Technical University of Munich

Department of Mathematics

Master's Thesis

Partial Homoscedasticity in Graphical Models

Marvin Sylejmani

Supervisor: Prof. Dr. Mathias Drton

Advisor: M.Sc. Jun Wu

Submission Date: 23.10.2023

I hereby declare that this thesis is my own work and that no other sources have been used except those clearly indicated and referenced.

Marvin Sylejmani

Munich, 23.10.2023

Acknowledgements

First, I would like to thank Professor Mathias Drton for giving me the great opportunity to write my thesis in the fascinating area of causal inference and for giving me interesting ideas and suggestions when needed. Further, I also want to thank Jun Wu for his amazing supervision, answering all kinds of questions, and engaging in deep discussions about the topics.

Finally, I want to thank my family, my girlfriend, and all my friends for supporting me during this intense time. Without them, I would not have made it.

Abstract

Graphical models associated with directed acyclic graphs (DAGs) are commonly used probabilistic models where the nodes represent random variables, and the edges encode conditional independence relations. These can also be thought of as structural equation models (SEMs), allowing for a natural causal interpretation. Linear Gaussian SEMs, one of the most frequently used models due to their computability, have been examined in prior work for the case of arbitrary error variances and the case of all error variances equal.

In this thesis, we study the case of groupwise equal error variances (Wu and Drton (2023)). As we will see, partial homoscedasticity of errors leads to the exhibition of algebraic constraints and, as a consequence, to specific model equivalence statements. The resulting equivalence classes can be represented by completed partially directed acyclic graphs (CPDAGs). In a simulation study, we show that integrating the learning of partition blocks in greedy search-based approaches can lead to strong results in estimating the CPDAG.

Zusammenfassung

Graphische Modelle, verknüpft mit gerichteten azyklischen Graphen (DAGs), sind häufig genutzte probabilistische Modelle, bei denen die Knoten Zufallsvariablen darstellen und die Kanten bedingte Unabhängigkeit kodieren. Diese Modelle können auch als Strukturgleichungsmodelle (SEMs) betrachtet werden, die eine natürliche kausale Interpretation ermöglichen. Lineare Gaußsche SEMs, die aufgrund ihrer Berechenbarkeit zu den am häufigsten verwendeten Modellen gehören, wurden in früheren Arbeiten für den Fall beliebiger Fehlervarianzen und für den Fall, dass alle Fehlervarianzen gleich sind, untersucht.

In dieser Arbeit untersuchen wir den Fall gruppenweise gleicher Fehlervarianzen (Wu and Drton (2023)). Wie wir sehen werden, führt die partielle Homoskedastizität der Fehler zum Auftreten algebraischer Bedingungen und infolgedessen zu spezifischen Aussagen über Modelläquivalenz. Die sich daraus ergebenden Äquivalenzklassen können durch vervollständigte, teilweise gerichtete azyklische Graphen (CPDAGs) dargestellt werden. In einer Simulationsstudie zeigen wir, dass die Integration des Lernens von Partitionsblöcken in auf Greedy-Algorithmen basierenden Ansätzen zu guten Ergebnissen bei der Schätzung des CPDAG führen kann.

Contents

1 Introduction

Recent breakthroughs in machine learning, especially in the domain of large language models (LLMs), have sparked big debates about their capabilities and, therefore, potential to replace humans. One of the major topics discussed is the ability of these models to reason and develop a causal understanding of the world (Kıcıman et al. (2023)). But what does causal understanding mean? Judea Pearl, one of the most influential scientists in the area of causal inference, describes the "three levels of causation" (Pearl and Mackenzie (2018)), a hierarchy of causal reasoning problems. The first level is "seeing" and "observing" (Associations), the second one is about "doing" (Interventions), and the highest level assumes "imagining" (Counterfactuals). His explanation of the causal hierarchy helps illustrate the kind of queries one can ask given a model and data.

Structural equation models (SEMs) provide a powerful mathematical framework for analyzing causal relationships between variables in both observational studies and experimental interventions (Pearl (2009), Spirtes et al. (2000)). Initially developed by Wright (1921) in the field of genetics using an approach based on path analysis, structural equation modeling has found applications in various disciplines. Haavelmo (1944) and Koopmans (1945) further developed structural models in econometrics, while Duncan (1975) introduced them in social sciences. The increasing computing power in the subsequent years enabled many scientists to apply structural equation models on large datasets.

Cause and effect in an SEM are modeled through functional relationships as specific assignment mechanisms, given by a different equation for each variable. Directed graphs are used to illustrate this, with every node corresponding to a variable and every directed edge representing the causal structure between variables. This thesis will focus on directed acyclic graphs (DAGs). A SEM can also be viewed as a directed graphical model, i.e., a probabilistic model expressing the conditional independence structure between random variables using a DAG. This characterization leads to criteria for deciding on observational equivalence of two models, i.e., when two different SEMs (and different DAGs) define the same statistical model.

1 Introduction

The challenge lies in learning the DAG from data (Structure Learning) since different SEMs can generate the same joint distribution. Depending on the type of SEM, we can make statements about identifiability. For the following classes of models, assuming causal minimality (Peters et al. (2017)), one can identify the underlying DAG from the joint distribution: Linear SEMs with non-Gaussian noise or LiNGAM (Shimizu et al. (2006)); non-linear additive noise models (Hoyer et al. (2008); Peters et al. (2011)); and linear SEMs with Gaussian noise that have equal variance (Peters and Bühlmann (2014)). In the case of linear Gaussian SEMs with arbitrary variance, one can learn the graph only up to the (Markov) equivalence class from observational data (Peters et al. (2017)).

This thesis will focus on a newly formulated type of linear Gaussian SEM, lying between the cases of arbitrary error variances and equal error variances. We look at the so-called partially homoscedastic linear Gaussian SEM, as described in Wu and Drton (2023). The set of variables is grouped into partition blocks, indicating equal error variance within the block and possibly different error variance between blocks. Trivially, the two extreme cases arising are the already known ones with arbitrary error variance (every variable represents one block) and full homoscedasticity of error variance (all variables in one block). In this novel setup, we will investigate the emergence of constraints due to the group-wise error variances and, as a result, new criteria of model equivalence. Structure learning becomes more challenging in this framework because, in addition to estimating graph structures, we also have to incorporate partitioning of the variable set into possible algorithms and methods. Using a greedy score-based approach, we design different schemes of learning partition and DAG.

The thesis is structured as follows: Chapter 2 briefly introduces graph theory and SEMs, focusing on the linear Gaussian case. Then, in Chapter 3, we describe partially homoscedastic linear Gaussian SEMs and dive into equal variance constraints, which lead to a formulation of model equivalence in this modified setup. In Chapter 4, we develop different ways of learning DAG and partition from data. In Chapter 5, we present the results of testing the described approaches on simulated datasets. Chapter 6 closes the thesis with a conclusion and brief discussion.

2 Preliminaries

In this chapter, we introduce necessary background and terminology about graphical structures and structural equation models. We first look at directed acyclic graphs (DAGs), which will be important when defining structural equation models (SEMs). As we will see, every model implies a graph, making DAGs a powerful tool to visualize causal structures between variables. We finish this chapter by stating important results about conditional independence and model equivalence in linear Gaussian SEMs.

2.1 Graph Theory

A directed graph is a tuple $G = (V, E)$, where V is the set of nodes or vertices and $E \subseteq V \times V$ is the set of edges. $(v,w) \in E$ encodes the edge $v \to w$, where v is called *tail* of edge and w *head* of edge. We will only consider directed graphs without self-loops, which means $E \cap \{(v, v) : v \in V\} = \emptyset$. Two vertices $v, w \in V$ are called *adjacent*, if $(v, w) \in E$ or $(w, v) \in E$. The subgraph induced by $A \subseteq V$ is $G_A = (A,\, E \cap (A \times A))$.

Paths

In a directed graph $G = (V, E)$ a *path* is an alternating sequence $\pi = (v_1, e_1, v_2, e_2, \ldots, v_{n-1}, e_{n-1}, v_n)$ of nodes from V and edges from E, such that $e_i = (v_i, v_{i+1})$ or $e_i = (v_{i+1}, v_i)$ for all $i = 1, \ldots, n-1$. In this definition, we allow the path to contain a node more than once. If $e_i = (v_i, v_{i+1})$ for all $i = 1, \ldots, n-1$, then π is a *directed path.*

A *trek* is a path of the form
$v_l^L \leftarrow v_{l-1}^L \leftarrow \cdots \leftarrow v_1^L \leftarrow v_{top} \rightarrow v_1^R \rightarrow \cdots \rightarrow v_{r-1}^R \rightarrow v_r^R$

where the left side of the trek is a directed path from v_{top} to v_l^L and the right side is a directed path from v_{top} to v_r^R. The node v_{top} is called *top node.*

Relations among vertices

In an edge $v \to w$, we call v a *parent* of w and w a *child* of v. We can denote the sets of parents and children of a node i by $\mathrm{pa}(i)$ and $\mathrm{ch}(i)$, respectively. Additionally we define *ancestors* and *descendants* of i in G by

$$\mathrm{an}(i) := \{k \in V : \exists k \to \cdots \to i \text{ in } G\} \text{ and } \mathrm{de}(i) := \{k \in V : \exists i \to \cdots \to k \text{ in } G\},$$

where we use the convention, that $i \notin \mathrm{an}(i)$ and $i \in \mathrm{de}(i)$. We say that G contains a *cycle* if a pair (j, k) exists with directed paths from j to k and k to j. A graph without a cycle is called a *directed acyclic graph* (DAG). The *skeleton* of a DAG is the undirected graph formed by removing directions of all the edges in the DAG.

Topological ordering becomes crucial when using DAGs to describe data-generating procedures in structural equations.

Definition 2.1. Let $G = (V, E)$ be a DAG with $|V| = p$. A bijective mapping $\delta : \{1, \ldots, p\} \to \{1, \ldots, p\}$ is called *topological ordering of* G, if for all $i, j \in V$ we have that $i \in \mathrm{an}(j)$ implies $\delta(i) < \delta(j)$.

The ordering is such that for $(i, j) \in E$, it follows that i appears before j in the ordering. We know from Peters et al. (2017) that for every DAG, there exists a (non-necessarily unique) topological ordering. Topological ordering can be naturally extended to a 2-tuple of nodes.

d-Separation

In a directed graph $G = (V, E)$ we call a triplet of vertices (i, j, k) a *collider triple*, if $i \to k$ and $j \to k$. We call k a *collider* and distinguish between an *unshielded collider* or *immorality* if i and j are non-adjacent, and a *shielded collider* otherwise. Let $\pi = (v_1, e_1, v_2, e_2, \ldots, v_{n-1}, e_{n-1}, v_n)$ be a path. We call v_i with $i \in \{2, \ldots, n-1\}$ a *collider relative to this path* if $v_{i-1} \to v_i$ and $v_{i+1} \to v_i$ are edges in π.

Closely related to the notion of colliders is the construction of directional separation criteria in DAGs.

Definition 2.2. Two nodes v,w in a DAG $G = (V, E)$ are *d-connected given* $C \subseteq V \setminus \{v, w\}$, if there exists a path π from v to w, for which the following conditions hold :

i) every collider on π is in C, and

ii) every non-collider on π is not in C.

If no path between v and w satisfies these conditions, we say that v and w are *d-seperated given* C. We then write $v \perp\!\!\!\perp_G w \mid C$. Let $A, B, C \subseteq V$ be pairwise disjoint sets. We then write $A \perp\!\!\!\perp_G B \mid C$, if there are no vertices $a \in A$ and $b \in B$ such that a and b are d-connected given C.

A trek, for example, is a path without a collider. Hence, its endpoints are d-connected given $\emptyset$. To build the bridge between DAGs and conditional independence statements, we will look at the *global Markov property* of joint distributions.

Definition 2.3 (Global Markov property)**.** Let $G = (V, E)$ be a DAG with $V = \{1, \ldots, p\}$ and $X = (X_1, \ldots, X_p)$ a multivariate random variable with joint distribution P^X. Let $A, B, C \subseteq V$ be pairwise disjoint sets.
We say that P^X satisfies the *global Markov property* or is *Markovian* with respect to G, if

$$A \perp\!\!\!\perp_G B \mid C \Rightarrow A \perp\!\!\!\perp B \mid C,$$

where $A \perp\!\!\!\perp B \mid C$ means that A is conditionally independent of B given C.

We also define the following converse.

Definition 2.4. Same setup as in Def 2.3.
We say that P^X is *faithful* with respect to G, if

$$A \perp\!\!\!\perp B \mid C \Rightarrow A \perp\!\!\!\perp_G B \mid C.$$

In summary, if a distribution is Markovian to G, we can read conditional independence statements from the graph. If additionally it is faithful, all conditional independence statements are encoded in G.

2.2 Structural Equation Models

We now look at the general definition of structural equation models before studying our special case of interest. The main part of this section adopts Peters et al. (2017) and Drton (2018).

Definition 2.5 (Structural equation model). A *structural equation model* (SEM) $\mathfrak{C} = (S, P^{\epsilon})$ for random vector $X = (X_1, \dots, X_p)$ is a collection S of p structural assignments

$$X_i := f_i(X_{\text{pa}(i)}, \epsilon_i), \quad i = 1, \dots, p,$$

where $\text{pa}(i) \subseteq \{X_1, \dots, X_p\} \setminus \{X_i\}$ are called *parents* of X_i, and a joint distribution P^{ϵ} over the noise variables $\epsilon = (\epsilon_1, \dots, \epsilon_p) \sim P^{\epsilon}$. We require the noise variables to be jointly independent; P^{ϵ} is a product distribution.
The directed graph of this SEM is obtained by creating one vertex for each variable X_i and drawing directed edges from each element of $\text{pa}(i)$ to X_i for all $i = 1, \dots, p$. From now on, we assume the graph to be a DAG. Within the framework of causal modeling, we call the elements of $\text{pa}(i)$ *direct causes* of X_i, and X_i a *direct effect* of its direct causes.

This thesis will concentrate on an important case of structural equation models.

Definition 2.6. A *linear structural equation model* (linear SEM) over p-dimensional random vector $X = (X_1, \dots, X_p)$ is of the form

$$X_i := \sum_{k \in \text{pa}(i)} \lambda_{ki} X_k + \varepsilon_i, \quad i = 1, \dots, p$$

or in matrix notation

$$X = \Lambda^T X + \epsilon,$$

where $\Lambda := (\lambda_{ij}) \in \mathbb{R}^{p \times p}$ is the matrix of coefficients, which represents the causal structure among the variables. $\epsilon = (\epsilon_1, \dots, \epsilon_p)$ is the random vector modeling the error terms with covariance matrix $Var[\epsilon] := \Omega$. Since we assume independent errors, $\Omega = \text{diag}(w_1, \dots, w_p) \in \mathbb{R}^{p \times p}$ is a diagonal matrix containing the positive noise variances. The focus will be on covariance matrices; therefore w.l.o.g, we can assume that $E[\epsilon] = 0$.

Let I be the identity matrix. If $I - \Lambda$ is invertible, then the linear equation system is uniquely solved by $X = (I - \Lambda)^{-T} \epsilon$ and

$$Var[X] = (I - \Lambda)^{-T} Var[\epsilon] (I - \Lambda)^{-1} = (I - \Lambda)^{-T} \Omega (I - \Lambda)^{-1}.$$

Consequently, we can encode the structure of a DAG in a linear SEM by setting all entries λ_{ij} of the coefficient matrix Λ to zero whenever the corresponding edge $i \to j$ is missing in the graph (see Figure 2.1)

$$\begin{aligned} X_1 &= \epsilon_1 \\ X_2 &= \lambda_{12} X_1 + \epsilon_2 \\ X_3 &= \lambda_{13} X_1 + \lambda_{23} X_2 + \epsilon_3 \end{aligned}$$

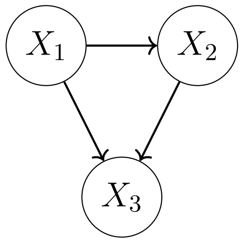


Figure 2.1: Example of a linear SEM and corresponding DAG

Let $G = (V, E)$ be a DAG with $|V| = p$. We then define by

$$\mathbb{R}^E = \{\Lambda = (\lambda_{ij}) \in \mathbb{R}^{p \times p} : \lambda_{ij} = 0, \text{if } i \to j \notin E\}$$

the set of $p \times p$ matrices with support in E. Combined with the error variances, we can now parametrize the covariance matrix of X by following map

$$\phi_G : \mathbb{R}^E \times (0, \infty)^p \mapsto PD, \ \ (\Lambda, w) \mapsto (I - \Lambda)^{-T} \text{diag}(w)(I - \Lambda)^{-1},$$

where $w = (w_1, \dots, w_p)$ is the vector of error variances, and PD is the cone of positive definite matrices. As a result, $I - \Lambda$ is invertible for all $\Lambda \in \mathbb{R}^E$, since G is a DAG and hence Λ can be permuted into lower-triangular form.

By taking the error vector to follow a Gaussian distribution and looking at the fact that the linear transformation of a Gaussian random vector is again Gaussian; we can define the following special case, which we will focus on for the rest of the thesis.

Definition 2.7. Let $G = (V, E)$ be a DAG with $|V| = p$. The *linear Gaussian SEM* (or *directed Gaussian graphical model*) given by G is the family of all multivariate normal distributions in $\mathbb{R}^p$ with covariance matrices in the following set:

$$M_G = \{\Sigma : \Sigma = \phi_G(\Lambda, w), \Lambda \in \mathbb{R}^E, w \in (0, \infty)^p\}.$$

We can calculate the covariances between individual entries of X with the result known as *trek rule* , using the parametrization of the covariance matrix by Λ and w, see Drton (2018)[Theorem 4.1].

