120)
161.18 X			60	1560 (240)
161.19			60	180 (S)
161.20			60	120 (S)
161.21			20	40 (S)
161.22			10	20 (S)
161.23			60	120 (S)
161.24			120	360 (120)
161.25			120	360 (120)
161.26			120	360 (120)
161.27			120	360 (120)

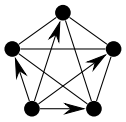
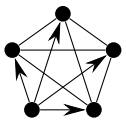
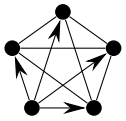
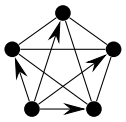
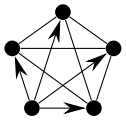
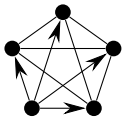
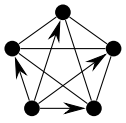
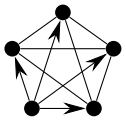
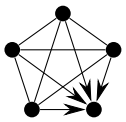
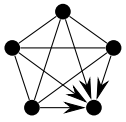
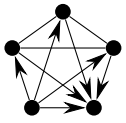
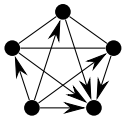
161.28		120	360 (120)
161.29		60	180 (S)
161.30		120	480 (120)
161.31		120	480 (120)
161.32		60	1260 (120)
161.33		20	60 (S)
161.34		60	900 (120)
161.35		120	1680 (480)
161.36		60	840 (240)
161.37		60	120 (S)
161.38		120	720 (480)
161.39		60	360 (240)

161.40						60	360 (240)
	non-simple DEC **161.41**-**161.95** omitted					4040	79760
162.1		$3 \perp\!\!\!\perp 4$	10	5640		10	2230
162.2						10	20 (S)
162.3						60	120 (S)
162.4						30	60 (S)
162.5						60	120 (S)
	non-simple DEC **162.6**-**162.23** omitted					910	3090
163.1		$3 \perp\!\!\!\perp 4 \mid \{2,5\}$	30	4110		30	3390
163.2						60	600 (240)
163.3						60	120 (S)
164.1		$3 \perp\!\!\!\perp 4 \mid 1$	30	3450		30	1350
164.2						30	300

164.3						30	300
164.4						30	300
164.5						60	600
164.6						60	600
165.1		$3 \perp\!\!\!\perp 4$ $3 \perp\!\!\!\perp 4 \mid \{1,2,5\}$	10	660		60	180
165.2						30	120
165.3						10	360
166		$3 \perp\!\!\!\perp 4$ $3 \perp\!\!\!\perp 4 \mid \{2,5\}$	30	60		30	60
167.1		$3 \perp\!\!\!\perp 4 \mid 1$ $3 \perp\!\!\!\perp 4 \mid \{1,2,5\}$	30	900		30	300
167.2						60	600
		Total **XXII** :	150	236120		8810	236120
168.1		(None)	1	479306		1	202516

168.2 X		10	11420 (120)
168.3 X		15	9990 (120)
168.4 X		10	630 (60)
168.5 X		30	600 (120)
168.6		120	600 (240)
168.7		60	360 (120)
168.8		120	720 (120)
168.9		120	3000 (240)
168.10		60	360 (60)
168.11		60	180 (S)
168.12		120	840 (120)
168.13		30	150 (S)

168.14		5	20 (S)
168.15		60	360 (120)
168.16		60	480 (120)
168.17		120	1200 (120)
168.18		20	220 (40)
168.19		30	1050 (60)
168.20		60	480 (120)
168.21		60	2640 (180)
168.22 X		30	1140 (120)
168.23 X		30	1140 (120)
168.24		20	60 (S)
168.25		120	480 (120)

168.26		60	180 (S)
168.27		120	480 (120)
168.28		120	480 (120)
168.29		60	900 (120)
168.30		120	480 (120)
168.31		120	480 (120)
168.32		20	60 (S)
168.33		60	1260 (120)
168.34 X		30	780 (120)
168.35		20	60 (S)
168.36		60	120 (S)
168.37		20	40 (S)

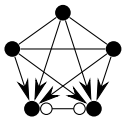
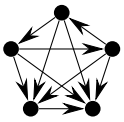
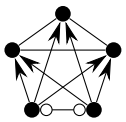
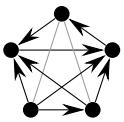
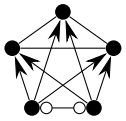
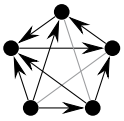
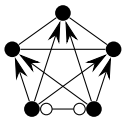
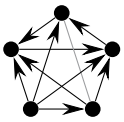
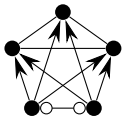
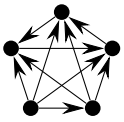
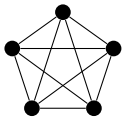
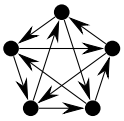
168.38			10	180 (40)
168.39			30	180 (120)
168.40			60	360 (240)
168.41			60	360 (240)
168.42			10	60 (40)
168.43			12	156 (24)
	non-simple DEC **168.44**-**168.217** omitted		8771	232054
	Total **XXII** : 1	479306	11134	479306

Table 5.4 Atlas: $p = 5$

6 Causal Structure Learning

Causal structure learning is the task of estimating or learning the causal relationships from given data. In the last decades technological developments made it possible to collect and store *big data* with huge numbers of variables and sample sizes. Without the help of automated search methods causal discovery would be impossible, but the availability of faster computers and larger memories also allow the implementation of computationally expensive search algorithms over large search spaces.
We assume that the causal relations are represented by a directed graph and the inputs are a set of observable data points, that is, there are no unmeasured causes. The goal of causal structure learning is to use computationally and statistically efficient algorithms for discovering the causal relationships that are correct in the large sample limit. The output is then a set of graphs or a possibly single graph, where a directed edge from i to j means that i is a direct cause of j. In this task there are two main problems of interest: (1) estimate the edge weights, i.e. the strength of causal effects, and (2) learn the underlying graph structure. A common method for finding the strength of causal effects is to first perform a search for a causal graph (or a set of causal graphs) and then estimate parameter values using the causal graph together with the data. Compare [Maathuis et al., 2018][Chapter 18, Introduction] for a more detailed description of the problem; and [Uhler, 2017] for dealing with problem (1): estimate the edge weights in the Gaussian setting.
In the following we will tackle the second problem: Learn the underlying graph structure from data.
One major difficulty we face with this task is the vast search space, the number of directed graphs grows super-exponentially with the number of variables (for example, for 10 vertices we roughly have $1,23 \cdot 10^{27}$ directed graphs), so checking each graph is impossible.

We propose a new hybrid algorithm which combines the score-based algorithm by [Ghassami et al., 2020] using greedy search methods, with the CCD algorithm by [Richardson, 1996b] to restrict the search space. In the update steps we compute the maximum likelihood estimates using a block-coordinate descent method proposed by [Drton et al., 2019].
Before we start let us review some of the work that has been done in this area.

6.1 Prior Work

There exists several algorithms for learning the structure of a DAG. Here one divides mainly between score-based and constraint-based approaches. The most prominent representatives of these two approaches are unarguably the Greedy equivalence search (GES) algorithm proposed by [Chickering, 2002] and the PC algorithm from [Spirtes et al., 1993].
GES is a score-based algorithm that minimizes a penalized likelihood score. GES starts with an empty graph, conducting first a forward phase which only adds edges and then a backward phase which only deletes edges. Both phases end if there is no improvement in score. The output is a CPDAG that represents the MEC. Moreover GES assumes the Markov and Faithfulness assumption. In [Nandy et al., 2018] it is shown that GES is consistent in the sparse high-dimensional setting. *Consistency* of a scoring criterion means that the ground truth structure has a lower score than any DAG that that is not in the MEC of the ground truth. This is not the only assumption on the score made for GES; *score equivalence* means that all DAGs in the MEC get the same score and *score decomposition* allows for fast computation of the score difference between DAGs that only differ by a few edges.
The PC algorithm is a constraint-based algorithm that assumes Faithfulness and causal sufficiency, i.e. there are no latent common causes. It performs conditional independence test to learn the structure of the

underlying DAG. The algorithm starts with a complete graph and in the first step determines the skeleton. In the second step it searches for unshielded colliders and in a last step further edge orientation rules are applied. The output is then a CPDAG. The PC algorithm was shown to be consistent in the sparse high-dimensional setting [Kalisch and Bühlmann, 2007].
The Fast Causal Inference (FCI) algorithm [Spirtes et al., 1993] is a modification of the PC algorithm, such that FCI also allows for latent variables. The output of the FCI is then a PAG.
Another approach for structure learning combines the two previous methods, so called hybrid methods. These algorithms usually start by conducting a constraint-based method to restrict the search space, followed by a score-based method that applies edge orientation rules. An example is given by [Nandy et al., 2018]. In this algorithm, called adaptively restricted GES, first the conditional independence graph is computed (this can be done by the first step of the PC algorithm) and then a greedy-search, based on GES, in the restricted search space is performed.
The mentioned algorithms aim to learn the MEC of a DAG, described by the CPDAG. Using some additional assumptions it is also possible to learn the true DAG. This idea was first used for linear non-Gaussian acyclic models (LiNGAM), proposed by [Shimizu et al., 2006]. Recall equation 2.2, LiNGAM is based on the fact that $X = A\varepsilon$ with $A = (I - B)^{-1}$. This means that X is a linear, invertible mixture of independent error with mixing matrix A. For non-Gaussian errors, A can be identified up to scaling and permutation of the columns by independent component analysis.
For a more detailed overview of existing approaches, including undirected graphical models and latent variable models, compare [Drton and Maathuis, 2017].

Structure learning algorithms in the presence of feedback loops in the underlying structure are considerably less studied. We already presented the CCD algorithm in *Section 3*, which can be seen as the first structure learning algorithm that can deal with cycles. Under the assumptions of causal sufficiency, Faithfulness, independent and identically distributed samples, positive variance of error terms, uncorrelated error terms, consistent tests for zero partial correlations and linearity of the structural equations, in the large sample limit, CCD outputs a PAG that represents the true graph [Richardson, 1996b]. As already mentioned, we use the CCD algorithm to construct a hybrid algorithm, and we will see that the restriction of the search space due to the PAG leads to preciser estimation of the ground truth structure as well as faster computation compared to a purely score-based approach.
Despite the more challenging task by including cycles in the structure, there has been advances in constructing structure learning algorithms that can deal with this problem, even when neglecting several assumptions such as causal sufficiency.
The Cyclic Causal Inference (CCI) algorithm proposed by [Strobl, 2019] can handle cycles, latent variables and several degrees of selection bias. Roughly speaking, selection bias occurs if the probability of a unit being sampled depends on certain properties of the unit [Zhang, 2008a]. The CCI algorithm is a constraint-based approach and starts with discovering the skeleton and orientation of v-structures as it is done in the first two steps of the FCI algorithm. Long range d-separation relations, as they occur in cyclic graphs (compare *Theorem 3.2*), are checked in CCI's third step. Then non-minimal d-separating sets that not have been found in the first step are discovered in the fourth step. The remaining three steps of CCI apply additional orientation rules via the d-separating sets discovered in the previous steps. Finally the algorithm outputs a *maximal almost ancestral graph* (MAAG), where an almost ancestral graph satisfies only the first two conditions of Definition 3.6. Furthermore it has been shown that under the Markov and Faithfulness assumption, the CCI algorithm is sound [Strobl, 2019, Theorem 2].
A generalized version of the LiNGAM algorithm, called the LiNG algorithm, has been proposed by [Lacerda et al., 2012]. In this approach the assumption of acyclity in the graph is dropped.
Furthermore under the Markov and Faithfulness assumption, it has been shown that the FCI algorithm, originally invented for DAGs with latents, is sound and d-separation complete when applied in the cyclic setting. Moreover FCI can be used to consistently estimate the presence and absence of causal relations, the absence of latent confounders and the absence of specific cycles in the graph [M. Mooij and Claassen, 2020].

6.2 Likelihood Inference

Recall the density function of the multivariate Gaussian distribution (2.6), the likelihood function is

$$L(\mu, \Sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^p \det \Sigma}} \exp\Big\{ -\frac{1}{2}(X^{(i)} - \mu)^\top \Sigma^{-1}(X^{(i)} - \mu)\Big\}. \tag{6.1}$$

Given n i.i.d observations $X^{(1)}, \ldots, X^{(n)}$ from $\mathcal{N}(\mu, \Sigma)$, define the sample covariance matrix as

$$S = \frac{1}{n}\sum_{i=1}^{n}(X^{(i)} - \bar{X})(X^{(i)} - \bar{X})^\top$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X^{(i)}$ is the sample mean. The Gaussian log-likelihood in terms of (μ, Σ) is

$$\begin{aligned} l(\mu, \Sigma) &\propto -\frac{n}{2}\log\det(\Sigma) - \frac{1}{2}\sum_{i=1}^{n}(X^{(i)} - \mu)^\top \Sigma^{-1}(X^{(i)} - \mu) \\ &= -\frac{n}{2}\log\det(\Sigma) - \frac{1}{2}\operatorname{tr}\left(\Sigma^{-1}\Big(\sum_{i=1}^{n}(X^{(i)} - \mu)^\top (X^{(i)} - \mu)\Big)\right) \\ &= -\frac{n}{2}\log\det(\Sigma) - \frac{n}{2}\operatorname{tr}(S\Sigma^{-1}) - \frac{n}{2}(\bar{X} - \mu)^\top \Sigma^{-1}(\bar{X} - \mu), \end{aligned}$$

so in the unconstrained model where $(\mu, \Sigma) \in \mathbb{R}^p \times PD_V$ the MLE is given by

$$\hat{\mu} = \bar{X} \quad \text{and} \quad \hat{\Sigma} = S,$$

assuming $S \in PD_V$. Since Gaussian graphical models only pose constraints on the covariance matrix, we restrict ourselves to models where the mean μ is unconstrained. In this case we have $\hat{\mu} = \bar{X}$ and the MLE problem for Σ simplifies to

$$\begin{aligned} &\max_{\Sigma} -\log\det(\Sigma) - \operatorname{tr}(S\Sigma^{-1}) \\ &\text{subject to } \Sigma \in PD_V. \end{aligned} \tag{6.2}$$

Using that $-\log\det(\Sigma) = \log\det(\Theta)$, the ML estimation problem writes

$$\begin{aligned} &\max_{\Theta} \quad \log\det(\Theta) - \operatorname{tr}(S\Theta) \\ &\text{subject to} \quad \Theta \in PD_V^{-1}. \end{aligned} \tag{6.3}$$

The function $f(Y) = \log\det(Y) - \operatorname{tr}(SY)$ is concave on PD_V, so ML estimation for Gaussian graphical models solves a convex optimization problem [Uhler et al., 2013].
For learning DAGs a standard approach to penalize for overly complex models is to use a BIC score. Following the argumentation of the original paper we use the l_0-regularized negative log-likelihood function as the score [Ghassami et al., 2020].
The l_0-regularized maximum likelihood estimator solves the following unconstrained optimization problem:

$$\min_{\mathcal{G}} \min_{(B,\Omega):supp(B)\subseteq supp(B_G)} \mathcal{L}(\mathbf{X} : B, \Omega) + \lambda\|B\|_0, \tag{6.4}$$

where

$$\mathcal{L}(\mathbf{X} : B, \Omega) = -n \log(\det(I - B)) + \sum_{i=1}^{p} \frac{n}{2} \log(\sigma_i^2) + \frac{1}{2\sigma_i^2} \|\mathbf{X}_{\cdot,i} - \mathbf{X} B_{\cdot,i}\|_2^2 \tag{6.5}$$

is the negative log-likelihood function, and

$$\|B\|_0 := \sum_{i,j} \mathbb{I}_{x \neq 0}(B_{i,j})$$

counts the number of non-zero entries in the weighted adjacency matrix. Similar to the BIC score, set $\lambda = 0.5 \log n$.
In practice we do not need to compute the least squares estimate (6.5) for every variable $i \in [p]$, instead we can compute the sample covariance matrix beforehand and optimize (6.3) directly; compare *Algorithm 5, ComputeScore*.
Since we penalize over the number of edges we favour graphs with less edges, so the estimator (6.4) never outputs a reducible DG [Ghassami et al., 2020, Remark 2].