Theorem 2.8 (Trek rule). *Let $G = (V, E)$ be a DAG with $|V| = p$ and let $\Lambda \in \mathbb{R}^E, w \in (0, \infty)^p$. For $i, j \in V$, let $\mathcal{T}(i, j)$ be the set of all treks between i and j. We define for a trek τ with top node i_0 the trek monomial*

$$\tau(\Lambda, w) = w_{i_0} \prod_{k \to l \in \tau} \lambda_{kl}.$$

The covariance between X_i and X_j is the sum of all trek monomials between i and j

$$\phi_G(\Lambda, w)_{ij} = \sum_{\tau \in \mathcal{T}(i,j)} \tau(\Lambda, w), \quad i, j \in V.$$

Before we look closer at conditional independence statements in linear Gaussian SEMS, we first state an important result about conditional independence in Gaussian distributions. More details in Drton et al. (2009).

Theorem 2.9. *Let $X = (X_1, \ldots, X_p)$ be a Gaussian random vector with covariance matrix Σ. Let $i, j \in \{1, \ldots, p\}$ be two distinct indices, and let $C \subseteq \{1, \ldots, p\} \setminus \{i, j\}$. Then we have $i \perp\!\!\!\perp j \mid C$ if and only if $\det(\Sigma_{i \cup C, j \cup C}) = 0$, i.e., the corresponding minors of the covariance matrix vanish.*

Linear SEMs entail conditional independence statements, which can be read off the corresponding DAG using d-separation criteria. We summarize these facts in following theorem, see Spirtes (2013) and Richardson and Spirtes (2002) for further discussion.

Theorem 2.10. *Let $G = (V, E)$ be a DAG. Let i, j be two distinct nodes, and let $C \subseteq V \setminus \{i, j\}$.*

i) The joint distribution P^X of a linear SEM corresponding to the DAG G satisfies the global Markov property with respect to G.

ii) A matrix $\Sigma \in PD$ is in M_G if and only if $\det(\Sigma_{i \cup C, j \cup C}) = 0$ for all triples (i, j, C) with $i \perp\!\!\!\perp_G j \mid C$.

Different DAGs can encode the same set of conditional independence; they define the same statistical model. This notion is called *Markov equivalence.*

Definition 2.11 (Markov equivalence)**.** Two DAGs G_1 and G_2 are *Markov equivalent*, if $M_{G_1} = M_{G_2}$.

Verma and Pearl (1990) provide a compact characterization of Markov equivalent graphs, which makes it simple to determine equivalence:

Lemma 2.12. *Two DAGs G_1 and G_2 are Markov equivalent if and only if they have the same skeleton and unshielded colliders.*

Figure 2.2 shows an example of Markov equivalent DAGs. DAGs who are Markov equivalent entail the same d-separation statements. They form an equivalence class and can be represented by a completed partially directed acyclic graph (CPDAG), more on that in Section 3.4.

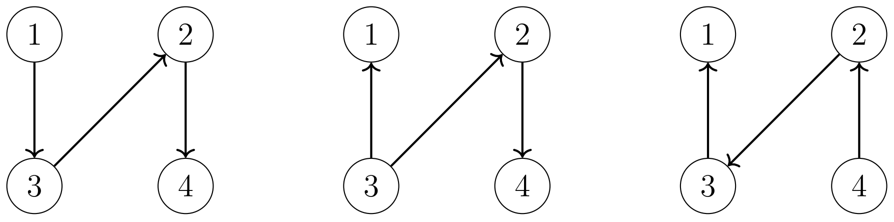


Figure 2.2: Example of Markov equivalent DAGs.

3 Partial Homoscedasticity in Linear Gaussian SEM

In this chapter, we introduce partially homoscedastic linear Gaussian SEM and develop algebraic constraints for that model. We then fully characterize this new class of models and investigate model equivalence in this setup. Section 3.1-Section 3.4 closely adopt Wu and Drton (2023), including all proofs (except for Theorem 3.13). In the final section, we look at examples of the new model equivalence classes and compare them to classic Markov equivalence classes.

3.1 Definition

In order to define our new class of models, we first need to establish the concept of partitioning the vertex set in a DAG.

Definition 3.1. Let $G = (V, E)$ be a DAG. We say that $\Pi = \{\pi_1, \ldots, \pi_k\}$ forms a *partition* of V, if $\pi_1, \ldots, \pi_k$ are non-empty and pairwise disjoint subsets of V such that $\dot{\cup}_{l=1}^{k} \pi_l = V$.

We can define an equivalence relation on V by setting $i \sim j$, whenever i and j are in the same *block* π_l and will write $i \sim_\Pi j$ for the rest of the thesis to indicate that the two vertices i and j are in the same block of the partition Π. Our main idea now is to incorporate a priori assumptions about the error variances of the linear Gaussian SEM using the partition of the vertex set. We have equal error variances within partition blocks and possibly different error variances between partition blocks.

Definition 3.2. Let $G = (V, E)$ be a DAG with $|V| = p$, and let Π be a partition of V. The *partially homoscedastic linear Gaussian SEM* given by G and Π is the family of all multivariate normal distributions in $\mathbb{R}^p$ with covariance matrices in the following set:

$$M_{G,\Pi} = \{\Sigma : \Sigma = \phi_G(\Lambda, w), \Lambda \in \mathbb{R}^E, w \in (0, \infty)^p \text{ with } w_i = w_j \text{ if } \; i \sim_\pi j\}.$$

$$
\begin{aligned}
X_1 &= \epsilon_1, \quad \epsilon_1 \sim \mathcal{N}(0, w_1^2) \\
X_2 &= \lambda_{12} X_1 + \epsilon_2, \quad \epsilon_2 \sim \mathcal{N}(0, w_2^2) \\
X_3 &= \lambda_{13} X_1 + \lambda_{23} X_2 + \epsilon_3, \quad \epsilon_3 \sim \mathcal{N}(0, w_2^2)
\end{aligned}
$$

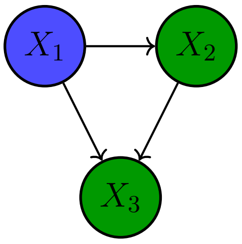


Figure 3.1: Example of a partially homoscedastic linear Gaussian SEM with two blocks and the corresponding DAG

This partially homoscedastic setup includes the standard heteroscedastic case with arbitrary variances by setting $\Pi = \{\{1\}, \{2\}, \dots, \{p\}\}$ ($M_{G,\Pi} = M_G$) and the fully homoscedastic case (Peters and Bühlmann (2014)) with all error variances being equal by setting $\Pi = \{\{1, 2, \dots, p\}\}$.
Figure 3.1 shows an example of a partially homoscedastic linear Gaussian SEM. As we will see later, this new model setup leads to a refinement of the standard Markov equivalence classes. But before we look closer at model equivalence, we first dive into algebraic constraints, which will help us make statements about the identifiability of our model.

3.2 Equal Variance Constraints

Unlike the standard case of arbitrary error variances, algebraic constraints arise in the partially homoscedastic case. In order to describe them, we first look at how to obtain the error variances from the covariance matrix.

Theorem 3.3. *Let $G = (V, E)$ be a DAG with $|V| = p$, and let $\Sigma = \phi_G(\Lambda, w)$ for $\Lambda \in \mathbb{R}^E$ and $w \in (0, \infty)^p$. Then for any $i \in V$, the error variance w_i can be calculated from the covariance matrix Σ as*

$$w_i = \sigma_i - \Sigma_{i,A}(\Sigma_{A,A})^{-1}\Sigma_{A,i}, \tag{3.1}$$

where $\sigma_i := \Sigma_{i,i}$ denotes the variance of $i \in V$ and A may be any subset such $\mathrm{pa}(i) \subseteq A \subseteq V \setminus \mathrm{de}(i)$.

Proof. We adapt the proof of Drton (2018)[Theorem 7.1], where $A = \mathrm{pa}(i)$. Let $a \in A$, $i \in V$. We now look at all possible treks between a and i. If there is a trek between a and i, which ends at i with an edge of the form $k \leftarrow i$, then the trek looks the following: $a \leftarrow \cdots \leftarrow k \leftarrow i$. This would mean that a is a descendant of i.

But that is not possible since $A \subseteq V \setminus \mathrm{de}(i)$. So every trek between a and i must end with the edge $k \to i$. Using the trek rule (Theorem 2.8), we can now conclude

$$\Sigma_{a,i} = \sum_{l \in \mathrm{pa}(i)} \Sigma_{a,l} \Lambda_{l,i}.$$

It follows immediately that

$$\Sigma_{A,i} = \Sigma_{A,\mathrm{pa}(i)} \Lambda_{\mathrm{pa}(i),i} = \Sigma_{A,A} \Lambda_{A,i}, \tag{3.2}$$

where the second equality comes from the fact that $\mathrm{pa}(i) \subseteq A$ and $\Lambda_{k,i} = 0$ for $k \notin \mathrm{pa}(i)$. We now look at treks between i and i. Since G is a DAG, the trek $i \to \cdots \to i$ is not possible. Using the trek rule again, we get

$$\sigma_i = w_i + \sum_{r \in \mathrm{pa}(i)} \Lambda_{r,i} \sum_{l \in \mathrm{pa}(i)} \Sigma_{r,l} \Lambda_{l,i},$$

where w_i is the trek monomial of the trivial trek, which contains only the vertex i. Converting to matrix notation and using the zeros in $\Lambda_{A,i}$, we get

$$\sigma_i = w_i + \Lambda^T_{\mathrm{pa}(i),i} \Sigma_{\mathrm{pa}(i),\mathrm{pa}(i)} \Lambda_{\mathrm{pa}(i),i} = w_i + \Lambda^T_{A,i} \Sigma_{A,A} \Lambda_{A,i}.$$

Rearranging this term and inserting (3.2), we finish the proof. □

Consequently, we get the following result about equal variance assumption.

Corollary 3.4. *Let ϵ_i, ϵ_j be two random Gaussian errors with equal variance, i.e., i and j are in the same block of a partition Π. Then all covariance matrices in $M_{G,\Pi}$ satisfy that*

$$\sigma_i - \Sigma_{i,A_i} (\Sigma_{A_i,A_i})^{-1} \Sigma_{A_i,i} = \sigma_j - \Sigma_{j,A_j} (\Sigma_{A_j,A_j})^{-1} \Sigma_{A_j,j} \tag{3.3}$$

for all subsets A_i and A_j such that $\mathrm{pa}(i) \subseteq A_i \subseteq V \setminus \mathrm{de}(i)$ and $\mathrm{pa}(j) \subseteq A_j \subseteq V \setminus \mathrm{de}(j)$.

The following theorem shows the converse.

Theorem 3.5. *Let $G = (V, E)$ be a DAG with $|V| = p$ and let $i \in V$. Let $A \subseteq V \setminus \{i\}$. Fix any vector of positive error variances $w \in (0, \infty)^p$. If for all $\Lambda \in \mathbb{R}^E$ the covariance matrix $\Sigma = \phi_G(\Lambda, w)$ satisfies equation (3.1), then it must hold that $\mathrm{pa}(i) \subseteq A \subseteq V \setminus \mathrm{de}(i)$.*

Proof. Let us assume that there exists a node $k \in \mathrm{pa}(i) \setminus A$. We now choose Λ so that all entries are zero, except for λ_{ki}. By the trek rule, we get $\Sigma_{i,A} = 0$, since $k \notin A$. Using that together with (3.1) we conclude that $w_i = \sigma_i$.

But looking at the fact that the only treks between i and i are the trivial one and $i \leftarrow k \rightarrow i$, we consequently get by the trek rule $\sigma_i = w_i + \lambda_{ki}^2 w_k > w_i$, which is a contradiction to (3.1).
Now we will assume that there exists a node $k \in A \setminus (V \setminus \mathrm{de}(i)) = \mathrm{de}(i) \cap A$. This means that k is a descendant of i. Hence there exists a directed path $i \rightarrow \cdots \rightarrow k$. We can always pick k as this path's first node in A. Therefore the path has the form $i \rightarrow m_1 \rightarrow \cdots \rightarrow m_t \rightarrow k$ with $m_1, m_2, \ldots, m_t \notin A$. We now choose Λ such that all entries are zero except for $\lambda_{im1}, \lambda_{m_1 m_2}, \ldots, \lambda_{m_{t-1} m_t}, \lambda_{m_t k}$. By the trek rule, we get $\sigma_i = w_i$ since the only non-zero trek (no edge with zero weight) between i and i is the trivial one. Also, by the trek rule, we get that

$$\Sigma_{i,k} = w_i \lambda_{im_1} \lambda_{m_t k} \prod_{i=2}^{t} \lambda_{m_{s-1} m_s}.$$

Inserting that in (3.1) and taking into account all the zeros in Σ we get

$$\begin{aligned} w_i &= \sigma_i - \left(w_i \lambda_{im_1} \lambda_{m_t k} \prod_{i=2}^{t} \lambda_{m_{s-1} m_s} \right)^2 [(\Sigma_{A,A})^{-1}]_{kk} \\ &= \sigma_i - \left(w_i \lambda_{im_1} \lambda_{m_t k} \prod_{i=2}^{t} \lambda_{m_{s-1} m_s} \right)^2 \frac{1}{\sigma_k} < \sigma_i = w_i, \end{aligned}$$

which is a contradiction. This concludes the proof. □

We can now combine Theorem 3.3 and Theorem 3.5 to characterize the equal variance constraints which hold in a partially homoscedastic model.

Theorem 3.6. *Let $G = (V, E)$ be a DAG, and let Π be a partition of V. Let $i, j \in V$ bet two nodes such that $i \sim_\Pi j$, i.e., they are in the same partition block, and let $A_i \subseteq V \setminus \{i\}$ and $A_j \subseteq V \setminus \{j\}$. Then the equation (3.3) holds for all matrices $\Sigma \in M_{G,\Pi}$ if and only if* $\mathrm{pa}(i) \subseteq A_i \subseteq V \setminus \mathrm{de}(i)$ *and* $\mathrm{pa}(j) \subseteq A_j \subseteq V \setminus \mathrm{de}(j)$.

Proof. The "if" direction follows directly from Corollary 3.4. In the "only if" direction, we will distinguish different cases for the set A_i using proof by contradiction similar to the proof of Theorem 3.5. The proof for the set A_j is analogous.

a) $\exists\, k \in \mathrm{pa}(i) \setminus A_i$: We choose Λ such that all entries are zero, except for λ_{ki}. Using the trek rule we get $\Sigma_{i,A_i} = 0$, since $k \notin A_i$. Therefore (3.3) yields :

$$\sigma_i = \sigma_j - \Sigma_{j,A_j}(\Sigma_{A_j,A_j})^{-1}\Sigma_{A_j,j} \leq \sigma_j.$$

By the trek rule we also get $\sigma_j = w_j$ and $\sigma_i = w_i + \lambda_{ki}^2 w_k > w_i$, which follows from the fact that the only non-zero treks between i and i are the trivial one and $i \leftarrow k \rightarrow i$. It follows that

$$\sigma_i \leq \sigma_j = w_j = w_i < w_i + \lambda_{ki}^2 w_k = \sigma_i,$$

which is a contradiction. We conclude that $\mathrm{pa}(i) \subseteq A_i$.

b) $\exists\, k \in \mathrm{de}(i) \cap A_i$: We deduce that there exists a directed path $i \rightarrow \cdots \rightarrow k$. Analogous to the proof of Theorem 3.5 we will choose k such that the path is "minimal", which means $i \rightarrow m_1 \rightarrow \cdots \rightarrow m_t \rightarrow k$ with $m_1, m_2, \ldots, m_t \notin A_i$. We continue by distinguishing three subcases (illustrated in Figure 3.2) :

   i) We first look at the case where the minimal path from i to k does not intersect j. Similar to the proof of Theorem 3.6, we set all edge weights in Λ to zero, except for those in the path. The trek rule asserts that $\sigma_i = w_i = w_j = \sigma_j$ and

$$\Sigma_{i,k} = w_i \lambda_{im_1} \lambda_{m_t k} \prod_{i=2}^{t} \lambda_{m_{s-1}m_s}.$$

   Consequently we get under equation (3.3)

$$w_i = \sigma_i - \left(w_i \lambda_{im_1} \lambda_{m_t k} \prod_{i=2}^{t} \lambda_{m_{s-1}m_s}\right)^2 \frac{1}{\sigma_k} < \sigma_j = w_j,$$

   which is a contradiction.

   ii) Now we consider the case where every minimal path from i to k contains j and additionally $A_j \cap \mathrm{de}(j) \neq \emptyset$. Let $k' \in A_j \cap \mathrm{de}(j)$. That means we have a directed path from j to $k' \in A_j$. The node i is not part of this path. Otherwise, we would have a directed cycle from i to i, which contradicts G being a DAG. We can see that we have the analogous case of subcase i), with i and j switched, and hence can construct a contradiction to equation (3.3).

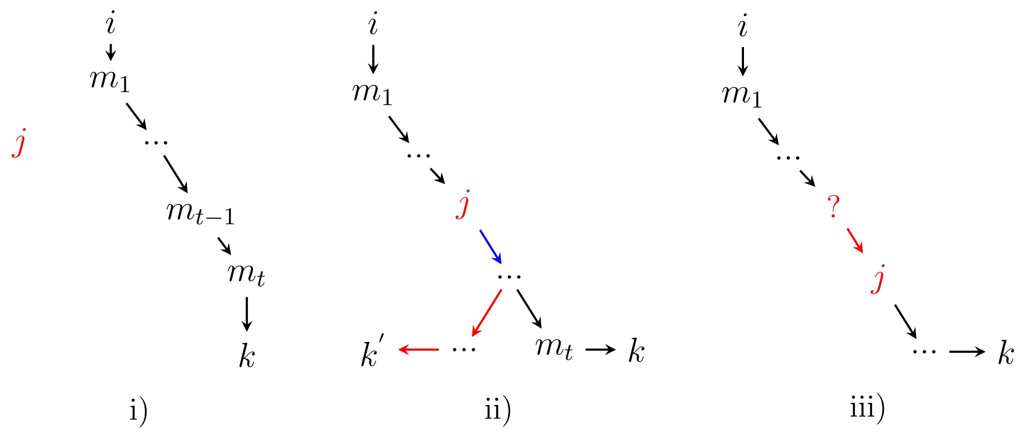


Figure 3.2: The three subcases where there exists a node $k \in \mathrm{de}(i) \cap A_i$.

iii) In the last subcase, we consider that every minimal path from i to k contains j and that j is being visited only after the path intersects with nodes in A_j. If the node preceding j is not in A_j, we are again in the case of a) with i and j flipped, since we would have $\mathrm{pa}(j)\backslash A_j \neq \emptyset$. Otherwise, we set all edge weights to zero except for those on the path. Let $A_j^{'}$ be the nodes of A_j in the path. In the new DAG containing only the edges of the directed path, $A_j^{'}$ satisfies $\mathrm{pa}(j) \subseteq A_j^{'} \subseteq V \setminus \mathrm{de}(j)$. We can now use Theorem 3.3 and obtain

$$w_j = \sigma_j - \Sigma_{j,A_j'}(\Sigma_{A_j',A_j'})^{-1}\Sigma_{A_j',j} = \sigma_j - \Sigma_{j,A_j}(\Sigma_{A_j,A_j})^{-1}\Sigma_{A_j,j}.$$

But computing the left-hand side of (3.3) using the same steps as in i), we get the following strict inequality:

$$\sigma_i - \Sigma_{i,A_i}(\Sigma_{A_i,A_i})^{-1}\Sigma_{A_i,i} = w_i - \left(w_i\lambda_{im_1}\lambda_{m_tk}\prod_{i=2}^{t}\lambda_{m_{s-1}m_s}\right)^2\frac{1}{\sigma_k} < w_i$$
$$= w_j = \sigma_j - \Sigma_{j,A_j}(\Sigma_{A_j,A_j})^{-1}\Sigma_{A_j,j}.$$

□

Theorem 3.6 gives us an algebraic description of partially homoscedastic linear Gaussian models. We can express the equal variance constraints in the form of equations between conditional variances, where the choice of conditioning sets can vary within a given range of sets. In the following result, we order this range of sets from smallest to largest one, where we partially order sets by set inclusion and extend the ordering to pairs of sets, i.e., $(A_i, A_j) \leq (B_i, B_j)$ if $A_i \subsetneq B_i$ or if $A_i = B_i$ and $A_j \subseteq B_j$.

Corollary 3.7. *Let $G = (V, E)$ be a DAG, and let Π be a partition of V with nodes $i, j \in V$ such that $i \sim_\Pi j$, i.e., they are in the same partition block. Let $\mathcal{A}_{ij}$ be the family of all pairs (A_i, A_j) with $A_i \subseteq V \setminus \{i\}$ and $A_j \subseteq V \setminus \{j\}$ for which equation (3.3), i.e.,*

$$\sigma_i - \Sigma_{i,A_i}(\Sigma_{A_i,A_i})^{-1}\Sigma_{A_i,i} = \sigma_j - \Sigma_{j,A_j}(\Sigma_{A_j,A_j})^{-1}\Sigma_{A_j,j},$$

holds for all covariance matrices $\Sigma \in M_{G,\Pi}$. It follows that

i) $\mathcal{A}_{ij}$ contains a unique minimal pair, namely, $A_i = \mathrm{pa}(i)$ and $A_j = \mathrm{pa}(j)$, and

ii) $\mathcal{A}_{ij}$ contains a unique maximal pair, namely, $B_i = V \setminus \mathrm{de}(i)$ and $B_j = V \setminus \mathrm{de}(j)$.

3.3 Characterization of the Models

In the last section, we developed algebraic constraints for our partially homoscedastic model setup. In order to make statements about model equivalence, we additionally need conditional independence constraints derived from d-separation. We now show that equal variance assumptions do not alter the set of conditional independence statements in a linear SEM. We will use the proof of Geiger and Pearl (1990)[Theorem 1 and 3], where they showed soundness and completeness of d-separation in SEMs, by modifying it for our setting.

Proposition 3.8. *Let $G = (V, E)$ be a DAG, and let Π be a partition of V. Let i, j be two distinct nodes and let $S \subseteq V \setminus \{i, j\}$. Then the conditional independence $X_i \perp\!\!\!\perp X_j \mid S$ holds for all multivariate normal random vectors X with covariance matrix in $M_{G,\Pi}$ if and only if the d-separation $i \perp\!\!\!\perp_G j \mid S$ holds in G.*

Proof. The "if" direction follows directly from Theorem 2.10 ii), since $M_{G,\Pi} \subseteq M_G$. For the "only if" direction, we assume that i and j are not d-separated given S in G. This means there is a path q which d-connects i and j given S, i.e., every collider is in S and every non-collider is outside of S. If $S = \emptyset$, a trek exists between i and j. Setting all edge weights to zero, except for those on the trek,

$$\begin{tikzcd}
i \arrow[r] & \cdots \arrow[r] & h_1 \arrow[d] & \cdots \arrow[l] \arrow[r] & h_2 \arrow[d] & \cdots \arrow[l] & & \cdots \arrow[r] & h_k \arrow[d] & \cdots \arrow[l] & j \arrow[l] \\
 & & \vdots \arrow[d] & & \vdots \arrow[d] & & & & \vdots \arrow[d] & & \\
 & & z_1 & & z_2 & & & & z_k & &
\end{tikzcd}$$

Figure 3.3: The active path q.

we can conclude by the trek rule that in the covariance matrix $\Sigma_{i,j} \neq 0$ holds and hence $X_i \not\perp\!\!\!\perp X_j$. Now let $S \neq \emptyset$. We denote by $S^{'} = \{z_1, z_2, \ldots, z_k\} \subseteq S$ all colliders (see Figure 3.3) in the path. We will now construct a covariance matrix Σ, such that $X_i \not\perp\!\!\!\perp X_j \mid S$. To make sure that our covariance matrix is in $M_{G,\Pi}$ and not merely in M_G, we set all error variances $w_i = 1$. All edge weights are set to zero, except for those edges on the path q, which we assign the same value $\rho \in (0, 1)$. Let $\Sigma = \phi_G(\Lambda, w)$ be the covariance matrix obtained by the resulting choice of Λ and w.
Using the trek rule for the diagonal entries of $\Sigma = (\sigma_{kl})$ we get

$$\sigma_{kk} = 1 \ \forall k \in S \setminus S^{'},$$

since all nodes which are in both S and the path q are the colliders in the set $S^{'}$. Now we can observe that between each pair of consecutive nodes in the sequence $i \equiv z_0, z_1, z_2, \ldots, z_k, z_{k+1} \equiv j$ there exists a unique non-zero trek. Denoting with r_t the number of edges that go from z_t to z_{t+1} in the path q, the trek rule satisfies for all $t = 0, \ldots, k$:

$$\sigma_{z_t, z_{t+1}} = \rho^{r_t}.$$

Using all that information and ordering the nodes as $i, z_1, \ldots, z_k, j$ followed by the nodes in $S \setminus S^{'}$, we obtain that

$$\Sigma_{ijS,ijS} = \left(\begin{array}{cccccc|c}
\sigma_{i,i} & \rho^{r_0} & 0 & \ldots & 0 & 0 & \\
\rho^{r_0} & \sigma_{z_1,z_1} & \rho^{r_1} & \ldots & 0 & 0 & \\
0 & \rho^{r_1} & \sigma_{z_2,z_2} & \ddots & 0 & 0 & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \\
0 & 0 & 0 & \ddots & \sigma_{z_k,z_k} & \rho^{r_k} & \\
0 & 0 & 0 & \vdots & \rho^{r_k} & \sigma_{j,j} & \\
\hline
\multicolumn{6}{c|}{0} & I_{S \setminus S'}
\end{array} \right)$$

By removing the i-th row and j-th column, we get a lower triangular matrix and hence $\det(\Sigma_{iS,jS}) = \rho^{\sum_{t=0}^{k} r_t} \neq 0$. By Theorem 2.9 we deduce that $X_i \not\perp\!\!\!\perp X_j \mid S$ and finish the proof.

$\square$

Summarizing all facts about conditional independence and equal variance constraints, we can now fully describe a partially homoscedastic Gaussian linear model.

Theorem 3.9. *Let $G = (V, E)$ be a DAG with $|V| = p$, and let Π be a partition of V. Then a covariance matrix $\Sigma \in PD$ is in the partially homoscedastic Gaussian linear model $M_{G,\Pi}$ if and only if Σ satisfies all conditional independence constraints given by d-separation and all equal variance constraints from Corollary 3.4.*

Proof. The "only if" direction follows directly from Proposition 3.8 and Corollary 3.4. For the "if" direction we assume that Σ satisfies all condition independence and equal variance constraints associated with G. Theorem 2.10 ii) tells us that a covariance matrix, which satisfies all conditional independence constraints given by d-separation, is an element of M_G. Consequently there exist $\Lambda \in \mathbb{R}^E$ and $w \in (0, \infty)^p$ such that $\Sigma = \phi_G(\Lambda, w) \in M_G$. Theorem 3.3 implies that $w_i = w_j$ for $i \sim_\Pi j$. It follows that $\Sigma \in M_{G,\Pi}$. $\square$

3.4 Model Equivalence

We can now discuss model equivalence in partially homoscedastic Gaussian linear SEMs. First, we formalize the idea of model equivalence in partially homoscedastic linear Gaussian SEMs analogous to the standard case.

Definition 3.10. Let Π be a partition of the vertex set V. We call two DAGs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ Π-*model equivalent* if $M_{G_1,\Pi} = M_{G_2,\Pi}$. In this case, we write $G_1 \approx_\Pi G_2$.

Lemma 2.12 gives us graphical criteria for determining whether two DAGs are Markov equivalent, i.e., when they describe the same d-separation relations. We extend these criteria now for the partially homoscedastic setup.

Theorem 3.11. *Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be DAGs with $|V| = p$, and let Π be a partition of V. Then G_1 and G_2 are Π-model equivalent if and only if the following conditions hold :*

i) G_1 and G_2 have the same skeleton and unshielded colliders, and

ii) $\mathrm{pa}_{G_1}(i)= \mathrm{pa}_{G_2}(i)$ for all nodes i that belong to a partition block $\pi_k \in \Pi$ with size $|\pi_k| \geq 2$.

Proof. For the "if" direction we assume that conditions $i)$ and $ii)$ hold. From $i)$ follows that G_1 and G_2 entail the same d-separation relations and hence $M_{G_1} = M_{G_2}$ by Lemma 2.12. Let $\Sigma \in M_{G_1,\Pi}$. Since $M_{G_1,\Pi} \subseteq M_{G_1} = M_{G_2}$, we can find $\Lambda^{(2)} \in \mathbb{R}^{E_2}$ and $w^{(2)} \in (0,\infty)^p$ such that $\Sigma = \phi_{G_2}(\Lambda^{(2)}, w^{(2)})$. Now let $i \neq j$ be two nodes which belong to the same partition block π_k with size $|\pi_k| \geq 2$. Using that $\Sigma \in M_{G_1,\Pi}$, we get by Corollary 3.4

$$\sigma_i - \Sigma_{i,\mathrm{pa}_{G_1}(i)}(\Sigma_{\mathrm{pa}_{G_1}(i),\mathrm{pa}_{G_1}(i)})^{-1}\Sigma_{\mathrm{pa}_{G_1}(i),i} = \sigma_j - \Sigma_{j,\mathrm{pa}_{G_1}(j)}(\Sigma_{\mathrm{pa}_{G_1}(j),\mathrm{pa}_{G_1}(j)})^{-1}\Sigma_{\mathrm{pa}_{G_1}(j),j}.$$

Following (ii) we have $\mathrm{pa}_{G_1}(i) = \mathrm{pa}_{G_2}(i)$ and $\mathrm{pa}_{G_1}(j) = \mathrm{pa}_{G_2}(j)$ and hence

$$\begin{aligned} w_i^{(2)} &= \sigma_i - \Sigma_{i,\mathrm{pa}_{G_2}(i)}(\Sigma_{\mathrm{pa}_{G_2}(i),\mathrm{pa}_{G_2}(i)})^{-1}\Sigma_{\mathrm{pa}_{G_2}(i),i} \\ &= \sigma_j - \Sigma_{j,\mathrm{pa}_{G_2}(j)}(\Sigma_{\mathrm{pa}_{G_2}(j),\mathrm{pa}_{G_2}(j)})^{-1}\Sigma_{\mathrm{pa}_{G_2}(j),j} = w_j^{(2)}. \end{aligned}$$

Consequently, we have by Theorem 3.9 that $\Sigma \in M_{G_2,\Pi}$ and hence $M_{G_1,\Pi} \subseteq M_{G_2,\Pi}$. Swapping the roles of G_1 and G_2 we conclude that $M_{G_1,\Pi} = M_{G_2,\Pi}$.
For the "only if" direction we assume that $M_{G_1,\Pi} = M_{G_2,\Pi}$ holds. Theorem 3.9 implies, that G_1 and G_2 induce the same conditional independence constraints given by d-separation. It follows that condition $i)$ holds. Let $i \neq j$ be two nodes belonging to the same partition block π_k. Theorem 3.9 also implies, that G_1 and G_2 induce the same equal variance constraints, given as in Corollary 3.4. That means that the set $\mathcal{A}_{ij}$, defined as in Corollary 3.7, is the same for G_1 and G_2. We now use the result from Corollary 3.7, which says, that the unique minimal element of $\mathcal{A}_{ij}$ consists of the parent set of i and j in both G_1 and G_2. Therefore $\mathrm{pa}_{G_1}(i) = \mathrm{pa}_{G_2}(i)$ and $\mathrm{pa}_{G_1}(j) = \mathrm{pa}_{G_2}(j)$, and hence condition $ii)$ holds. □

Remark 3.12. If follows directly that in the standard heteroscedastic case with $\Pi = \{\{1\}, \{2\}, ..., \{p\}\}$ this theorem reduces to the standard Markov equivalence result (Lemma 2.12). In that case, the graph can be identified by the joint distribution only up to its Markov equivalence class (assuming faithfulness, see Lemma 7.2

in Peters et al. (2017)). Whereas in the full homoscedastic case $\Pi = \{\{1, 2, ..., p\}\}$ with all noise variables having the same variance, we conclude that no two distinct graphs can define the same model since all edges are fixed by condition *ii*). Also proven in Peters and Bühlmann (2014).

We know from Drton (2018) that the parametrization $\phi_G(\Lambda, w)$ is injective, hence $\phi_G(\Lambda, w_1) \neq \phi_G(\Lambda, w_2)$ for $w_1 \neq w_2$. It follows that for a DAG $G = (V, E)$ and partitions $\Pi_1 \neq \Pi_2$ we have in general $M_{G,\Pi_1} \neq M_{G,\Pi_2}$. In this novel result, we look further at which necessary and sufficient conditions two partially homoscedastic linear Gaussian SEMs are the same.

Theorem 3.13. *Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be DAGs with $|V| = p$. Let Π_1 and Π_2 be partitions of V. Then $M_{G_1,\Pi_1} = M_{G_2,\Pi_2}$ if and only if* $\Pi_1 = \Pi_2$ *and $G_1 \approx_{\Pi_1} G_2$ (or $G_1 \approx_{\Pi_2} G_2$).*

Proof. The "if" direction follows directly from Theorem 3.11. For the "only if" direction, we do a proof by contradiction. Suppose that $\Pi_1 \neq \Pi_2$. That means there exists $i, j \in \{1, \dots, p\}$ such that $i \sim_{\Pi_2} j$ and $i \not\sim_{\Pi_1} j$. We now choose $\Lambda^{(1)} = 0$ to be the edge weight matrix and $w^{(1)} = (1, \dots, 1, w_i, \dots, w_j, 1, \dots, 1) \in \mathbb{R}^p$ with $w_i \neq w_j$ to be the the corresponding vector of error variances. The covariance matrix obtained by this parametrization is hence $\Sigma^{(1)} = \phi_{G_1}(\Lambda^{(1)}, w^{(1)}) = \text{diag}(1, \dots, 1, w_i, 1, \dots, w_j, 1, \dots, 1) \in M_{G_1,\Pi_1}$. Since we have $M_{G_1,\Pi_1} = M_{G_2,\Pi_2}$ it follows that $\Sigma_1 \in M_{G_2,\Pi_2}$. Let $\Sigma^{(1)} = \phi_{G_2}(\Lambda^{(2)}, w^{(2)})$ be the parametrization where $\Lambda^{(2)} \in \mathbb{R}^{E_2}$ and $w^{(2)} \in (0, \infty)^p$, such that if follows the partition of Π_2.