The structure $\mathcal{G}$ constrains the entries of the precision matrix Θ, refer to such constraints as *distributional constraints* of $\mathcal{G}$. So every distribution in $\Theta(\mathcal{G})$ should satisfy the distributional constraints of $\mathcal{G}$. Two structures are distribution equivalent, by *Definition 4.2*, if and only if they satisfy the same distributional constraints [Ghassami et al., 2020]. We have already seen in *Example 4.2* and the following discussion, that a graph, distribution equivalent to the complete DAG by definition, has another distribution constraint on the precision matrix (compare *Proposition 4.2*).
Moreover call a distributional constraint a *hard constraint* if the set of values satisfying that constraint is Lebesgue-measure zero over the space of values involved in the constraint. Denote the set of hard constraints by $H(\mathcal{G})$.

In structure learning we are given only one distribution generated from the ground truth and the task is to find the data-generating graph. However there might be a another structure that can also generate this distribution but is not distribution equivalent to the ground truth. So in general it is not possible to find the ground truth structure given only one distribution [Ghassami et al., 2020]. Therefore a weaker notion of equivalence is proposed.

Definition 6.1 *[Ghassami et al., 2020, Definition 9]. Let $\theta_{\mathcal{G}}$ be the set of linearly independent parameters needed to parametrize any distribution $\Theta \in \Theta(\mathcal{G})$. For DGs $\mathcal{G}_1$ and $\mathcal{G}_2$, let μ be the Lebesgue measure defined over $\theta_{\mathcal{G}_1} \cup \theta_{\mathcal{G}_2}$. Then $\mathcal{G}_1$ and $\mathcal{G}_2$ are* quasi equivalent*, denoted $\mathcal{G}_1 \cong \mathcal{G}_2$, if*

$$\mu(\theta_{\mathcal{G}_1} \cap \theta_{\mathcal{G}_2}) \neq 0.$$

So two structure are quasi equivalent if the set of distribution they can both generate has non-zero Lebesgue measure. It follows that two quasi equivalent structures share the same hard constraints [Ghassami et al., 2020]. The following assumption for structure learning is proposed, which generalizes Faithfulness:

Definition 6.2 *[Ghassami et al., 2020, Definition 10]. A distribution Θ is* generalized faithful *to $\mathcal{G}$ if Θ satisfies a hard constraint κ if and only if $\kappa \in H(\mathcal{G})$.*

Assumption 1 *The generated distribution is generalized faithful to the ground truth structure.*

The set of distributions not generalized faithful to $\mathcal{G}$ is measure zero, with respect to the Lebesgue measure over $\theta_{\mathcal{G}}$ [Ghassami et al., 2020, Proposition 8]. Consistency of the score (6.5) is then shown by the following theorem:

Theorem 6.1 *[Ghassami et al., 2020, Theorem 3]. Under assumption 1, the global minimizer of (6.5) with $\lambda = 0.5 \log n$ outputs $\hat{\mathcal{G}} \cong \mathcal{G}^*$.*

Block-Coordinate Descent Algorithm: For the computation of the MLEs we use the block-coordinate descent (BCD) algorithm proposed by [Drton et al., 2019]. This method iterates through all nodes and performs updates steps, where in the update for node i, the log-likelihood function with respect to all parameters corresponding to edges with an arrowhead at i are maximized, while holding all other parameters fixed. The updated parameters determine the i-th row of B and the i-th row and column of the matrix Ω. This work is an extension of the RICF algorithm [Drton et al., 2009] and is also guaranteed to produce a feasible positive definite covariance matrix after every iteration. The update steps preserve the structural zeros of the matrices B and Ω such that $(I-B)$ remains invertible, so the algorithm constructs a sequence in $\mathbf{B}(\mathcal{G}) \times \mathbf{\Omega}(\mathcal{G})$. Every accumulation point of this sequence is a critical point of the likelihood function and either a local maximum or a saddle point, where a local maximum can be checked by negative definiteness of the Hessian matrix of the log-likelihood function. Due to the possible multi-modality of the likelihood function there is no guarantee that the algorithm converges to a global maximum. It is also shown that for simple graphs, the BCD algorithm is well-defined [Drton et al., 2019, Theorem 3 & Proposition 4], where well-defined means that the new matrix Ω is positive definite. For certain models the algorithm may be ill-defined, however it is shown that these models are non-identifiable, meaning that one cannot uniquely recover the pair (B, Ω) from the covariance matrix (compare *Section 2.2.3* about parameter identification). For the R programming language the algorithm is available through the package BCD.

6.3 A score-based algorithm

We now present the score-based algorithm proposed by [Ghassami et al., 2020]. The algorithm start with an empty graph and performs one edge operation at every step. A *legal move* is an edge operation which can either add, remove or reverse an edge given the current support. For every legal move we compute the score of the likelihood function with the move applied, and take the one move (or several moves) that gives the highest decrease in score. Note that we minimize the scoring function.

Algorithm 5 Score-based search [Ghassami et al., 2020]

function COMPUTESCORE(ξ, X, S)
  $\hat{B}, \hat{\Omega} \leftarrow$ BCD(ξ, X) ▷ [Drton et al., 2019]
  $\Theta \leftarrow (I-B)\Omega^{-1}(I-B)^{\top}$
  return(trace($S\Theta$) - log(det(Θ)) + $\lambda\|B\|_0$)
end function

function FINDMOVE(ξ)
  $LegalMoves \leftarrow$ list all legal moves of current support ξ
  $BestScore \leftarrow \infty$
  $BestMove \leftarrow \emptyset$
  for $Move$ in $LegalMoves$ **do**
    if $Move$ is in Ψ **then** ▷ Hybrid method: check if edge is in PAG Ψ
      $\tilde{\xi} \leftarrow$ apply $Move$ to ξ
      $Score \leftarrow$ COMPUTESCORE($\tilde{\xi}, X, S$)
      if $Score < BestScore$ **then**
        $BestScore \leftarrow Score$
        $BestMove \leftarrow Move$
      end if
    end if
  end for
  return($BestMove, BestScore$)
end function

The function FINDMOVE first lists all legal moves we can make given the current support matrix, and then for every move, we compute the score of the support with the move applied. Here we use the function COMPUTESCORE, where the MLEs are computed via the block-coordinate descent algorithm from [Drton et al., 2019]. Then if the score gets improved, we assign the current move as best move. After considering every move the function outputs a *best move* and a *best score*, which is then applied to the current support.
This procedure is repeated until a convergence criterion is met, usually until there is no legal move such that the score can be further improved.

Based on this score-based algorithm we propose a new hybrid structure learning algorithm. Here we first compute the partial ancestral graph that represents the Markov equivalence class using the CCD algorithm from Thomas Richardson (compare *Algorithm 1*). Recall that the CCD algorithm starts with a complete graph and removes every edge between two variables i, j if there is a subset of variables $S \subset V \setminus \{i, j\}$ such that i and j are conditionally independent. Subsequently, the algorithm checks the unshielded triples whether they are unshielded conductors or unshielded non-conductors (*Theorem 3.2*), so CCD adds some ancestral relations. The PAG outputted from the CCD then restricts the search space for the score-based method. In practice we check for every legal move in the function FINDMOVE if it is restricted by the PAG.

Furthermore we utilize two optimization methods that minimize the scoring function, *hill climb-* and *beam-*search [Ghassami et al., 2020]. These are local search methods that greedily try to find the best graph structure. We start with an empty support matrix and perform the *move* that gives the highest decrease in score, given by the function FINDMOVE. In hill climb search we only consider the one best move while for beam search we take the k best moves, where k is called the *beam-width*. The algorithm then outputs k estimated structures, which we also call the *beam*.

Score decomposition. When a DG is acyclic, the distribution generated by a linear Gaussian structural equation model satisfies the local Markov property (*Definition 2.2*), which is in turn equivalent to the global Markov property for DAGs. From this it follows that the joint distribution of $\mathbf{X}$ can be factorized into the product of distributions of the variables conditioned on their parents, i.e.

$$f(\mathbf{X}) = \prod_{i \in V} f(X_i | \mathbf{Pa}(X_i)).$$

This reduces the computational complexity drastically since we only need to evaluate local changes which does not change the score of other parts of the DAG.