Since $\Sigma^{(1)}$ is a diagonal matrix, we can conclude that $\Lambda^{(2)} = 0$. We prove this claim by contradiction. Let us assume that $\Lambda^{(2)} \neq 0$ and (k, l) be the "minimal" pair of nodes (minimal according to the topological ordering of pair of nodes in G_2), which have a non-zero edge between them, i.e., $\Lambda^{(2)}_{k,l} \neq 0$. Because of this "minimality", there are no non-zero treks of the form $k \leftarrow \cdots \leftarrow t \rightarrow \cdots \rightarrow l$ or $k \rightarrow \cdots \rightarrow t \rightarrow \cdots \rightarrow l$ between k and l. The only non-zero trek is therefore $k \rightarrow l$ and applying the trek rule, we get $\Sigma^{(1)}_{kl} \neq 0$, which is a contradiction to $\Sigma^{(1)}$ being a diagonal matrix.
Using now that $\Lambda^{(2)} = 0$, we conclude that $\Sigma^{(1)} = \text{diag}(w^{(2)})$ with $w^{(2)}_i \neq w^{(2)}_j$, which is a contradiction to $i \sim_{\Pi_2} j$. This finishes the proof. □

In classical Markov equivalence theory an edge $x \rightarrow y$ is called *compelled* in a DAG G, if for every DAG G' Markov equivalent to G, $x \rightarrow y$ exists in G'. Otherwise, an edge $x \rightarrow y$ is termed *reversible* if there exists a DAG G' that is equivalent to G and in which the direction of that edge is reversed. As mentioned before,

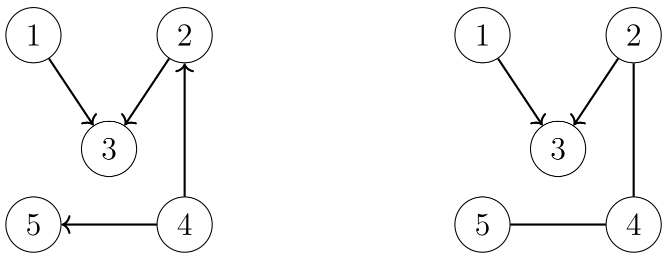


Figure 3.4: DAG (left) and corresponding CPDAG (right)

Markov equivalence classes can be represented by a so-called *completed partially directed acyclic graph* (CPDAG), as shown in Andersson et al. (1997). CPDAGs are *mixed graphs*, meaning they consist of directed and undirected edges. The directed edges in the CPDAG are the compelled edges of the represented equivalence class. Undirected edges stand for the reversible edges in that class (see Figure 3.4).

We now adapt these ideas for our partially homoscedastic setup.

Definition 3.14. Let $G = (V, E)$ be a DAG and Π a partition of the vertex set. The CPDAG (*completed partially directed acyclic graph*) of G *under the partition* Π is obtained by taking the union of all DAGs that are equivalent to G :

$$G^*_\Pi := \cup \; (G' |\, G' \approx_\Pi G).$$

Adapting the notion of compelled and reversible edges in the standard heteroscedastic case to our Π-model equivalence, G^*_Π consists of directed edges for compelled edges and undirected edges for reversible edges. An edge $x \to y$ is therefore called *compelled* in a DAG G if for every DAG G' Π-model equivalent to G, $x \to y$ exists in G'. Analogously, an edge $x \to y$ is termed *reversible* if there exists a DAG G' that is Π-model equivalent to G and in which the direction of that edge is reversed. It follows from Theorem 3.11 that, besides unshielded colliders, all edges of nodes from a partition block of size bigger than two are compelled in a Π-model equivalence class.

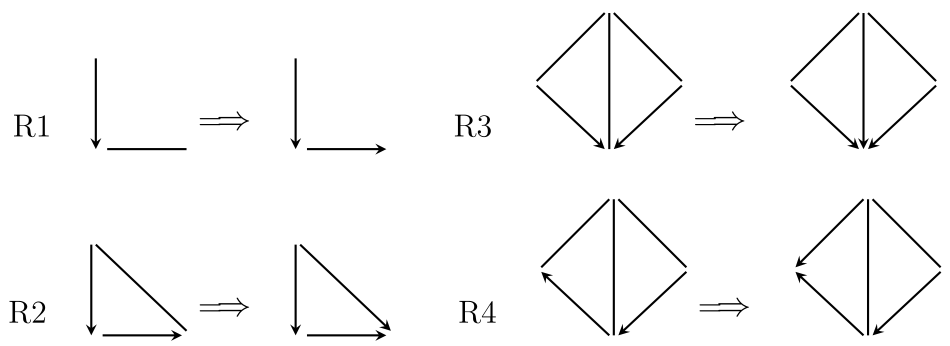


Figure 3.5: The 4 orientation rules

We now focus on the task of obtaining a CPDAG given DAG G and partition Π. In the standard heteroscedastic case, Meek (1995) shows how to construct a CPDAG given conditional independence statements. Furthermore, Meek (1995) details the procedure for constructing a CPDAG when provided with background knowledge regarding certain edges. Background knowledge is encoded in $\mathcal{K} = \langle \mathbf{F}, \mathbf{R} \rangle$, where $\mathbf{F}$ consists of edges not in DAG and $\mathbf{R}$ of edges in DAG. The whole process can be summarized as follows:

I. Conditional independence statements are translated into adjacencies and unshielded collider triples.

II. The first 3 of the 4 orientation rules by Verma and Pearl (1992)(see Figure 3.5) are applied.

Phases I and II give the classical CPDAG without background knowledge. In the final phase, background knowledge is integrated, and the existence of a compatible CPDAG is verified. CPDAG at the current step is denoted by G^*.

III. The following steps are taken :

    S1 If there is an edge $i \rightarrow j$ in $\mathbf{F}$ such that $i \rightarrow j$ in G^* then FAIL.

    S1' If there is an edge $i \rightarrow j$ in $\mathbf{R}$ such that $j \rightarrow i$ in G^* or i, j not adjacent in G^* then FAIL.

    S2 Randomly choose one edge $i \rightarrow j$ from $\mathbf{R}$ and let $\mathbf{R} = \mathbf{R} \setminus \{i \rightarrow j\}$.

    S3 Orient $i \rightarrow j$ in G^* and close orientations under R1, R2, R3 and R4.

    S3 If $\mathbf{R} \neq \emptyset$, then go to S1.

Algorithm 1 Constructing CPDAG given DAG and partition

```
Require: DAG $G$, partition $\Pi$ of vertex set $V$
1: Create an empty graph $G'$
2: Copy the skeleton and all edge orientations with unshielded colliders of $G$ to $G'$
3: Apply rules R1, R2 and R3 on $G'$ until no more edges can be oriented
4: for $i \in V$ with $i \in \pi_k$ and $|\pi_k| \geq 2$ do
5:     Copy the orientation of edges in $G$ having one endpoint at $i$ to $G'$
6: end for
7: Apply rules R1 and R2 on $G'$ until no more edges can be oriented
8: return $G^*_\Pi = G'$
```

We now adapt this procedure to our partially homoscedastic setup by using a simplified version of it, see Algorithm 1. The background knowledge consists of all edges of nodes from a block π_k with size $|\pi_k| \geq 2$. A proof is needed to show the correctness of the algorithm.

Theorem 3.15. *Let $G = (V, E)$ be a DAG, and let Π be a partition of V. Then Algorithm 1 outputs the correct corresponding CPDAG G^*_Π.*

Proof. We now try to fill the gaps between the general procedure of constructing a CPDAG with background knowledge and the simplified procedure in Algorithm 1. The general algorithm has been proven in Meek (1995)(Theorems 2-4).

First, we create the CPDAG without background knowledge (CPDAG in classic heteroscedastic setup). Because we know the DAG G, all conditional independence statements can be read off and are therefore known. Thus, we only need to copy the skeleton and unshielded colliders of G to empty graph G'. Then, we apply the orientation rules R1, R2, and R3 to the graph. This concludes Phase I and II.

Next, we insert the background knowledge from the given partition Π into G'. Equal variance constraints determine the adjacencies of all vertices in a block π_k with size $|\pi_k| \geq 2$. Hence $\mathcal{K} = \langle \mathbf{F}, \mathbf{R} \rangle$, where $\mathbf{R}$ consists of all those edges with one endpoint in these vertices and $\mathbf{F}$ consists of the reversal of those edges.

We adopt the steps of Phase III the following :

i) Since DAG and partition are given, we know the existence of at least one member of the equivalence class; therefore, the general algorithm will not fail. Hence, the background knowledge checks S1 and S1' are redundant and can be omitted.

ii) We can now simultaneously add the edges from $\mathbf{R}$ and then close the orientations sequentially. Every newly directed edge relies on background knowledge. Once all the dependencies are accounted for and incorporated, the edge can be oriented without encountering any conflicts.

iii) As a result of this special type of background knowledge, only the situations of rule R1 and R2 can happen. Creating examples where the patterns in R1 and R2 originate from knowledge of the adjacency directions of specific nodes is simple.
In R3, there exists an unshielded collider triple. However, when propagating with background knowledge, no new collider triples emerge. Otherwise, the resulting CPDAG would not possess the same conditional independence statements as the original DAG. Consequently, any patterns observed in R3 must have arisen during the construction of the CPDAG without background knowledge and will not appear in the final propagation phase.
Let us assume that R4 appears. We now examine the first appearance of R4. The edge $i_3 \to i_4$ is not obtained during the construction of standard CPDAG without background knowledge because $i_4 \to i_1$ would have also been oriented, and hence pattern R4 appears, which is a contradiction. If the orientation $i_3 \to i_4$ is a direct result of background knowledge, it would imply that we also know the orientations of all adjacent edges connected to either i_3 or i_4. This would lead to the orientation of $i_2 - i_3$ or $i_2 - i_4$, which again is a contradiction. Figure 3.6 illustrates the scenario where the orientation $i_3 \to i_4$ is obtained from R1 using unshielded triple $l \to i_3 - i_4$. To maintain the non-orientation of $i_2 - i_3$, the edge $l \to i_2$ becomes necessary. Additionally, the undirected edge $i_2 - i_4$ implies an adjacency between l and i_4. However, this creates a shielded triple (l, i_3, i_4), contradicting the pattern described in R1. Figure 3.7 shows the case of $i_3 \to i_4$ resulting from R2. In order to keep $i_2 - i_4$ not oriented, the edge $l \to i_2$ must exist. But as a consequence, R2 can be applied and hence $i_2 - i_3$ is oriented as $i_3 \to i_2$, which again leads to a contradiction.

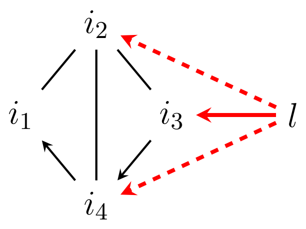


Figure 3.6: $i_3 \to i_4$ from R1

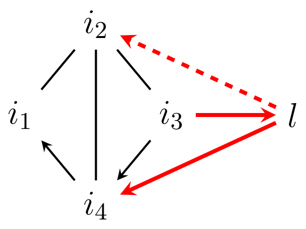
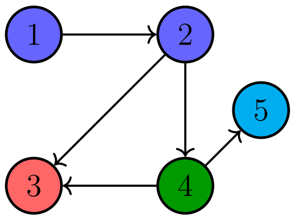


Figure 3.7: $i_3 \to i_4$ from R2

We proved that the modifications to the general algorithm using a special type of background knowledge are correct.

$\square$



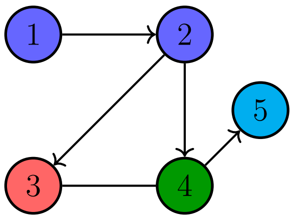


Figure 3.8: DAG (left) and corresponding CPDAG (right) under fixed partition

Consider the example in Figure 3.8. The vertex set is $V = \{1, 2, 3, 4, 5\}$ and the partition is given by $\Pi = \{\{1, 2\}, \{3\}, \{4\}, \{5\}\}$. Applying Algorithm 1 on the DAG to obtain the CPDAG, we do the following steps: First, we keep the skeleton and all unshielded colliders (in that case, there are none). Then, those edges containing node 1 or 2 are oriented the same as in the DAG. The last step is to propagate the rules R1 and R2. Consequently, edge $4 \to 5$ is oriented by R1, and $3 - 4$ stays undirected.

3.5 Examples for Π-model Equivalence Classes

The assumptions of partial homoscedasticity are interesting as their emergence of error variance constraints leads to a refinement of the Markov equivalence classes. Using the results of the last chapter, we illustrate this point by looking at some examples of Π-model equivalence classes and comparing them to classic Markov equivalence classes.

Enumerating the number of Markov equivalence classes is a difficult problem that a few publications have addressed. Radhakrishnan et al. (2018) conclude that counting the number of Markov equivalence classes must be NP-hard. Gillispie and Perlman (2001) were able to calculate the number of equivalence classes, as specified by Pearl and Verma (Lemma 2.12), up to 10 nodes. Table 3.1 shows their results. As we can see, this quantity grows extremely large.

Table 3.1: Equivalence classes by number of nodes p (from Gillispie and Perlman (2001))

p	Equivalence classes
1	1
2	2
3	11
4	185
5	8782
6	1067825
7	312510571
8	212133402500
9	326266056291213
10	1118902054495975141

Theorem 3.11 gives us criteria for deciding model equivalence in our new partially homoscedastic setup. In addition to skeleton and collider information, we need to check on the partition information of specific nodes. Consequently, the number of equivalence classes increases in most cases in comparison to the standard heteroscedastic case. We look in the following at some examples and how, depending on the partition, the classical equivalence classes are refined.

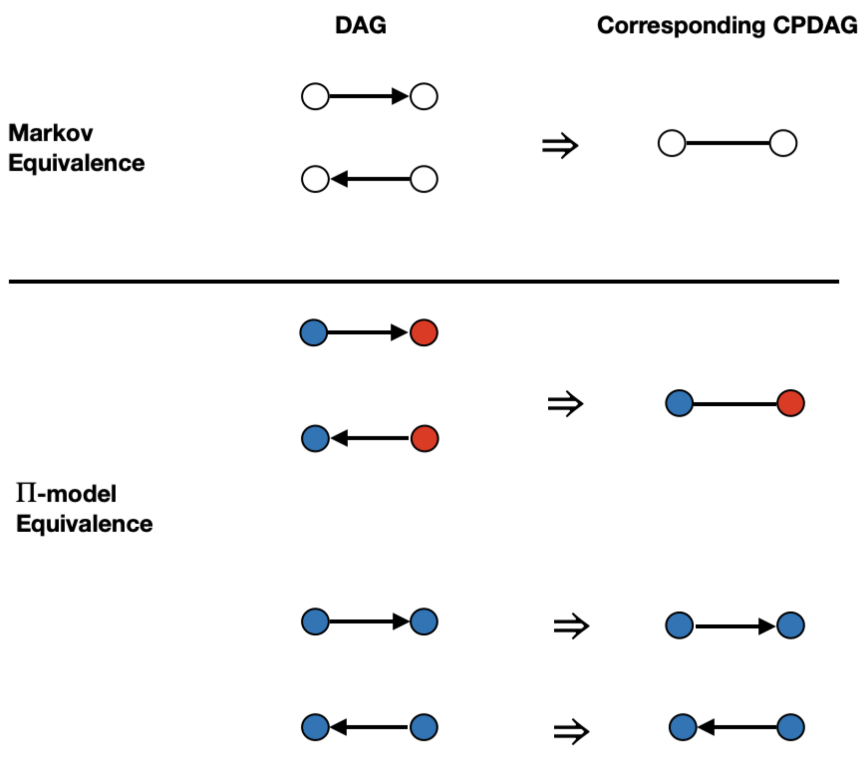


Figure 3.9: Comparison of Markov equivalence and Π-model equivalence in an example with two nodes. The color of nodes indicates partition grouping.

Figure 3.9 shows the simple case of an equivalence class with DAGs consisting of two nodes. In that case, we only have two possible partitions. The heteroscedatic case results in the classic Markov equivalence class, and full homoscedasticity results in every DAG defining its own equivalence class (see Remark 3.12). Figure 3.10 shows an example with three nodes and Π-model equivalence of two partitions. For three variables, we have five possible partitions (more on counting of partitions in 4.2). It is easy to see that, except for the finest possible partition, every partition leads to the situation of every DAG being its own CPDAG.
Figure 3.11 shows an example with four nodes and Π-model equivalence of two partitions. Both partitions lead to a refinement of the Markov equivalence class. As in the other examples, these refinements follow from the partition structure and Theorem 3.11, which specifies that the edges of nodes in a block with a size greater or equal to two are fixed.

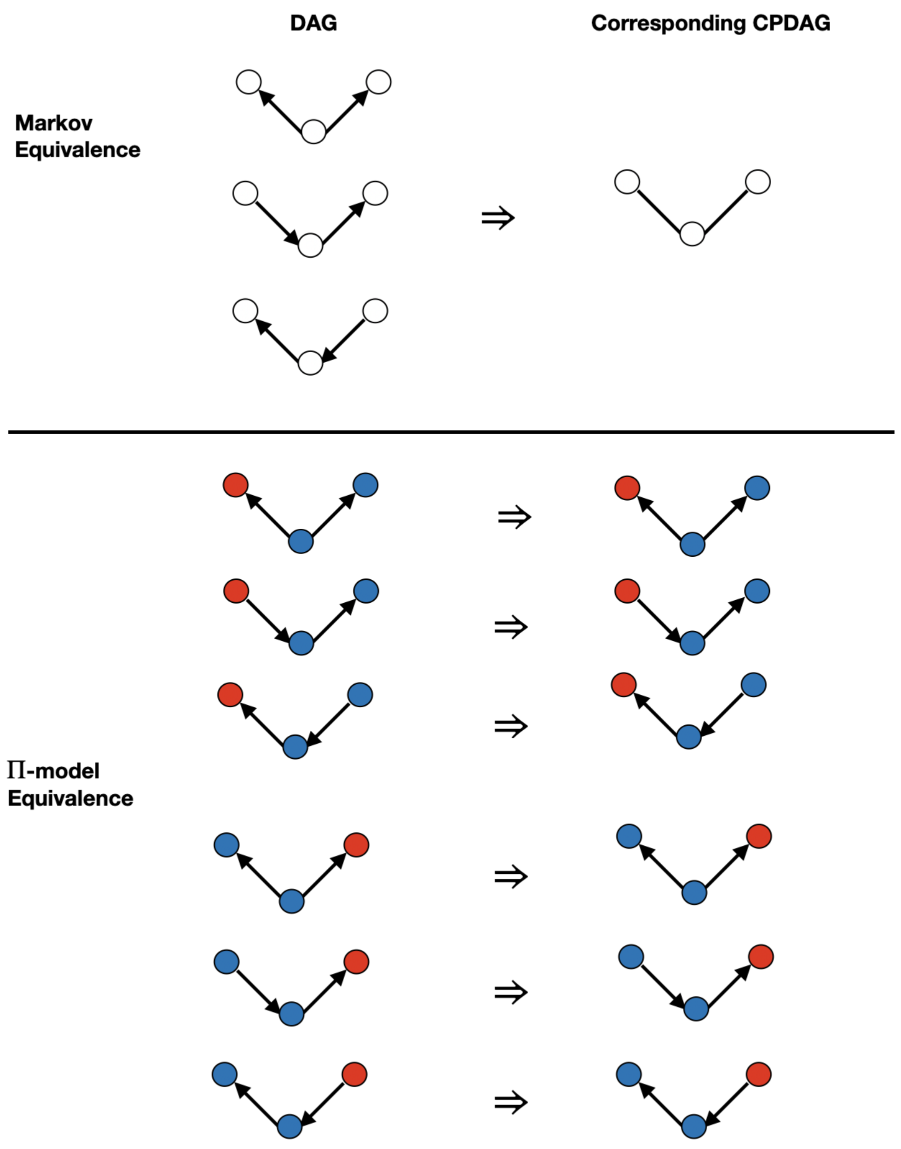


Figure 3.10: Comparison of Markov equivalence and Π-model equivalence for two different partitions in an example with three nodes. The color of nodes indicates partition grouping.

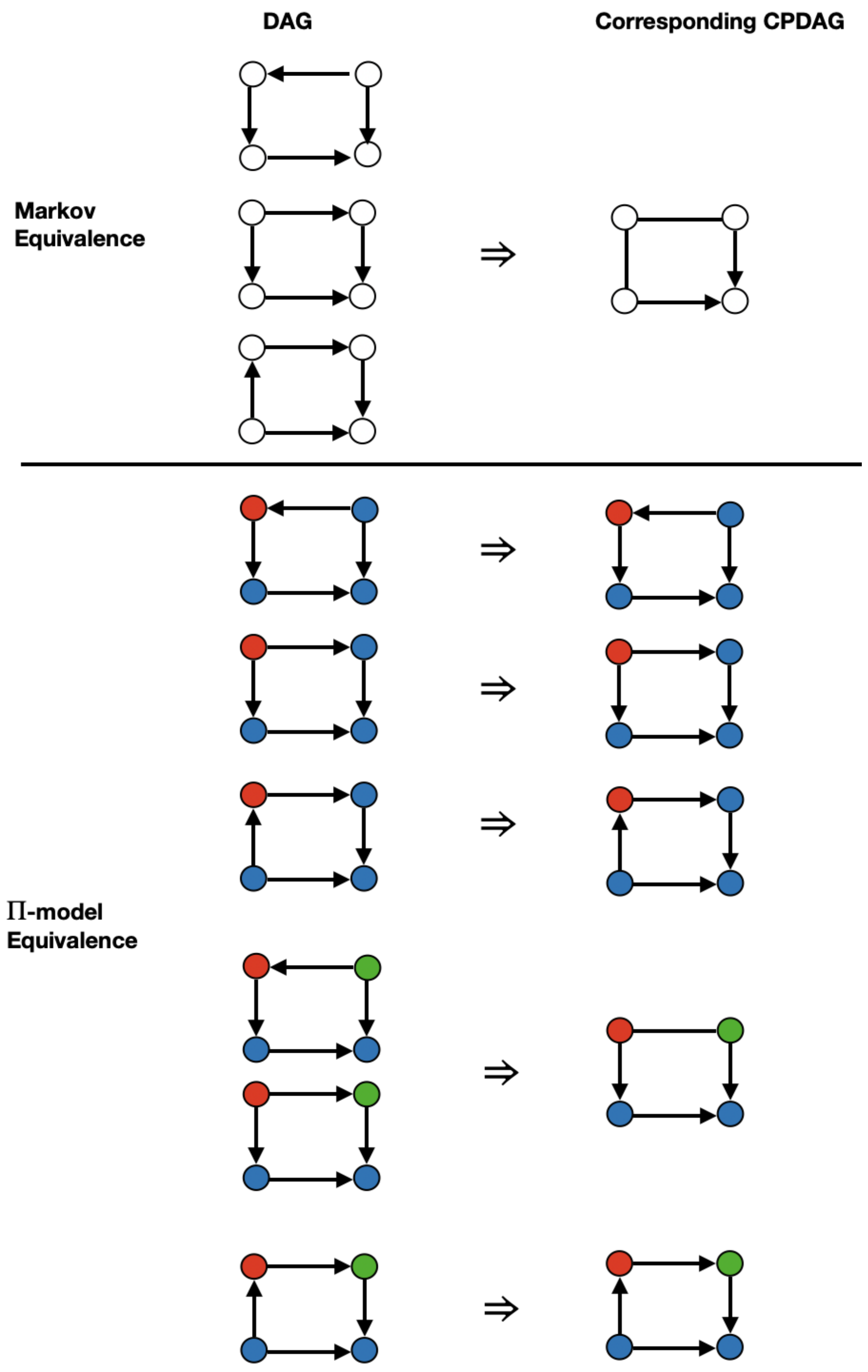


Figure 3.11: Comparison of Markov equivalence and Π-model equivalence for two different partitions in an example with four nodes. The color of nodes indicates partition grouping.

Remark 3.16. We can formalize these observations to a more general statement about the number of equivalence classes under partitioning. Let $p \geq 3$ be the number of nodes. Let Π be a partition of the nodes, such that there is at most one partition block $\pi_l \in \Pi$ with size $|\pi_l| = 1$. Then, the number of Π-model equivalence classes equals the number of DAGs.

4 Greedy Search Algorithms

In the last chapter, we developed the necessary theory for partially homoscedastic Gaussian linear SEMs. We now take the next step and look at how to learn DAG and partition of this model from data.
There have been different approaches to learning the structure of directed graphical models. Two of the most popular ones are constraint-based methods and score-based methods. Constrain-based methods perform statistical conditional independence tests on the data to infer the graph. One commonly used constraint-based method is the PC algorithm (Spirtes et al. (2000)). Score-based methods compare and evaluate different candidates of DAG/CPDAGs based on a scoring criterion (e.g., penalized likelihoods like BIC) and try to optimize the score. Popular examples of these types of methods are the GES algorithm (Chickering (2003)) and MMHC (Tsamardinos et al. (2006)).
In this thesis, the focus will be on greedy score-based search algorithms, and this chapter introduces searching schemes for partition and DAGs (Sections 4.2 + 4.3). These will serve as building blocks for constructing various sophisticated methods to estimate the CPDAG under partitioning (Section 4.4). We start by stating the likelihood of a partially homoscedastic Gaussian linear SEM and the maximum likelihood estimates of its parameters to calculate a score.

4.1 Likelihood Inference

Given DAG $G = (V, E)$ with $|V| = p$ and partition Π with $|\Pi| = K$ blocks, we can compute the maximum likelihood estimation of edge weight and error variance parameter (Λ, ω). Let $\mathbf{X} = (X_1, \ldots, X_p)^T \in \mathbb{R}^{p \times n}$ be a data matrix, with n i.i.d and centered columns generated from a multivariate normal distribution $\mathcal{N}(0, \Sigma)$ with covariance matrix Σ. Let $S = \mathbf{X}\mathbf{X}^T/n$ be the sample covariance matrix.

The likelihood function is given by

$$\begin{aligned}
L_G(\Lambda, \omega) &= \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^p \det(\Sigma)}} \ \exp\Big\{ -\frac{1}{2} X_{(i)}^T \Sigma^{-1} X_{(i)} \Big\} \\
&= (2\pi)^{-\frac{np}{2}} \ \det(\Sigma)^{-\frac{n}{2}} \prod_{i=1}^{n} \exp\Big\{ -\frac{1}{2} X_{(i)}^T \Sigma^{-1} X_{(i)} \Big\},
\end{aligned}$$

where $X_{(i)} \in \mathbb{R}^p$ denotes the i-th observation. Using now the parametrization of Σ by Λ and ω, we can rewrite

$$\det(\Sigma)^{-\frac{n}{2}} = \Big[\det((I-\Lambda))^{-2} \ \det(\operatorname{diag}(\omega)) \Big]^{-\frac{n}{2}}.$$

Using again the parametrization of Σ and trace rules, we obtain

$$\begin{aligned}
\prod_{i=1}^{n} \exp\Big(-\frac{1}{2} X_{(i)}^T \Sigma^{-1} X_{(i)} \Big) &= \exp\Big\{ \sum_{i=1}^{n} -\frac{1}{2} X_{(i)}^T (I-\Lambda) \operatorname{diag}(\omega)^{-1} (I-\Lambda)^T X_{(i)} \Big\} \\
&= \exp\Big\{ -\frac{n}{2} \sum_{i=1}^{n} \operatorname{tr}((I-\Lambda) \operatorname{diag}(\omega)^{-1} (I-\Lambda)^T \frac{1}{n} X_{(i)} X_{(i)}^T) \Big\} \\
&= \exp\Big\{ -\frac{n}{2} \operatorname{tr}((I-\Lambda) \operatorname{diag}(\omega)^{-1} (I-\Lambda)^T S) \Big\}.
\end{aligned}$$

Combining the results above, the log-likelihood is given by

$$\begin{aligned}
l_G(\Lambda, \omega) &= C + \frac{n}{2} \Big(-\log(\det(\omega)) + \log(\det((I-\Lambda))^2) - \operatorname{tr}((I-\Lambda) \operatorname{diag}(\omega)^{-1} (I-\Lambda)^T S) \Big) \\
&= C + \frac{n}{2} \sum_{k=1}^{K} \left(-|\pi_k| \log(\omega_k) - \frac{1}{n\omega_k} \left(\sum_{i \in \pi_k} \| X_i - \Lambda_{\mathrm{pa}(i),i}^T X_{\mathrm{pa}(i)} \|^2 \right) \right) \\
&= C + \frac{n}{2} \sum_{k=1}^{K} l_{G,\pi_k}(\Lambda, \omega_k),
\end{aligned}$$

where in the second equality, we use properties of diagonal and triangular matrices regarding determinants and trace rules. The sum can be decomposed into the sum of the log-likelihood values of the K blocks plus the constant $C = (-\frac{np}{2}) \log(2\pi)$.

The maximum log-likelihood value of Λ is calculated by performing linear regression at each node on its parents. The estimates of $(\widehat{\Lambda}, \widehat{\omega})$ are therefore given by

$$\widehat{\Lambda}_{\mathrm{pa}(i),i} = \underset{\beta \in \mathbb{R}^{|\mathrm{pa}(i)|}}{\operatorname{argmin}} \|X_i - \beta^T X_{\mathrm{pa}(i))}\|^2,$$

$$\widehat{\omega_k} = \frac{\|X_i - \widehat{\Lambda}^T_{\mathrm{pa}(i),i} X_{\mathrm{pa}(i)}\|^2}{n|\pi_k|}.$$

We can now compute the Bayesian information criterion (BIC), as described in Peters et al. (2017), for a DAG G under partition Π given observational data $\mathbf{X}$. The BIC score is decomposed into the sum of scores of each block.

$$\begin{aligned} s_{\mathrm{BIC}}(G, \Pi) &= \frac{1}{n}\Big(\widetilde{l}_G(\widehat{\Lambda}, \widehat{\omega}) - \frac{\log(n)}{2}|E|\Big) \\ &= \frac{1}{2}\sum_{k=1}^{K}\Big(- |\pi_k|\log(\widehat{\omega}_k) - |\pi_k| - \frac{\log(n)}{n}\sum_{i \in \pi_k}|\mathrm{pa}(i)|\Big), \end{aligned}$$

where $\widetilde{l}_G(\widehat{\Lambda}, \widehat{\omega})$ denotes the log-likelihood without the constant C, since it is not dependent on the data.

4.2 Search Schemes in Partition Space

We look now at ways to construct greedy search on the space of partitions. Let $G = (V, E)$ be a DAG with vertex set $V = \{1, 2, \dots, p\}$. The total number of partitions of V is given by the so-called *Bell number* B_p (Becker and Riordan (1948)). The first Bell numbers are $B_0 = 1, B_1 = 1, B_2 = 2$ and for $p \in \mathbb{N}$ they satisfy following recursion

$$B_{p+1} = \sum_{k=0}^{p}\binom{p}{k}B_k.$$

For 10 nodes/variables, we already get $B_{10} = 115975$ (OEIS Foundation Inc. (2023)) possible partitions and hence a large search space for partitions. Applying an exact score-based search would lead to an exhaustive search. For this reason, we propose two greedy search schemes, adapted from Andres (2020). As derived in the last section, we will use BIC as the scoring criterion. Let $s_{\mathrm{BIC}}(G, \Pi)$ be the BIC score for DAG $G = (V, E)$ and partition Π given observational data $X = (X_1, \dots, X_p)^T \in \mathbb{R}^{p \times n}$ with n i.i.d. and centered columns. The idea is now to traverse through the space of partitions while keeping the same DAG fixed. The DAG can either be already known or learned by algorithms like GES, for example.

Greedy joining algorithm

We first examine the so-called *greedy joining algorithm.* Typically starting from the finest partition possible ($\Pi_0 = \{\{1\}, \{2\}, \dots, \{p\}\}$), we recursively join pairs of subsets until no further improvement of BIC score is achievable. For any partition Π of V, let $A, B \in \Pi$ be two partition blocks. We define the partition, which is obtained by joining A and B :

$$\text{join}_{A,B}[\Pi] := (\Pi \setminus \{A, B\}) \cup \{A \cup B\}.$$

Starting from a partition, we first look at all pairwise combinations of subsets/partition blocks and construct the partitions derived by joining each one of these pairs together. All of these partitions form the so-called *local neighborhood.* For each partition in the neighborhood, we compute the BIC score and compare it amongst all resulting partitions. Consequently, we choose the highest-scoring partition and compare it to the BIC score of the current one. If it is higher, we move to the new partition and repeat all these steps until no improvement is possible. Subsets can only grow, and the number of subsets decreases by one in every step. The DAG is fixed before and does not change during the procedure. Figure 4.1 illustrates an exemplified running of the algorithm.

Algorithm 2 greedy-joining(Π,G)

```
Require: Initial partition $\Pi$, fixed DAG $G$, data $X$
  Choose $\{A, B\} \in \underset{\{A', B'\} \in \binom{\Pi}{2}}{\arg\max}\; s_{\text{BIC}}(G, \text{join}_{\{A',B'\}}[\Pi])$
  if $s_{\text{BIC}}(G, \text{join}_{\{A,B\}}[\Pi]) > s_{\text{BIC}}(G, \Pi)$ then
      $\Pi'$ := greedy-joining(join$_{\{A,B\}}[\Pi]$,$G$)
  else
      $\Pi' := \Pi$
  end if
  return $\Pi'$
```

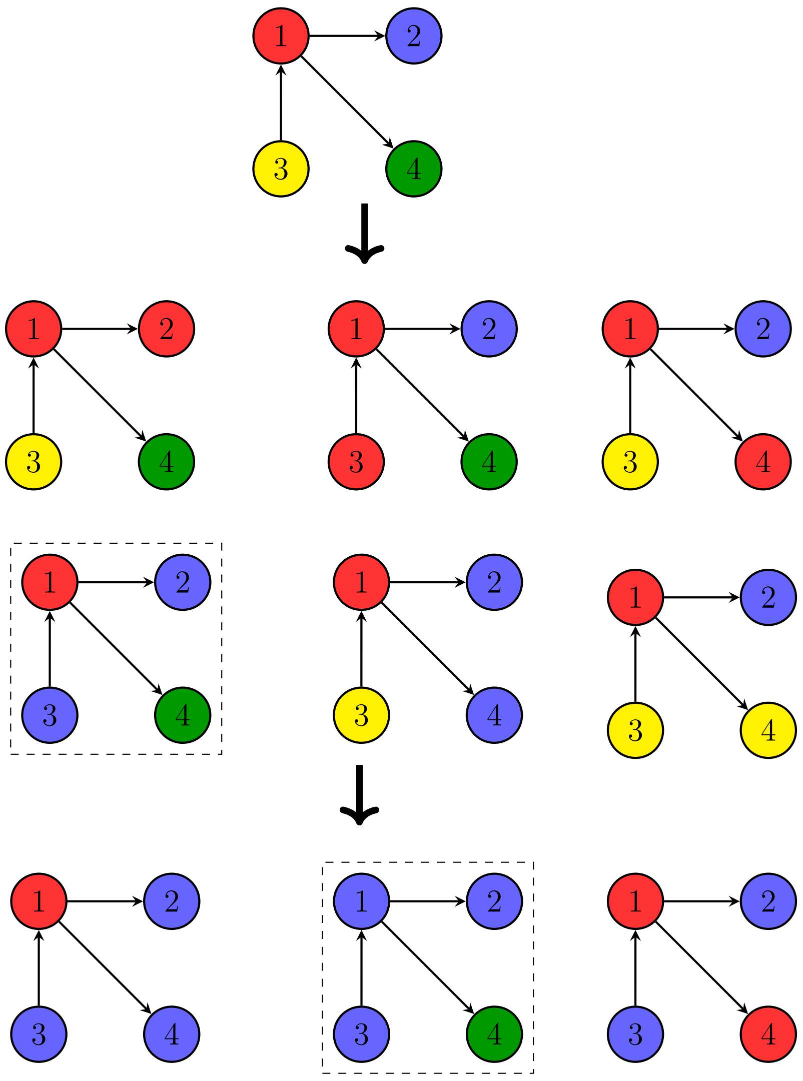


Figure 4.1: Example of a searching scheme in greedy joining algorithm for $p = 4$. Colors indicate partition blocks. We start with the initial partition $\Pi_0 = \{\{1\}, \{2\}, \{3\}, \{4\}\}$ and then construct each member of the local neighborhood by joining one pair of partition blocks. A partition with bounding boxes indicates the biggest improvement in BIC score compared to the currently selected partition. The score of the lastly selected partition $\Pi = \{\{1, 2, 3\}, \{4\}\}$ has to be compared with the full partition $\{\{1, 2, 3, 4\}\}$ in a final step.