As also mentioned, for cyclic DGs the distribution may not satisfy the local Markov property (compare *Figure* **??**), but still satisfies the global Markov property if the error terms are joint independent [Spirtes, 1995]. Therefore the search procedure by [Ghassami et al., 2020] factorizes the joint distribution into a product of conditional distributions over the variables in a maximal strongly connected subgraph (MSCS), conditioned on their parents outside of the MSCS. Denote a MSCS by S.

$$f(\mathbf{X}) = \prod_{S_i \subseteq \mathbf{X}} f(S_i | \mathbf{Pa}(S_i)).$$

After an operation the likelihoods of the involved MSCS are updated. An operation can merge several MSCS into one bigger MSCS, or break a MSCS into several smaller MSCSs [Ghassami et al., 2020, Supplementary Material Section Q]. The MSCS can be computed using Tarjans algorithm [Tarjan, 1972].

6.4 Simulation Studies

We compare two types of algorithms, the score-based method proposed by [Ghassami et al., 2020] and a hybrid method which combines the score-based search with the CCD algorithm from [Richardson, 1996b]. We further investigate how the hybrid method behaves if we use use the true underlying graph as a d-separation oracle to compute the conditional independencies. In practice this would be replaced by a conditional independence test. Since we only consider Gaussian distributions this is done by the function GaussCItest from the PCALG package [Kalisch et al., 2012].
In the following we denote the score-based (unconstrained) algorithm by $\mathbf{SB}$, the hybrid method using the conditional independence test by $\mathbf{SB}+\mathbf{CCD}$, and the hybrid method using a d-separation oracle by $\mathbf{SB}+\mathbf{CCD}^*$.

The following algorithm will be used to generate the cyclic ground truth structures.

Algorithm 6 Random DCG generator.

function GENERATE DCG(p, c)
  initialize $supp \leftarrow 0^{p\times p}$ ▷ Start with empty support
  $s \leftarrow \frac{2(p-c)}{p(p-1)-2c}$ ▷ Sparsity

Put in cycle of length c.
  for i in $1 : (c-1)$ **do**
    $supp[i+1, i] \leftarrow 1$
  end for
  $supp[1, c] \leftarrow 1$

Fill in remaining edges.
  for j in $1 : (p-1)$ **do**
    for i in $(j+1) : p$ **do**
      if i and j not adjacent **then**
        $U \leftarrow$ Uniform$([0, 1])$
        $b \leftarrow$ Bernoulli$(\{0, 1\}, 0.5)$
        if $U < s$ **then**
          $supp[i, j] \leftarrow b$ and $supp[j, i] \leftarrow 1 - b$
        end if
      end if
    end for
  end for
  $supp \leftarrow$ permute rows and columns of $supp$
  return($supp$)
end function

Note that *Algorithm 6* only generates simple DCGs. The sparsity s ensures an expected neighborhood size of $E[N] = 2$, since

$$c + \left(\frac{p(p-1)}{2} - c\right) \cdot \frac{2(p-c)}{p(p-1)-2c} = p.$$

6.4.1 Simulation Settings

Using *Algorithm 6*, for every setting in the table below we generate 100 support matrices.

Dimension p	6	8	10	12	15
Cycle length c	4	5	5	8	8

The non-zero entries of the support matrix are then replaced by independent realizations of Uniform$([-0.8,-0.2]\cup[0.2,0.8])$ random variables, ensuring the edge weights are bounded away from zero, which results in the weighted adjacency matrix B.
Then for every B, datasets $\mathbf{X}$ with sample sizes $n\in\{100,500,2000,10000,50000\}$ are sampled from

$$\mathbf{X}=B^{\top}\mathbf{X}+\varepsilon,$$

where $\varepsilon_1,\dots,\varepsilon_p$ are mutually independent $\mathcal{N}(0,1)$ random variables. Then the variables $X_1,\dots,X_p$ have a multivariate Gaussian distribution with zero mean and covariance matrix $\Sigma=(I-B)^{-1}(I-B)^{-\top}$.

So in total we create

$$\underbrace{3}_{\text{Algorithm}}\cdot\underbrace{2}_{\text{Search}}\cdot\underbrace{5}_{\text{Sample size}}\cdot\underbrace{5}_{\text{Dimension}}\cdot 100=15000$$

datasets

6.4.2 Evaluation methods

To evaluate the success of a structure learning algorithm we need to establish a relationship between the learnt graph and the ground truth structure. There are three main types of scoring metrics. The first type is based on derivations of the confusion matrix like Precision and Recall, the second type of metrics is based on structural discrepancies like the structural hamming distance (SHD) which measures the number of edge operations needed to transform one graph into another, and the third type is based on inference methods which are for instance the BIC or AUC. [Constantinou, 2019] gives an overview of evaluation methods used in several structure learning algorithms.
Recall the entries of the confusion matrix:

- True positive (TP): Number of true edges present in the learnt graph.
- False positive (FP): Number of false edges present in the learnt graph.
- False negative (FN): Number of missing edges in the learnt graph.

To evaluate our results we then use the SHD, Recall (True positive rate) and Precision. These are defined as follows:

$$\text{Recall} = \frac{TP}{TP+FN}=\frac{TP}{P}=TPR.$$

$$\text{Precision} = \frac{TP}{TP+FP}=1-FDR.$$

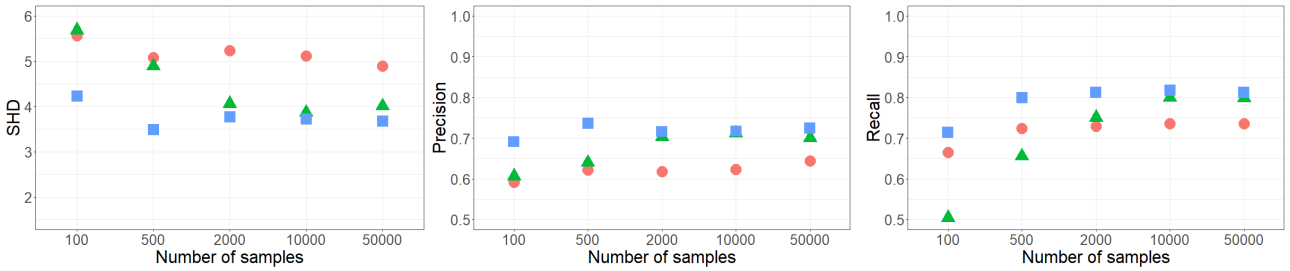
$$\text{SHD} = FP+FN.$$

Given the output $\hat{\mathcal{G}}$ of our learning algorithms, we compute the distribution equivalence class of the ground truth structure $\mathcal{G}^*$, and compare the estimated structure with every element in the DEC of $\mathcal{G}^*$ in terms of SHD, precision and recall. Then the element $\mathcal{G} \in [\mathcal{G}^*]_D$ which gives the lowest SHD serves as ground truth (as we cannot expect to find the data generating graph $\mathcal{G}^*$ when only relying on observational data). For beam search the algorithm outputs a set of length k of estimated structures. In this case we compare every $\hat{\mathcal{G}}_i,\ i \in [k]$ with every element of $[\mathcal{G}^*]_D$ to get the corresponding ground truth, compute the SHD, precision and recall, and then evaluate the estimated structure with the lowest SHD. So in general we use the SHD as main indicator for the success of the algorithms.
In the original paper by [Ghassami et al., 2020] the SHD is also used as a scoring metric. Due to the difficulties in discovering the true DEC given only one distribution, a multi-domain evaluation is additionally proposed:
Suppose Θ is the distribution generated by the ground truth $\mathcal{G}^*$, and let $\hat{\mathcal{G}}$ be the structure outputted by the algorithm. Due to possible violations of the generalized faithfulness assumption, $\hat{\mathcal{G}}$ may be able to maximize the likelihood but is not distribution equivalent to $\mathcal{G}^*$. The ansatz is as follows [Ghassami et al., 2020, Section 6]:

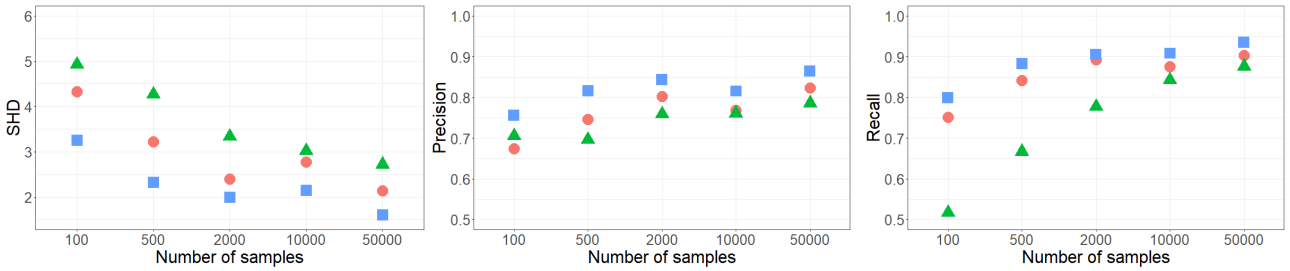
1. For the ground truth $\mathcal{G}^*$, generate d distributions $\{\Theta_1, \dots, \Theta_d\}$.
2. For each Θ_i, run the algorithm to obtain $\hat{\mathcal{G}}_i$.
3. For each $\hat{\mathcal{G}}_i$, optimize its edge weights and variances to generate distributions $\{\hat{\Theta}_{i,1}, \dots, \hat{\Theta}_{i,d}\}$ such that $\hat{\Theta}_{i,j}$ minimizes the KL-divergence to $\Theta_j \in \{\Theta_1, \dots, \Theta_d\}$.
4. The success rate of structure $\hat{\mathcal{G}}_i$ is the percentage of domains for which the minimizing KL-divergence in the previous step is below a threshold η.

The domain distributions are generated randomly, so if the success rate of $\hat{\mathcal{G}}_i$ is large, $\hat{\mathcal{G}}_i$ can actually generate a significant subset of the distributions set of $\mathcal{G}^*$. Therefore $\hat{\mathcal{G}}_i$ is quasi equivalent to $\mathcal{G}^*$ [Ghassami et al., 2020].
It is also mentioned that in some cases the SHD indicates that the estimated structure is not distribution equivalent to the ground truth, while the multi-domain approach shows high success rates.

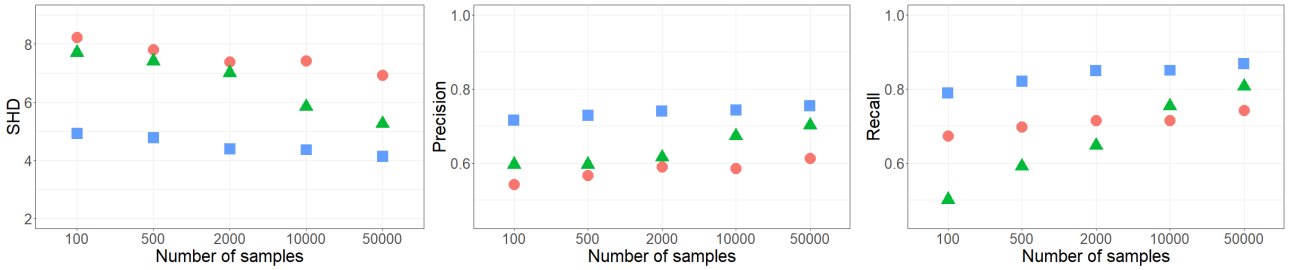
In the following the evaluation of the score-based and hybrid methods is presented. We divide between the number of nodes and show the evaluation of the algorithms in terms of SHD, precision and recall dependent on the sample size. Moreover we distinguish between hill-climb and beam search.
We will rather focus on the comparison of the presented algorithms instead of considering specific values. We will see that given the right search space, the hybrid methods can perform better than purely score-based methods. So the results presented in the following serve more as an incentive to use combined methods in learning the structure from data.



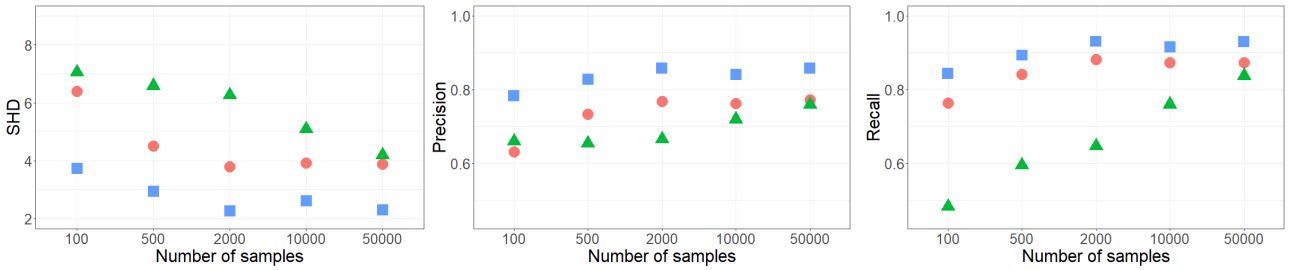
(a) Hill-climb search on $p = 6$ nodes.



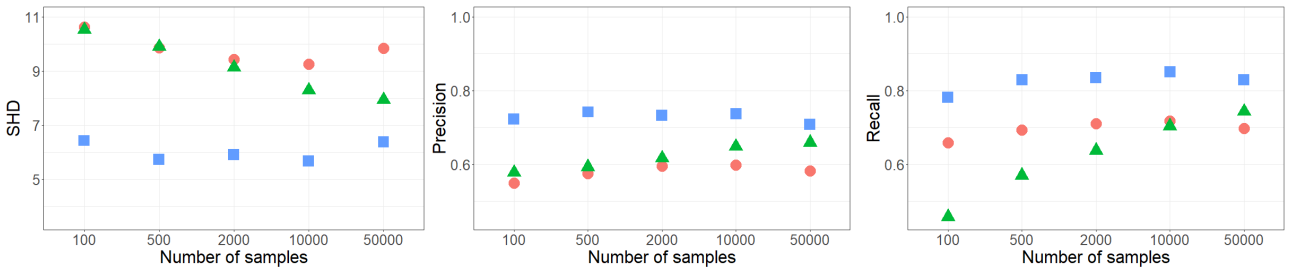
(b) Beam search on $p = 6$ nodes.



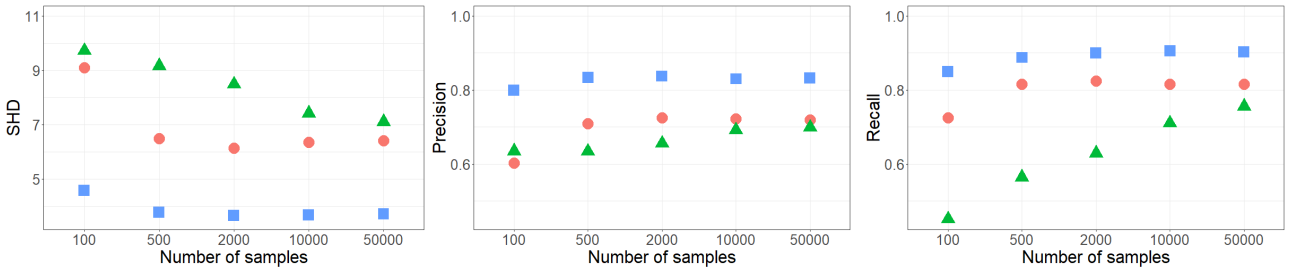
(c) Hill-climb search on $p = 8$ nodes.