Greedy moving algorithm

Another approach is the *greedy moving algorithm.* Starting from any initial partition, we recursively move nodes of partition blocks to another partition block until there is no improvement in the BIC score. Let Π be a partition of the vertex set V and $i \in V$. Also, let $U \in \Pi : i \notin U$ be any partition block that does not include node i already. We define the partition obtained by moving node i to partition block U :

$$\begin{aligned}\text{move}_{i,U}[\Pi] := \Pi \setminus \{U\} \setminus \{W \in \Pi \mid i \in W\} \\ \cup \{U \cup \{i\}\} \cup \bigcup_{\{W \in \Pi \mid i \in W \wedge \{i\} \neq W\}} \{W \setminus \{i\}\}.\end{aligned}$$

Having chosen the starting partition, the local neighborhood consists of all partitions that can be constructed by moving one node to another partition block. Figure 4.2 shows a schematic example of a local neighborhood. Analogous to the greedy joining algorithm, we calculate the BIC score of all local neighborhood states and choose the highest-scoring one. If the score exceeds the current state, we move to the new partition and repeat all steps until no improvement is possible. When the last element of a partition block is moved out of the subset, the number of partition blocks decreases. The DAG is fixed before and does not change during the procedure. Compared to the finite amount of local steps in the greedy joining algorithm, greedy moving offers a more flexible searching scheme across partition space.

Algorithm 3 greedy-moving(Π,G)

```
Require: Initial partition $\Pi$, fixed DAG $G$, data $X$
  Choose $(i, U) \in \underset{(i',U') \in V \times \Pi \setminus \{W \in \Pi \mid i' \in W\}}{\arg\max} s_{\text{BIC}}(G, \text{move}_{i',U'}[\Pi])$
  if $s_{\text{BIC}}(G, \text{move}_{i,U}[\Pi]) > s_{\text{BIC}}(G, \Pi)$ then
      $\Pi' :=$ greedy-moving(move$_{i,U}[\Pi]$,G)
  else
      $\Pi' := \Pi$
  end if
  return $\Pi'$
```

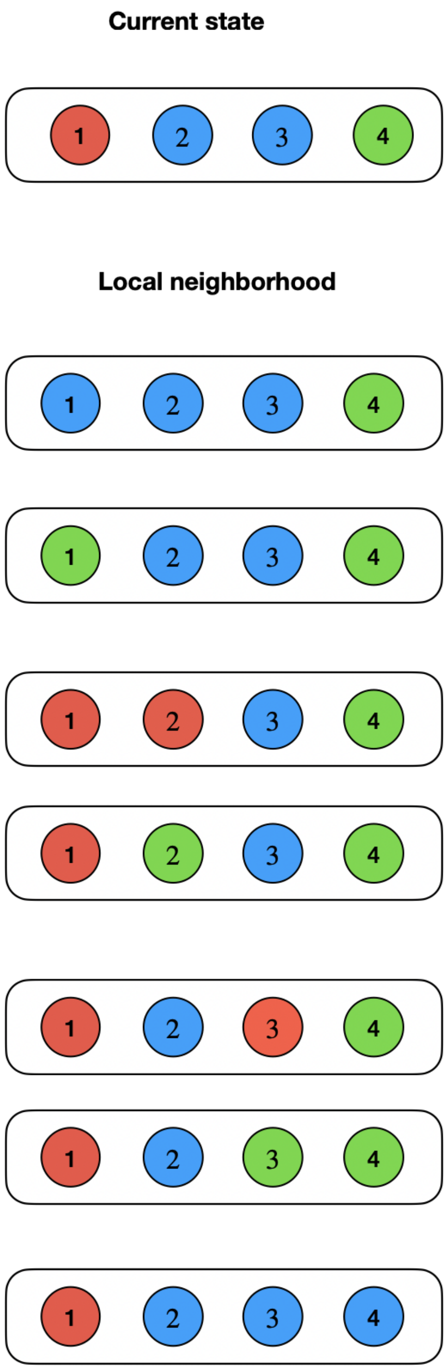


Figure 4.2: Local neighborhood in greedy moving algorithm for partition $\Pi = \{\{1\},\{2,3\},\{4\}\}$. The neighborhood states are all partitions created by moving one node to another partition block in Π.

4.3 Search Scheme in DAG space

After inspecting searching schemes for the partition while keeping the DAG fixed, we now look at the converse. Maximizing the BIC score over the space of all DAGs becomes infeasible since the number of DAGs grows exponentially with increasing number of nodes p. Robinson (1973) shows that the number of DAGs D_p on p nodes is given by the recurrence relation

$$D_p = \sum_{k=1}^{p} (-1)^{k-1} \binom{p}{k} 2^{k(p-k)} D_{p-k} \qquad p = 0, 1, 2, 3, \dots .$$

For $p = 8$ we already get $D_8 = 7.8 \times 10^{11}$ possible DAGs (OEIS Foundation Inc. (2023)). Hence, we adopt a greedy searching scheme also for DAGs.

Given any arbitrary DAG, we define the local neighborhood of that DAG as the set of DAGs that can be obtained by adding, deleting, or removing one edge. We call these *operators*. Let neighborhood(G) denote the set of all neighbors of DAG G. We may restrict the local neighborhood by a random subset of a given size bound or some threshold for the number of parents. So, starting from an initial DAG, we compare the BIC score of the current DAG with the score of all DAGs in the local neighborhood and move to the state with the biggest gain in BIC score, if possible. The algorithm stops when no score improvement is achievable. It is important to recall that the partition of the DAG is fixed before the procedure and does not change. Figure 4.3 shows an example of a local neighborhood in DAG-Space.

Algorithm 4 greedy-DAG-search(G,Π)

```
Require: Initial DAG $G$, fixed partition $\Pi$, data $X$
  Choose $\widetilde{G} \in \underset{G' \in \text{neighborhood}(G)}{\arg\max} \; s_{\text{BIC}}(G', \Pi)$
  if $s_{\text{BIC}}(\widetilde{G}, \Pi) > s_{\text{BIC}}(G, \Pi)$ then
      $G^* :=$ greedy-DAG-search($\widetilde{G}$,$\Pi$)
  else
      $G^* := G$
  end if
  return $G^*$
```

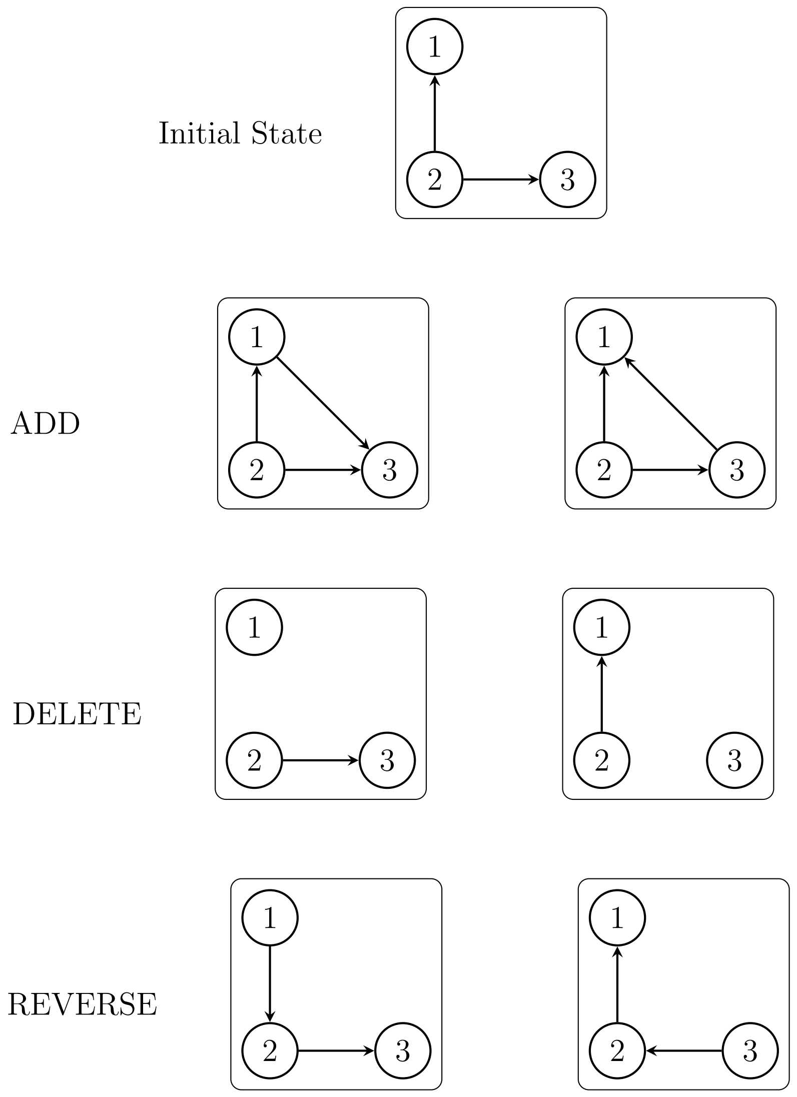


Figure 4.3: States resulting from applying one of the operators on a DAG

4.4 Constructing Greedy Search for both DAG and Partition

In the previous sections, we presented greedy search for learning the partition and DAG from data. In this section, we use these as components to build advanced methods to estimate the CPDAG under partition knowledge. DAG and partition are estimated in each method, and the CPDAG is constructed by applying Algorithm 1. The main idea is to exploit partition information through variations of greedy search schemes. We develop three approaches, which will be evaluated on simulated datasets in the next chapter.

4.4.1 Baseline Estimation Technique

The first method we are looking at combines commonly used methods for structure learning with greedy search strategies introduced in the preceding sections. We first learn the DAG using GES or PC, followed by the greedy joining algorithm to estimate the partition. Afterward, we fine-tune the results using alternated greedy partition and DAG steps.

The whole procedure has two phases and can be described as follows:

Phase I. We first learn the CPDAG from data using PC or GES and then randomly choose a representative DAG of that estimated equivalence class. Fixing that DAG, we run the greedy joining algorithm(Algorithm 2) to obtain an estimate of the partition of the vertex set.

Phase II. Starting from the estimates G^* and Π^* of Phase I, we execute alternating greedy DAG and partition steps until no improvement of the BIC score is possible: Initializing G^* and Π^*, we first fix the partition Π^* and construct the local neighborhood of DAG G^* (Section 4.3). We now compare the BIC score of G^* given Π^* with the score of each neighbor given Π^*. If no improvement is possible, the whole procedure stops. If possible, we move to the neighbor with the biggest gain in BIC score. Now we fix that DAG and construct the local neighborhood of partition Π^* obtained by doing a greedy moving step (Section 4.2). We now compare the BIC score of Π^* given G^* with the score of each partition neighbor given G^*. Again, if no improvement is possible, the whole procedure stops. If possible, we move to the neighbor with the biggest gain in BIC score. We fix the new partition and continue with the greedy DAG step again. It is important to stress that we do not run an entire greedy moving algorithm or DAG search at each point, which might consist of multiple consecutive movements across partition or DAG space. Only a single greedy moving or greedy DAG step is executed. This alternation of single partition and DAG steps is repeated until either one does not improve the BIC score. The algorithm returns DAG and partition at the point of termination.

The first phase gives us a baseline estimation of the DAG and its corresponding partition of the vertex set. The second phase aims to fine-tune the results from the first phase by alternating greedy DAG and partition steps. The goal is to improve the results from the first phase by adding local changes to the DAG and the partition. Figure 4.4 shows a sketch of the whole procedure.

4.4.2 Simultaneous DAG and Partitions Steps

The next method takes a different approach. Instead of starting with a baseline estimation using known learning methods, we can first initialize DAG and partition in two different ways for greedy search :

1. Empty DAG and the finest partition possible as a starting point. We refer to this initialization of the partition as an "empty" partition.
2. Random initiation of DAG and partition: In Section 5.1 we will see how to generate a random DAG. To generate a random partition, we sample uniformly p times with replacement from vector $(1, 2, \dots, p)$, where p is the number of variables. Looking at our sampled vector, variables with the same number on their position are grouped into one partition block. For example, the sampled vector (1,4,4,1) indicates that we have two partition blocks with each two variables.

After initialization, given the data, we proceed with a greedy search on both DAG and partition. But rather than doing alternated greedy DAG and partition steps, we search simultaneously on DAG and partition space. More precisely, the local neighborhood of each current state consists of all pairwise combinations of local neighbors in a greedy DAG step (Section 4.3) and local neighbors of a greedy moving step (Section 4.2).

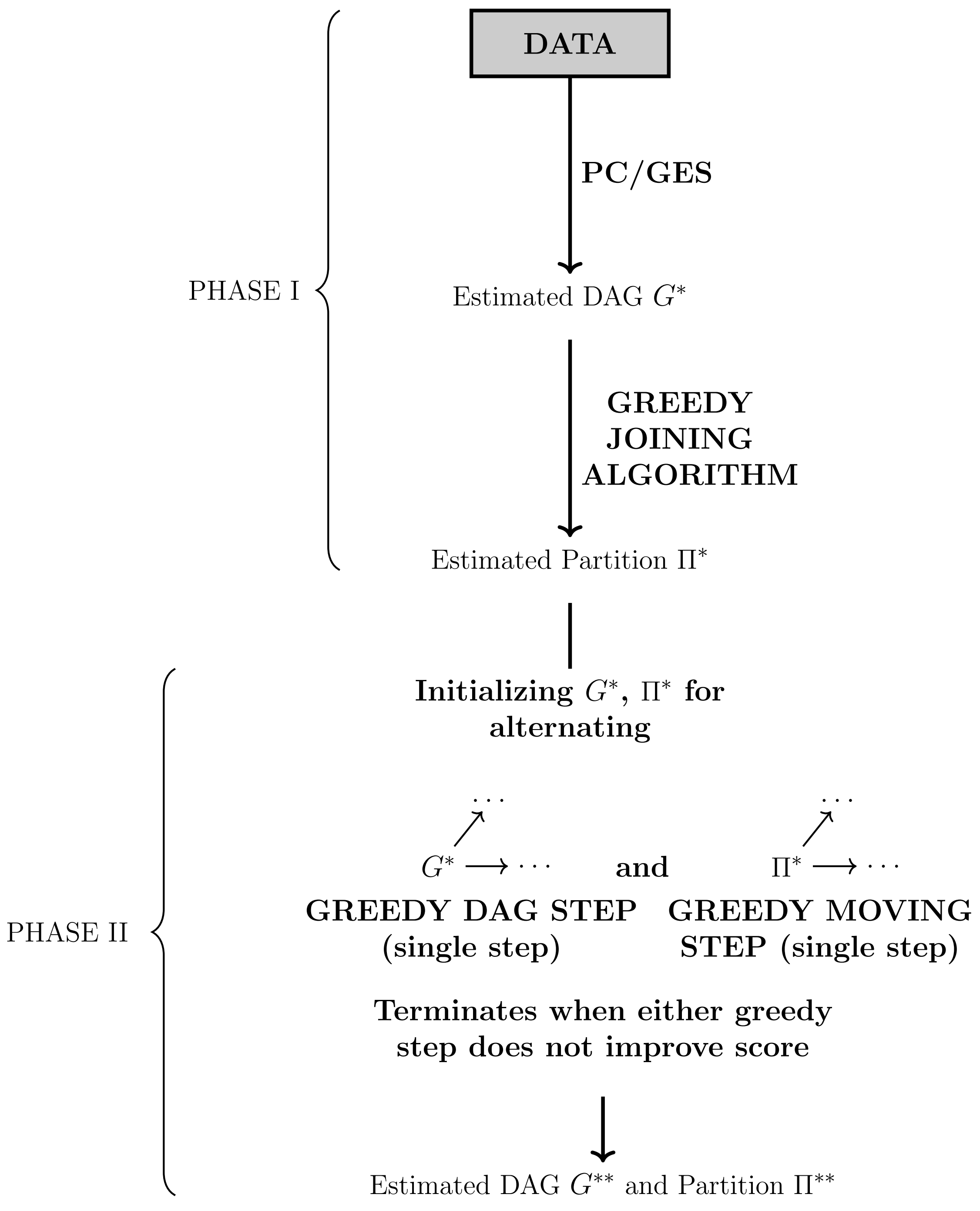

DATA
PC/GES
Estimated DAG G^*
PHASE I
GREEDY JOINING ALGORITHM
Estimated Partition Π^*
Initializing G^*, Π^* for alternating
$G^* \longrightarrow \cdots$
and
$\Pi^* \longrightarrow \cdots$
PHASE II
GREEDY DAG STEP (single step)
GREEDY MOVING STEP (single step)
Terminates when either greedy step does not improve score
Estimated DAG G^{**} and Partition Π^{**}

Figure 4.4: Baseline estimation technique

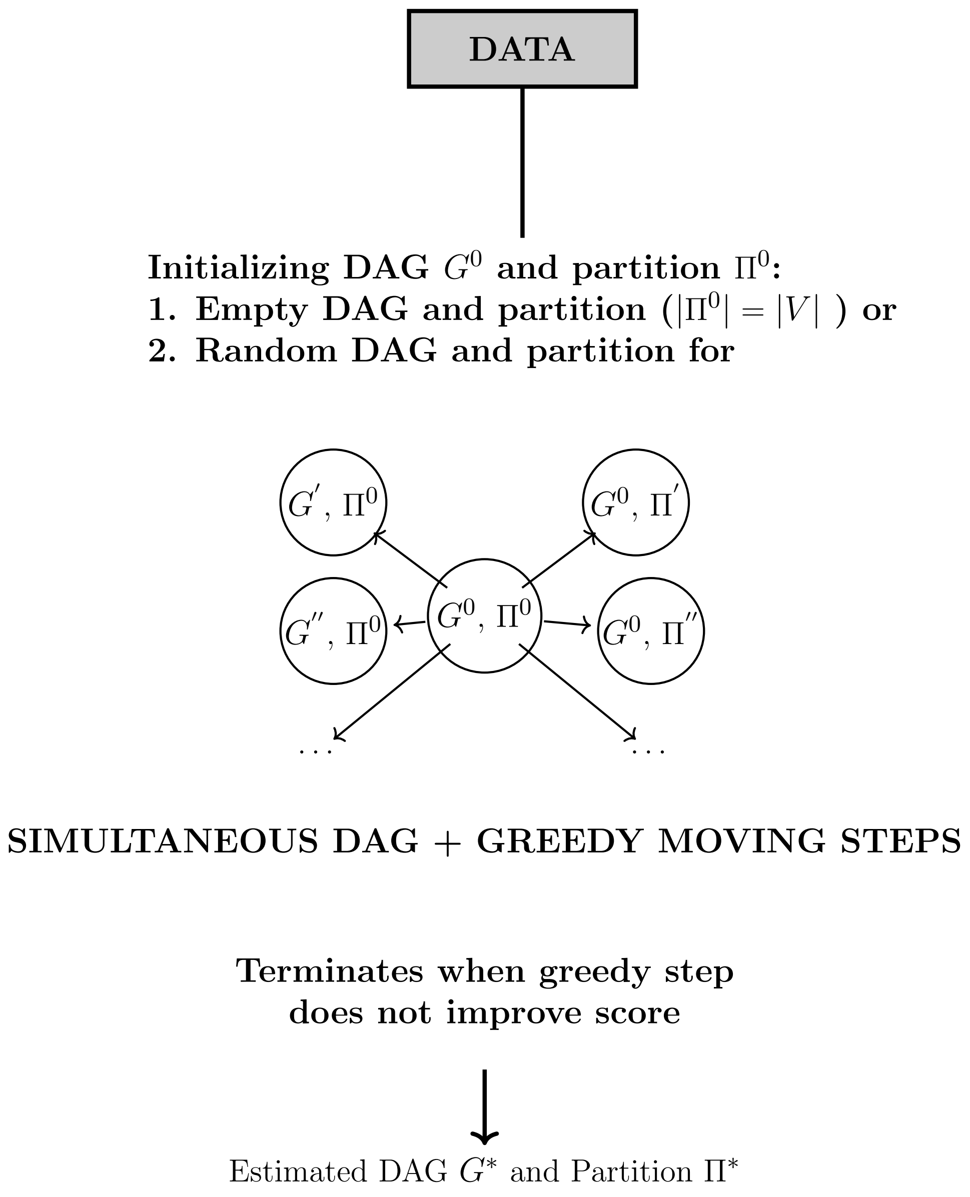

DATA
Initializing DAG G^0 and partition Π^0:
1. Empty DAG and partition ($|\Pi^0| = |V|$) or
2. Random DAG and partition for
G', Π^0
G^0, Π'
G'', Π^0
G^0, Π^0
G^0, Π''
...
...
SIMULTANEOUS DAG + GREEDY MOVING STEPS
Terminates when greedy step
does not improve score
Estimated DAG G^* and Partition Π^*

Figure 4.5: Estimation using simultaneous DAG and partition steps

The combined DAG plus partition neighbor, which leads to the biggest gain in the BIC score, is chosen as the following state. If no gain is possible, we terminate the greedy search. We move across DAG plus partition space through different partially homoscedastic models. Carrying out greedy joining steps (Section 4.2) instead would not be suitable in this design, as it would strongly limit the overall number of possible DAG movements. Figure 4.5 gives an overview of the method.

The size of the local neighborhood of a current state is the size of the DAG neighborhood times the size of the neighborhood obtained by a greedy moving step on the partition. A shortcoming is that for a large number of variables, we have to calculate the BIC score for a potentially enormous neighborhood. For example, with $p = 30$ and empty initiation, the local neighborhood at the first step consists of 378 450 DAG plus partition pairs.

4.4.3 Alternating DAG and Partition Search

The last technique we describe contains elements of both methods we have looked at so far. Analogously to the last method, we can first initialize DAG and partition as either empty or random. DAG and partition are then learned from data using an alternation of greedy DAG search (Algorithm 4) and greedy moving algorithm (Algorithm 3).

In detail, we first apply DAG search on the initialized DAG while fixing the initialized partition. Provided that the score can be improved, the obtained DAG is fixed, and we apply the greedy moving algorithm on the partition. Again, provided that the score can be improved, we fix the obtained partition and run the DAG search again, proceeding until neither greedy search improves the score. In summary, we alternate between greedy DAG search and greedy moving search, terminating when either algorithm executes no single greedy step. It is important to emphasize the difference between the procedure here and Phase II of the Baseline estimate technique (Section 4.4.1). In the latter one, we alternate between single DAG and greedy moving steps. Here, we alternate between possibly multiple consecutive DAG and greedy moving steps. Figure 4.6 shows a sketch of the procedure.

Each of these three proposed methods outputs an estimated DAG and partition of the variable set. The CPDAG under partition information is then obtained by applying Algorithm 1 in a final step.

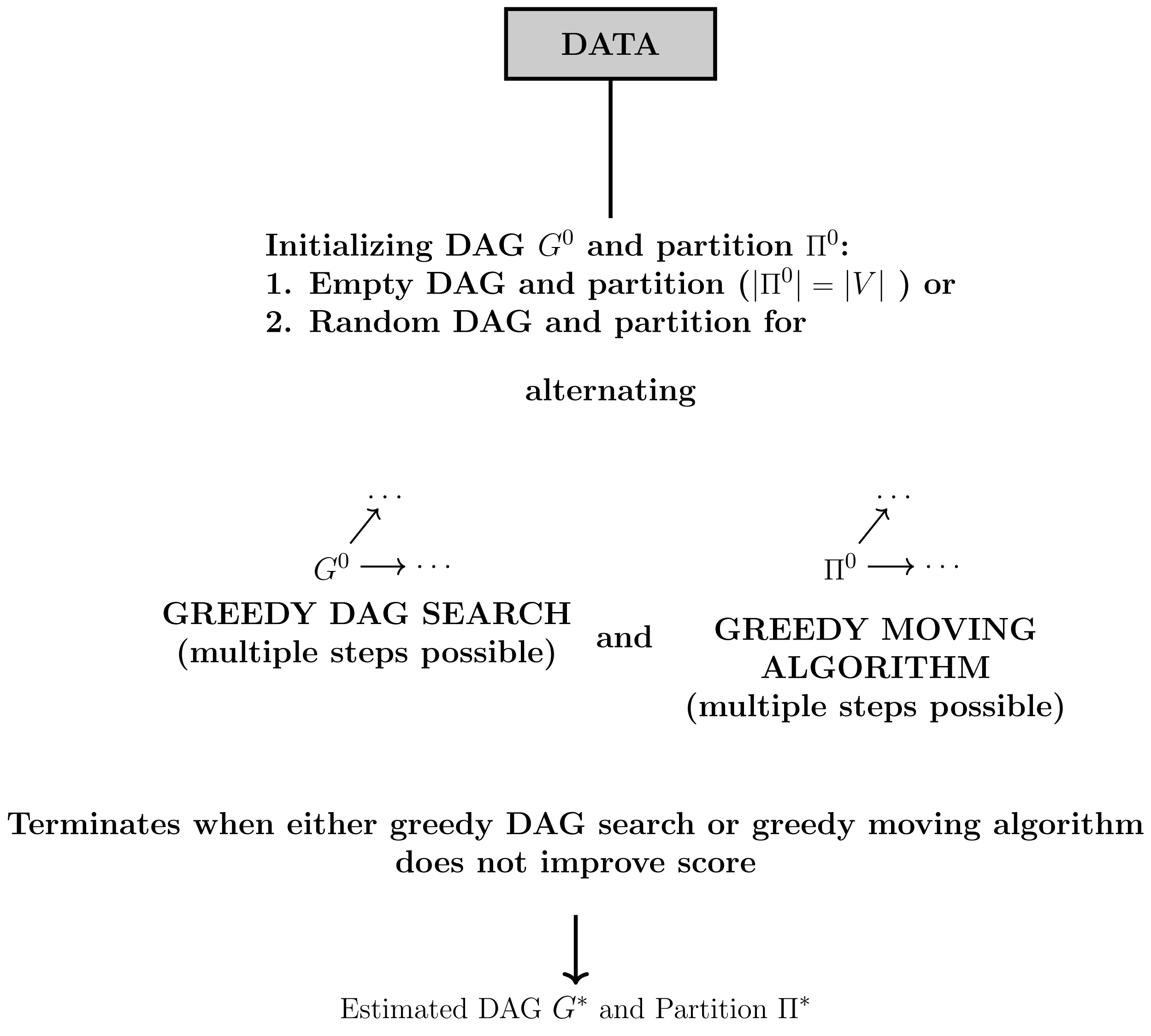

DATA
Initializing DAG G^0 and partition Π^0:
1. Empty DAG and partition ($|\Pi^0| = |V|$) or
2. Random DAG and partition for
alternating
$G^0 \longrightarrow \cdots$
$\Pi^0 \longrightarrow \cdots$
GREEDY DAG SEARCH (multiple steps possible)
and
GREEDY MOVING ALGORITHM (multiple steps possible)
Terminates when either greedy DAG search or greedy moving algorithm does not improve score
Estimated DAG G^* and Partition Π^*

Figure 4.6: Estimation using alternated DAG and partition search

5 Numerical Experiments

In this chapter, we evaluate the algorithms developed in the last chapter on synthetically created datasets. In addition to comparing performance, we also give additional insights into the number of greedy steps and runtime. We use the programming language R for generating data and implementing the algorithms.

5.1 Simulated Data

We start by describing the process of creating our simulated datasets.
To generate n i.i.d samples from a partially homoscedastic linear Gaussian SEM, we do the following steps :

1. Given the number of variables p, we generate a random DAG: First, we determine the sparsity of the DAG by distinguishing two different settings. In the sparse setting, the adjacency probability between every pair of nodes is $prob = 3/(2p-2)$, whereas in the dense scenario, the probability is set to 0.3. For each (i, j) pair with $i < j$, we generate independent uniform random variables $U_{ij} \sim U(0,1)$ and add the edge $i \to j$ if $U_{ij} < prob$. Once we have traversed all node pairs, we randomly shuffle the node labels.

2. The edge weights corresponding to the DAG are then uniformly drawn from $[-1,-0.3] \cup [0.3,1] \Rightarrow \Lambda^*$ coefficient matrix.

3. Given the number of partition blocks, we generate a random partition Π of the vertex set $\{1,\dots,p\}$ (as described in Section 4.4.2). The error variance of each block is then uniformly drawn from $[0.3,1] \Rightarrow w^*$ vector of error variances.

4. We simulate n i.i.d samples from from normal distribution $\mathcal{N}(0,\Sigma^*)$, where $\Sigma^* = \phi_G(\Lambda^*, w^*)$.

The CPDAG under partition information is obtained by applying Algorithm 1 on DAG and partition generated.

5.2 Results

5.2.1 Estimating Partition Blocks

We first look at how well the greedy joining algorithm (Algorithm 2) and greedy moving algorithm (Algorithm 3) perform at learning the correct partition blocks from data. The dataset is created as explained in Section 5.1, where we set following configurations :

- $n = 1000$,
- $p \in \{10, 15, 20, 25, 30\}$, where corresponding number of partition blocks is $\lceil p/3 \rceil + 1$, i.e., the number of blocks increases with the number of variables,
- sparse graphs.

We then evaluate the number of correctly learned blocks of a partition for three different settings. The first setting involves fixing the true DAG, while in the other two settings, the fixed DAG is learned using the GES and PC algorithm (more precisely, GES and PC algorithm return CPDAG, and we randomly pick a representative DAG of that corresponding Markov equivalence class). Important to mention is that the initialization of the partition differs: In the greedy joining algorithm, we start with the finest partition possible, while for the greedy moving algorithm, we initialize a random partition (described as in Section 4.4.2). The simulation runs 100 times for each configuration.

Figure 5.1 and Figure 5.2 show the experiment results. As expected, choosing the true DAG for the partition search leads to better performance, i.e., a higher number of correctly learned partition blocks. We can also observe the overall trend, especially for the case of fixed DAG learned by PC and GES, that with an increasing number of variables (and partition blocks), fewer partition blocks are correctly estimated. Especially looking at $p = 20$ (8 blocks) and higher, we can see that mostly never more than four correct blocks are learned. This indicates that both partition searches reach a limit when inferring a larger number of partition blocks. Comparing both algorithms, we observe a slightly better performance for the greedy moving algorithm.

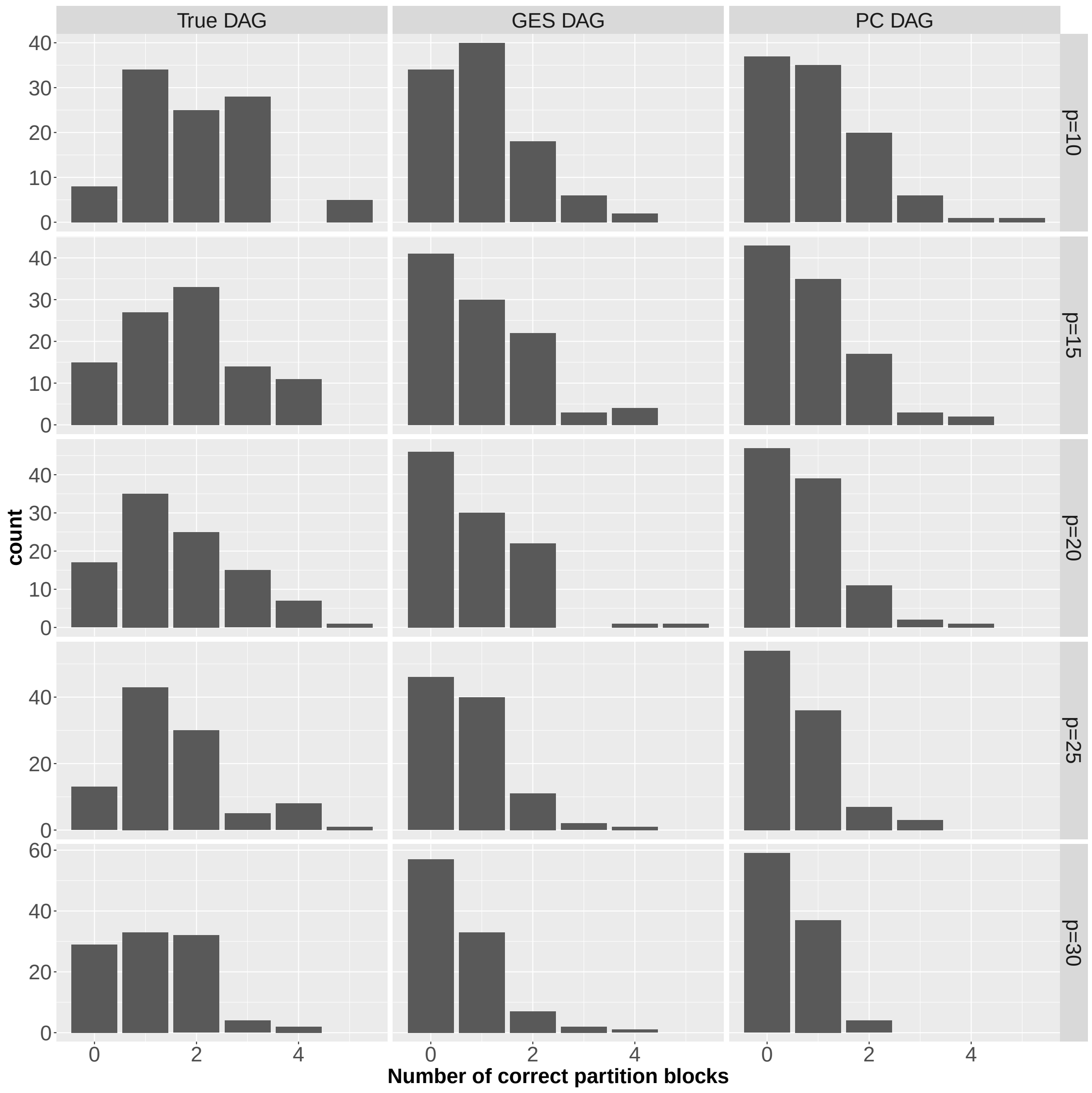


Figure 5.1: Barplots showing number of correctly predicted partition blocks for greedy joining algorithm by p and type of fixed DAG. Sparse graph setting with $\lceil p/3 \rceil + 1$ blocks and $n = 1000$.