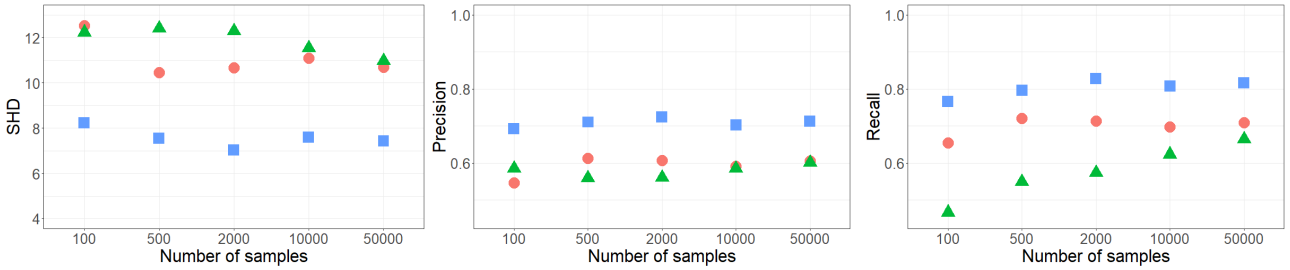
(d) Beam search on $p = 8$ nodes.



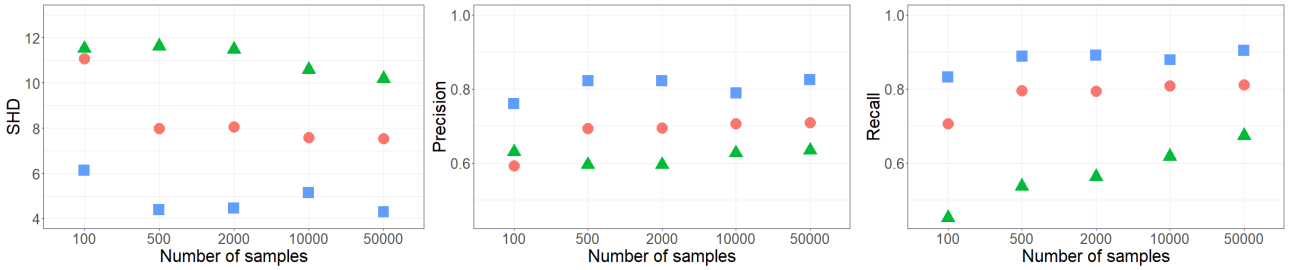
(e) Hill-climb search on $p = 10$ nodes.



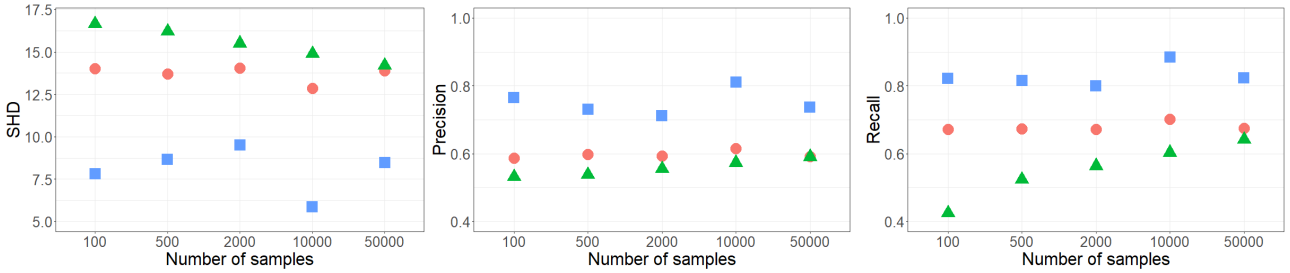
(f) Beam search on $p = 10$ nodes.



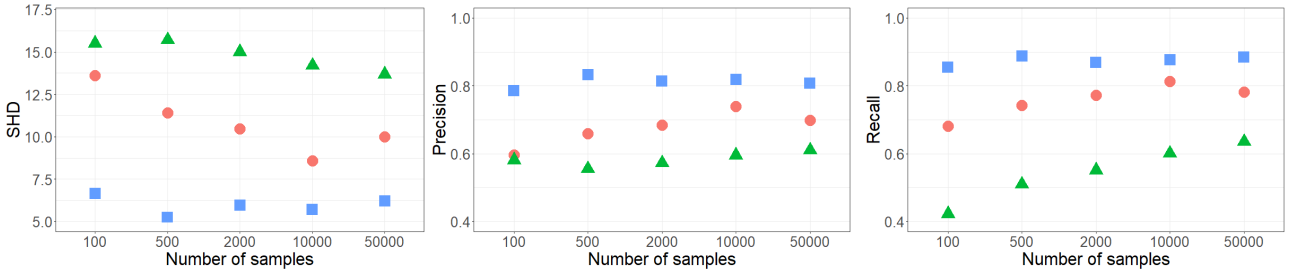
(g) Hill-climb search on $p = 12$ nodes.



(h) Beam search on $p = 12$ nodes.



(i) Hill-climb search on $p = 15$ nodes.



(j) Beam search on $p = 15$ nodes.

Figure 6.1 Evaluation of our score-based and hybrid methods subject to the number of samples.

Figure 6.1 shows the SHD, Precision and Recall of $\mathbf{SB}$ (red circles), $\mathbf{SB}+\mathbf{CCD}$ (green triangles) and $\mathbf{SB}+\mathbf{CCD}^*$ (blue squares) subject to the number of samples of the dataset $\mathbf{X}$. We distinguish between hill-climb search and beam search for a beam-width of $k = 3$.
We observe that $\mathbf{SB}+\mathbf{CCD}^*$ outperforms $\mathbf{SB}$ and $\mathbf{SB}+\mathbf{CCD}$ in almost every setting. This is no surprise since we have the ground truth structure at hand to compute the PAG. The other two methods rely solely on data as it is in practice. So the difference in the metrics between $\mathbf{SB}+\mathbf{CCD}^*$ and $\mathbf{SB}+\mathbf{CCD}$ is also an indicator how well we can estimate the conditional independence constraints in the CCD algorithm. We used a significance level of $\alpha = 0.005$ for the function GAUSSCITEST and it has to be further explored how this choice influences our results. (In the multivariate normal setting testing for conditional independence is equivalent to testing for vanishing partial correlation; see also [Drton and Perlman, 2007] for controlling error rates of incorrect edge inclusions in multiple testing procedures.) For the PC-algorithm a study for the choice of the significance level is performed, which yields a minimum SHD for values between $\alpha = 0.001$ and $\alpha = 0.005$ [Kalisch and Bühlmann, 2007, Section 4.2].

Comparing our two search methods, we see that beam search achieves lower values in SHD and higher precision and recall compared to hill-climb search. The reason for this is clear, we consider more structures through the search procedure. This depends mainly on the beam-width k. In our setting where we chose relatively small dimensions of our model, higher values of the beam-width k did not give noticeable better results. This might not be the case if we choose p in the hundreds or thousand, where the search space increases drastically. Increasing the beam-width is then also a tradeoff between more accurate algorithms and faster computation.

Observe also that for increasing sample sizes the algorithms clearly get more precise, but especially for $\mathbf{SB} + \mathbf{CCD}$ this is recognizable in terms of the recall (TPR). For a sample size of 100 we are only able to discovery half of the edges in the ground truth. For increasing sample size both, the CCD algorithm, and subsequent score-based search can make better estimates. Nevertheless the hybrid method $\mathbf{SB} + \mathbf{CCD}$ does not perform very well, and even worse than the purely score-based method $\mathbf{SB}$ for $p = 12$ and $p = 15$. For number of nodes up to 10 the hybrid method can reach similar results only if the sample size is large enough. Apparently if the conditional independence tests lead to a false constraint, then the score-based search can not recover from this 'mistake' and propagates this to more wrong statements. A solution might be to use a search space that is less restrictive, for example the conditional independence graph which is outputted by the CCD algorithm after the first step.

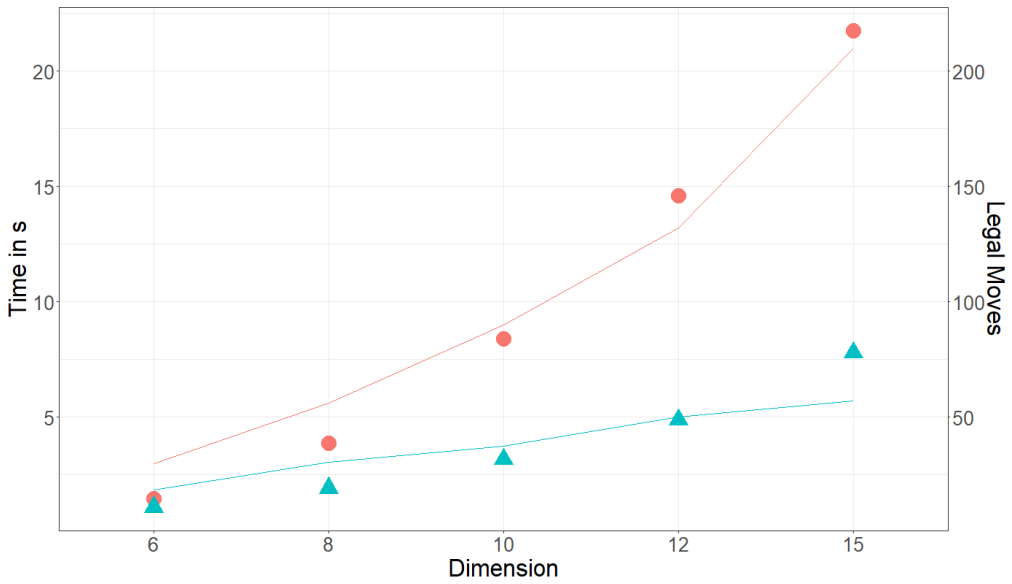


Figure 6.2 Time measurement for $\mathbf{SB}$ (circles) and $\mathbf{SB} + \mathbf{CCD}$ (triangles) in seconds for hill-climb search with a sample size of 100. The lines show the number of legal moves.

A major advantage from hybrid methods compared to the purely score-based methods is that the restriction of the search space leads to a much faster computation. Consider *Figure 6.2*, where the time in seconds needed to perform hill-climb search object to the dimension of our model for $\mathbf{SB}$ and $\mathbf{SB} + \mathbf{CCD}$ (with a sample size of 100) is depicted. The lines show the number of *legal moves* from which the search chooses the next edge. For the unconstrained method this behaves like $\mathcal{O}(p^2)$. Even though we have to compute the PAG for $\mathbf{SB} + \mathbf{CCD}$, this is compensated by a smaller search space and leads to an algorithm that is about three times faster for more than 10 vertices. Note that these two algorithms using hill-climb search achieve roughly the same evaluation in SHD.

6.4.3 Inconsistency of search methods

In hybrid algorithms the restriction of the search space of a score-based approach can lead to inconsistencies. In a work by [Nandy et al., 2018] the well known score-based algorithm Greedy equivalence search (GES) is restricted in its search space by first computing the conditional independence graph (CIG) and then only allowing additions if the edge is in the CIG. For DAGs we have the constraint of acyclicity, so GES might add a wrongly oriented edge and to reverse this edge one has to make sure that no cycles are created. In GES this is done by *covering* an edge, where the an edge $i \to j$ is covered if i and j have identical parents with the exception that i is not a parent of itself (this is the equivalent to a parent exchange). The problem arising for restricted GES is that an edge, needed for covering, may not be in the CIG, hence leading to inconsistency. This problem is then solved by adaptively restricting the search space of GES, called ARGES [Nandy et al., 2018]. For learning DCGs we do not have the constraint of acyclicity, moreover in our algorithm at every step we either add, remove or reverse an edge, where the reversal of an edge is equal to removing the edge followed by the addition of the reversed edge. So the same problem as for ARGES cannot occur here since every wrongly added edge can also be removed. Nevertheless there are search methods were inconsistencies can occur. Consider for example a forward-backward (FB) search method, similar to GES, where in a first procedure we only add edges until no improvement in score and then only removing edges until no improvement in score. Furthermore we restrict the search space by the output of the CCD algorithm, the PAG. This hybrid algorithm, is then similar to **SB-CCD** with the only difference that we cannot remove an edge at every step.

Figure 6.3 A weighted DCG with its corresponding PAG.

We use again our example graph from *Figure 2.1* with the edge weights given in *Figure 6.3*. Note that by the PAG we have the restriction that x and y are no descendants of a common child of a and b.

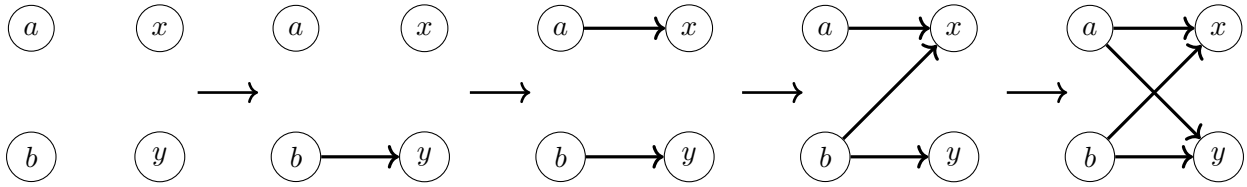


Figure 6.4 Trace of FB search with PAG restricted search space.

The trace is given in *Figure 6.4*. After the fourth step all edges between a, b and x, y are added such that the ancestor relations given in the PAG are preserved. Now x and y are common children of a and b, so the PAG forbids an edge between x and y as this would lead to one of them being a descendant of a common child of a and b. Hence, the forward phase ends here and the algorithm proceeds with the backward phase where edges are removed only.

7 Conclusion

In this work we considered the less studied class of directed graphs that can contain cycles. We reviewed the notion of equivalence based on conditional independence statements, that is, Markov equivalence. In the cyclic case the characterization of Markov equivalent structures is considerably more involved as one also has to check for long range separations. The cyclic equivalence theorem from [Richardson, 1996c] enables a polynomial-time algorithm for deciding Markov equivalence. Based on this theorem we also reviewed the CCD algorithm that aims to learn a representative of a Markov equivalence class [Richardson, 1996b].
A new notion of equivalence based on distributions a structure can generate has been presented, which was discovered recently by Amir Emad Ghassami. In this work three graphical transformations on a directed graph were presented. Regarding the class of simple graphs we presented conditions under which the transformations applied to a simple graph leads to another simple graph, or does not so. Still there are some open question in this setting regarding the existence of such a sequence of transformations. The algorithm to compute the distribution equivalence class, proposed by [Ghassami et al., 2020], was presented. This algorithm enabled us to depict how the elements of a distribution equivalence class are connected through the given transformations.
We utilized the algorithm for computing the distribution equivalence class of a given structure based on [Ghassami et al., 2020], and computed it of every directed graph on up to 5 nodes. To visualize and characterize the whole set of graphs we used the Markov equivalence class as a first criterion to organize the set of distribution equivalence classes. Here we computed the PAG for every Markov equivalence class with the CCD algorithm. Since CCD is only d-separation complete we completed the PAG by adding every ancestral relationship that can be made. We observed that some distribution equivalence classes can be distinguished based on its PAG, but there are also classes were the expressiveness of the PAG is not sufficient for representing a DEC, since it only focusses on representing Markov constraints and not the existence or absence of real and virtual adjacencies. Moreover some edges in a distribution equivalence class are essential which was not the case in the Markov equivalence class. To represent the distribution equivalence class uniquely, future work is necessary and more edge types have to be introduced.
Using the score-based approach for structure learning by [Ghassami et al., 2020], together with the CCD algorithm by [Richardson, 1996b], a hybrid algorithm was proposed. In this algorithm the PAG, outputted by CCD, restricted the search space such that the score-based approach can achieve better estimates in less computation time. Here two greedy search methods were used and the MLE estimates were computed using a block-coordinate descent method proposed by [Drton et al., 2019]. It has been shown that given the correct search space, the hybrid method can achieve better results than the purely score-based method.

Bibliography

[Ali et al., 2009] Ali, R. A., Richardson, T. S., and Spirtes, P. (2009). Markov equivalence for ancestral graphs. *The Annals of Statistics*, 37(5B).

[Amendola et al., 2020] Amendola, C., Dettling, P., Drton, M., Onori, F., and Wu, J. (2020). Structure learning for cyclic linear causal models. In Peters, J. and Sontag, D., editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 999–1008. PMLR.

[Andersson et al., 1997] Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505 – 541.

[Bollen, 1989] Bollen, K. A. (1989). *Structural Equations with Latent Variables*. John Wiley and Sons.

[Chickering, 2002] Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

[Constantinou, 2019] Constantinou, A. C. (2019). Evaluating structure learning algorithms with a balanced scoring function. *CoRR*.

[Darroch et al., 1980] Darroch, J. N., Lauritzen, S. L., and Speed, T. P. (1980). Markov Fields and Log-Linear Interaction Models for Contingency Tables. *The Annals of Statistics*, 8(3):522 – 539.

[Drton, 2009] Drton, M. (2009). Likelihood ratio tests and singularities. *The Annals of Statistics*, 37(2):979–1012.

[Drton, 2018] Drton, M. (2018). Algebraic problems in structural equation modeling. In *Advanced Studies in Pure Mathematics*. Mathematical Society of Japan.

[Drton et al., 2009] Drton, M., Eichler, M., and Richardson, T. (2009). Computing maximum likelihood estimates in recursive linear models with correlated errors. *Journal of Machine Learning Research*, 10:2329–2348.

[Drton et al., 2019] Drton, M., Fox, C., and Wang, Y. S. (2019). Computation of maximum likelihood estimates in cyclic structural equation models. *The Annals of Statistics*, 47(2):663 – 690.

[Drton et al., 2011] Drton, M., Foygel, R., and Sullivant, S. (2011). Global identifiability of linear structural equation models. *The Annals of Statistics*, 39(2):865–886.

[Drton and Maathuis, 2017] Drton, M. and Maathuis, M. H. (2017). Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4(1):365–393.

[Drton and Perlman, 2007] Drton, M. and Perlman, M. D. (2007). Multiple Testing and Error Control in Gaussian Graphical Model Selection. *Statistical Science*, 22(3):430 – 449.

[Drton and Yu, 2010] Drton, M. and Yu, J. (2010). On a parametrization of positive semidefinite matrices with zeros. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2665–2680.

[Edwards, 2000] Edwards, D. (2000). *Introduction to Graphical Modelling*. Springer New York, 2 edition.

[Forré and Mooij, 2017] Forré, P. and Mooij, J. (2017). Markov properties for graphical models with cycles and latent variables.

[Foygel et al., 2012] Foygel, R., Draisma, J., and Drton, M. (2012). Half-trek criterion for generic identifiability of linear structural equation models. *The Annals of Statistics*, 40(3).

[Ghassami et al., 2020] Ghassami, A., Yang, A., Kiyavash, N., and Zhang, K. (2020). Characterizing distribution equivalence and structure learning for cyclic and acyclic directed graphs. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3494–3504. PMLR.

[Glymour et al., 2019] Glymour, C., Zhang, K., and Spirtes, P. (2019). Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10.

[Golub and Van Loan, 1996] Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. The Johns Hopkins University Press, third edition.

[Heckerman and Geiger, 1995] Heckerman, D. and Geiger, D. (1995). Likelihoods and parameter priors for bayesian networks. *CoRR*, abs/2105.06241.

[Hyttinen et al., 2013] Hyttinen, A., Hoyer, P. O., Eberhardt, F., and Jarvisalo, M. (2013). Discovering cyclic causal models with latent variables: A general sat-based procedure.

[Kalisch and Bühlmann, 2007] Kalisch, M. and Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(22):613–636.

[Kalisch et al., 2012] Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., and Bühlmann, P. (2012). Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software*, 47(11):1–26.

[Lacerda et al., 2012] Lacerda, G., Spirtes, P., Ramsey, J., and Hoyer, P. (2012). Discovering cyclic causal models by independent components analysis. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*.

[Lauritzen, 1996] Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press.

[Lauritzen et al., 1990] Lauritzen, S. L., Dawid, A. P., Larsen, B. N., and Leimer, H.-G. (1990). Independence properties of directed markov fields. *Networks*, 20(5):491–505.

[Lauritzen and Spiegelhalter, 1988] Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224.

[Lauritzen and Wermuth, 1989] Lauritzen, S. L. and Wermuth, N. (1989). Graphical Models for Associations between Variables, some of which are Qualitative and some Quantitative. *The Annals of Statistics*, 17(1):31 – 57.

[M. Mooij and Claassen, 2020] M. Mooij, J. and Claassen, T. (2020). Constraint-based causal discovery using partial ancestral graphs in the presence of cycles. In Peters, J. and Sontag, D., editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1159–1168. PMLR.

[Maathuis et al., 2018] Maathuis, M., Drton, M., Lauritzen, S., and Wainwright, M. (2018). *Handbook of Graphical Models*. CRC Press, Inc., USA, 1st edition.

[Nandy et al., 2018] Nandy, P., Hauser, A., and Maathuis, M. H. (2018). High-dimensional consistency in score-based and hybrid structure learning. *The Annals of Statistics*, 46(6A):3151 – 3183.

[Ng et al., 2020] Ng, I., Ghassami, A., and Zhang, K. (2020). On the role of sparsity and dag constraints for learning linear dags.

[Park and Raskutti, 2016] Park, G. and Raskutti, G. (2016). Identifiability assumptions for directed graphical models with feedback. *CoRR*, abs/1602.04418.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Pearl, 2009] Pearl, J. (2009). *Causality*. Cambridge University Press, 2 edition.

[Raskutti and Uhler, 2013] Raskutti, G. and Uhler, C. (2013). Learning directed acyclic graphs based on sparsest permutations. *CoRR*, abs/1307.0366.

[Richardson, 1996a] Richardson, T. (1996a). Discovering cyclic causal structure. Technical report, Carnegie Mellon University.

[Richardson, 1996b] Richardson, T. (1996b). A discovery algorithm for directed cyclic graphs. In Horvitz, E. and Jensen, F. V., editors, *UAI '96: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, UAI'96, pages 454–461. Morgan Kaufmann.

[Richardson, 1996c] Richardson, T. (1996c). A polynomial-time algorithm for deciding equivalence of directed cyclic graphical models. In Horvitz, E. and Jensen, F. V., editors, *UAI '96: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, UAI'96, pages 462–469. Morgan Kaufmann.

[Richardson and Spirtes, 2002] Richardson, T. and Spirtes, P. (2002). Ancestral graph markov models. *The Annals of Statistics*, 30(4):962 – 1030.

[Shimizu et al., 2006] Shimizu, S., Hoyer, P. O., Hyvärinen, A., and Kerminen, A. (2006). A linear non-gaussian acyclic model for causal discovery. 7:2003–2030.

[Shpitser et al., 2014] Shpitser, I., Evans, R. J., Richardson, T. S., and Robins, J. M. (2014). Introduction to nested markov models. *Behaviormetrika*, 41:3–39.

[Spirtes, 1995] Spirtes, P. (1995). Directed cyclic graphical representations of feedback models. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, page 491–498, San Francisco, CA, USA. Morgan Kaufmann.

[Spirtes et al., 1993] Spirtes, P., Glymour, C., and Scheines, R. (1993). *Causation, Prediction, and Search*, volume 81.

[Strobl, 2019] Strobl, E. V. (2019). A constraint-based algorithm for causal discovery with cycles, latent variables and selection bias. *International Journal of Data Science and Analytics*, 8(1):33–56.

[Tarjan, 1972] Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160.

[Uhler, 2017] Uhler, C. (2017). Gaussian graphical models: An algebraic and geometric perspective. In Maathuis, M., Drton, M., Lauritzen, S., and Wainwright, M., editors, *Handbook of Graphical Models*, chapter 9, pages 219–240. CRC Press.

[Uhler et al., 2013] Uhler, C., Raskutti, G., Bühlmann, P., and Yu, B. (2013). Geometry of the faithfulness assumption in causal inference. *The Annals of Statistics*, 41(2).

[Verma and Pearl, 1990] Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, page 255–270, USA. Elsevier Science Inc.

[Whittaker, 1990] Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. Wiley Series in Probability and Statistics. Wiley.

[Wright, 1921] Wright, S. (1921). Correlation and causation. *Journal of Agricultural Research*, 20:557–585.

[Zhang, 2008a] Zhang, J. (2008a). Causal reasoning with ancestral graphs. *Journal of Machine Learning Research*, 9(47):1437–1474.

[Zhang, 2008b] Zhang, J. (2008b). On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16):1873–1896.

[Zhang, 2013] Zhang, J. (2013). A comparison of three occam's razors for markovian causal models. *British Journal for the Philosophy of Science*, 64(2):423–448.