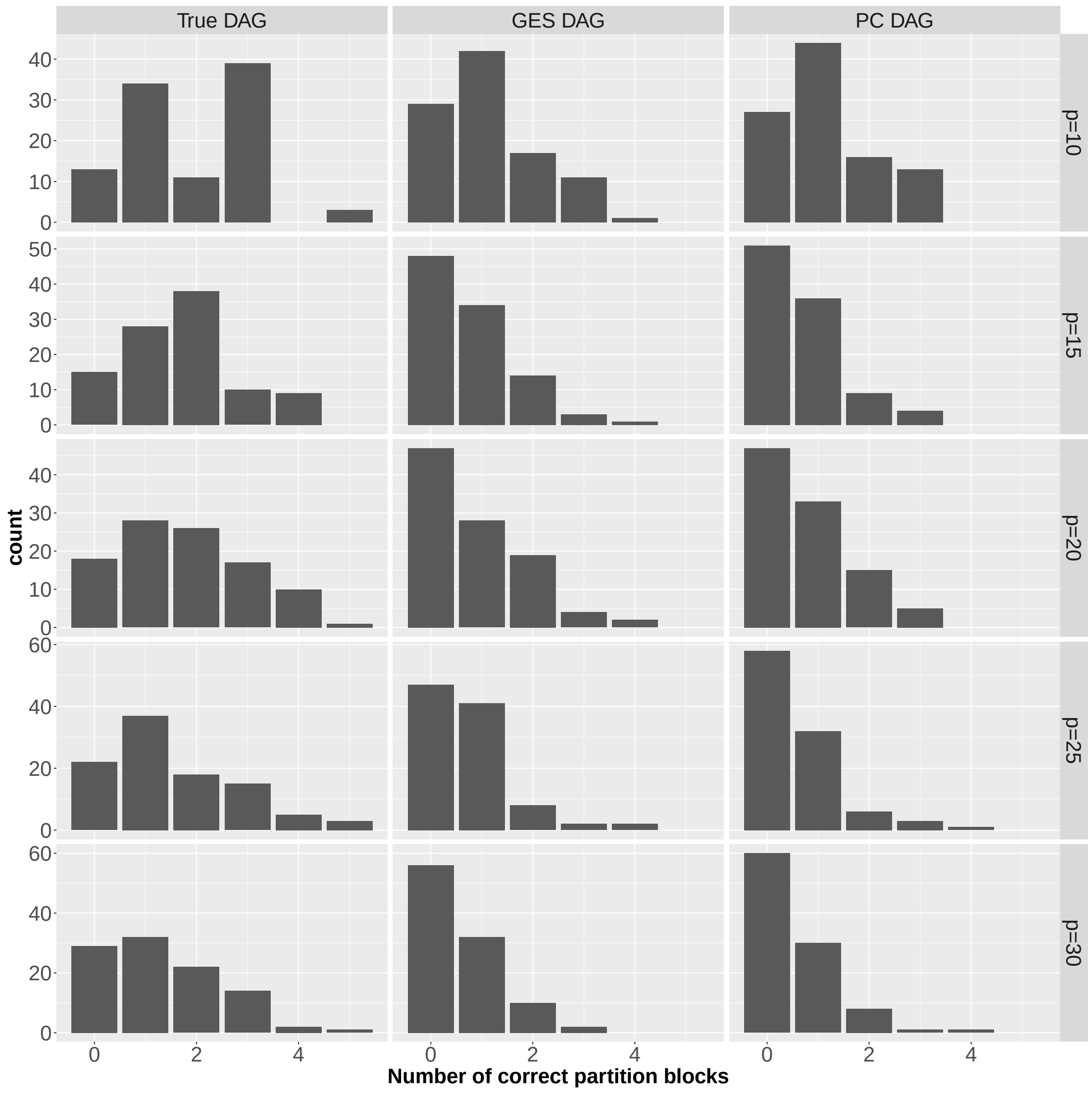


Figure 5.2: Barplots showing number of correctly predicted partition blocks for greedy moving algorithm by p and type of fixed DAG. Sparse graph setting with $\lceil p/3 \rceil + 1$ blocks and $n = 1000$.

5.2.2 Estimating CPDAG under Partition Information

Next, we investigate the performance of the methods in Section 4.4 in estimating the true CPDAG under partition knowledge. Throughout this section, we will compare these methods to known state-of-the-art algorithms GES and PC on our datasets simulated from partially homoscedastic linear Gaussian SEMs. We want to observe to what extent the incorporation of partition information in our developed methods can leverage the results compared to the standard methods that do not integrate the grouping of error variances into their algorithms.

As an error metric, we adopt the modified structural Hamming distance (SHD) from Peters and Bühlmann (2014). The traditional structural Hamming Distance (SHD) measures the total number of edge additions, deletions, and reversals required to transform one graph into another, i.e., all edge mistakes count as 1. The difference between an undirected edge and a directed edge also counts as 1. Considering the crucial role of parental information in our partially homoscedastic setup, we use the modified version, where a distance of 2 is assigned to each pair of reversed edges. Consequently, we evaluate our methods by calculating the SHD between true CPDAG and estimated CPDAG.

The PC algorithm uses conditional independence tests to learn edges in the graph, where α denotes the significance level. To compare score-based methods and constraint-based methods, we consider a range of values for α from 10^{-5} to 0.8, increasing by the ratio 1.1 (Harris and Drton (2013)). The value of α with the maximum BIC score is picked. We will do this throughout all experiments.

We also track the runtime performance of several methods. The following experiments were conducted on a Macbook Air (2020) with an M1 processor and 8GB RAM.

Baseline estimation technique

We start by evaluating the Baseline estimation technique (Section 4.4.1), where we further will compare the CPDAG estimate after Phase I to the CPDAG estimate after Phase II of the algorithm and assess how much fine-tuning through alternated greedy DAG and partition steps improves the result.
As explained in 4.4.1, Phase I consists of learning the DAG by GES or PC and then running the greedy joining algorithm. We call these procedures `ges-ph1` and `pc-ph1` respectively.

We set the following configurations :

- $n \in \{100, 500, 1000\}$,
- $p \in \{6, 12, 18, 30\}$, where corresponding number of partition blocks is $\lceil p/3 \rceil + 1$, i.e., the number of blocks increases with the number of variables.
- $sp \in \{sparse, dense\}$, where sp stands for sparsity of generated DAG.

The simulation runs 100 times for each configuration. Figure 5.3 and Figure 5.4 show the box plots of the SHD between true CPDAG and estimated CPDAG for different methods. In the sparse setting across all number of variables (except $p = 12$) and sample size $n = 500$ and $n = 1000$, we can see that `pc-ph1` performs marginally better than PC. Here, the exploitation of partition information gives `pc-ph1` only a small advantage. The same statements can be made about `ges-ph1` for a smaller number of variables ($p = 6$ and $p = 12$).
In the setting with dense graphs, however, one can observe little to no difference between the Phase I results and PC/GES. As expected, the dense setting is more challenging.

We saw in the results that learning the DAG by PC or GES and then running the greedy joining algorithm can only achieve slightly better results in estimating the CPDAG under the true partition than the common methods. Consequently, we want to evaluate the results after adding a fine-tuning phase (Phase II). We focus on the PC-based method since PC concentrates on testing conditional independence statements, and we know that in partially homoscedastic models, the conditional independence statements are not altered (Proposition 3.8). We refer to the method combining Phase I and II as `pc-ph2`.

In the next simulation, the performance between `pc-ph1` and `pc-ph2` in estimating the CPDAG is compared. The same configurations of n and p are used, but only at the sparse graph setting. Again, the simulation runs 100 times for each configuration. It is important to add that the local neighborhood size in each DAG step in `pc-ph2` is restricted to 300.

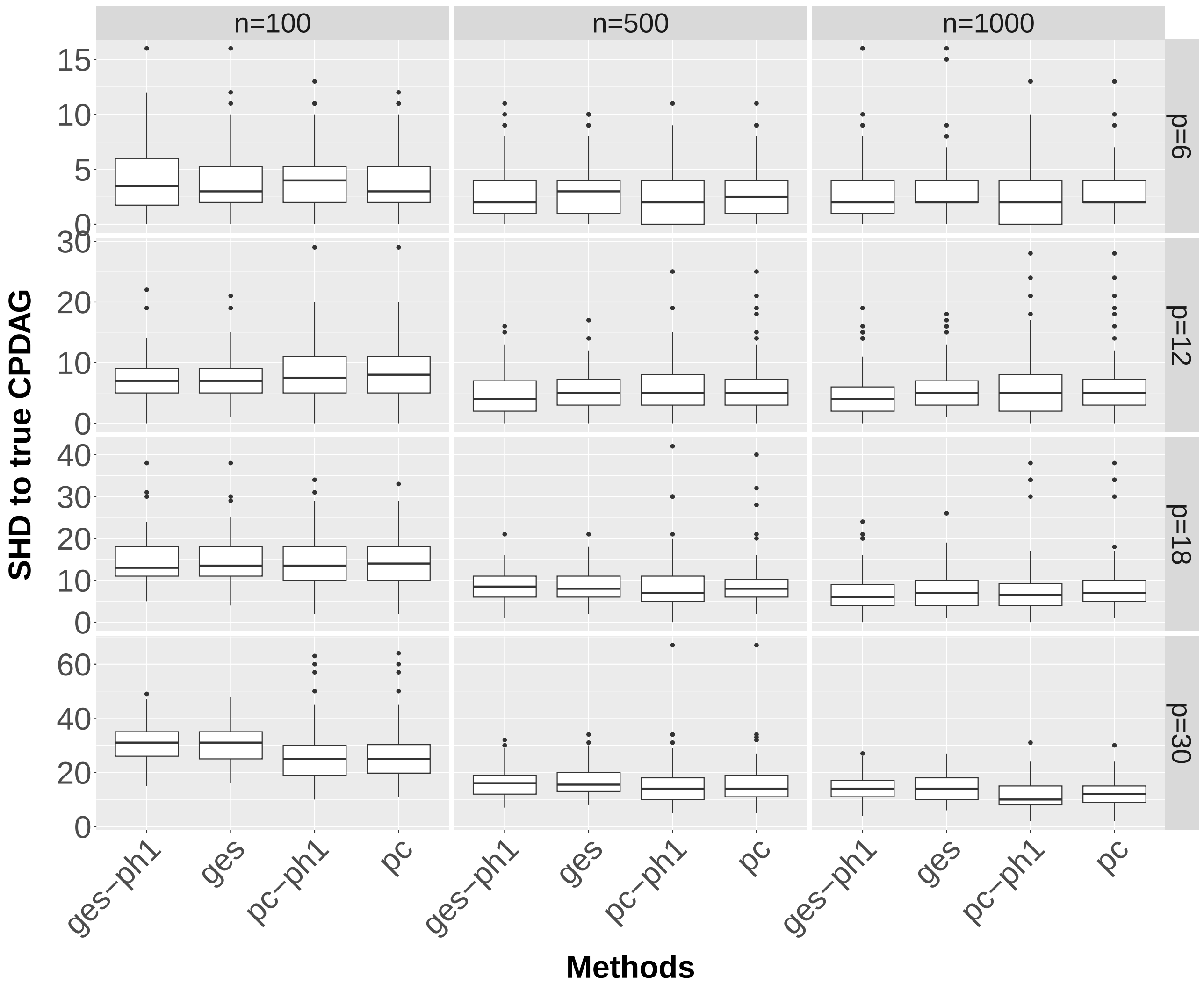


Figure 5.3: Box-Plots of SHD to true CPDAG by p and n for sparse graphs and $\lceil p/3 \rceil + 1$ blocks. Estimated CPDAG by method in 4.4.1 after Phase I (`ges-ph1`, `pc-ph1`) compared to GES (`ges`) and PC (`pc`).

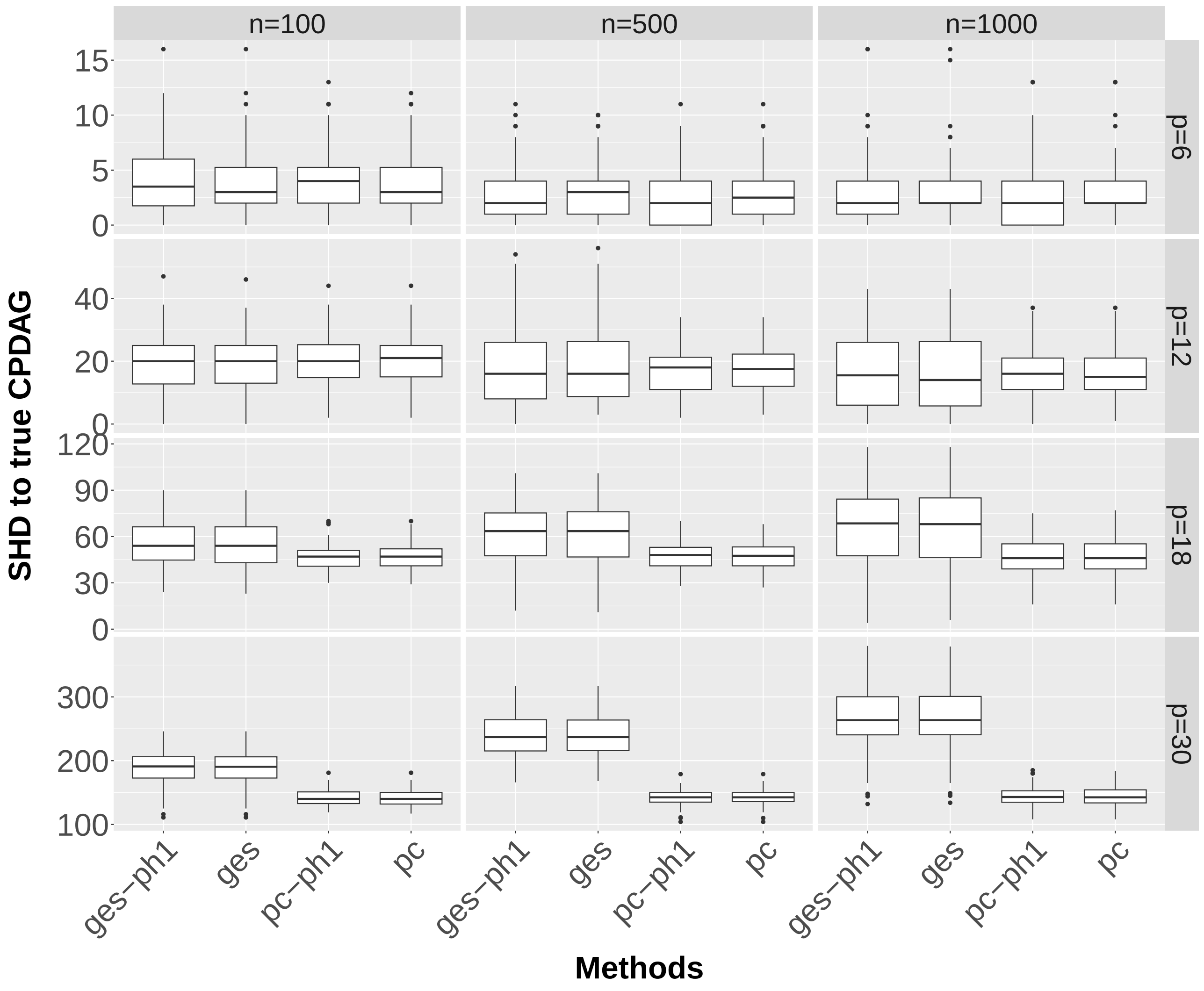


Figure 5.4: Box-Plots of SHD to true CPDAG by p and n for dense graphs and $\lceil p/3 \rceil + 1$ blocks. Estimated CPDAG by method in 4.4.1 after Phase I (`ges-ph1`, `pc-ph1`) compared to GES (`ges`) and PC (`pc`). Note that for $p = 6$, dense graph equals sparse graph in the generation procedure.

Figure 5.5 displays the average SHD between the CPDAGs estimated by `pc-ph1` and `pc-ph2`. The difference decreases when we grow the sample size n and increases across the number of variables p. Figure 5.6 shows the performance results. For $p = 6$ and $p = 12$ across sample sizes $n = 500$ and $n = 1000$, one can see slightly better performance of `pc-ph2` compared to `pc-ph1`, showing that the fine-tuning phase can add relevant local changes to DAG and partition estimated by `pc-ph1`. One can observe no improvement for a higher number of variables, indicating that fine-tuning does not enhance baseline estimation in those cases. Figure 5.7 shows that overall, the number of greedy steps performed in Phase II is small. Figure 5.8 adds information about the running time of the whole procedure. As expected, it increases with sample size and number of variables.
To demonstrate the possible advantage of `pc-ph2` to `pc-ph1` more clearly, we look at an additionally generated simulation where $p = 6$ ($\lceil p/3 \rceil + 1 = 3$ blocks) and $n = 1000$. Figure 5.9 and Figure 5.10 show the results of this example, where estimation of both DAG and partition improved after performing greedy steps in the fine-tuning phase.

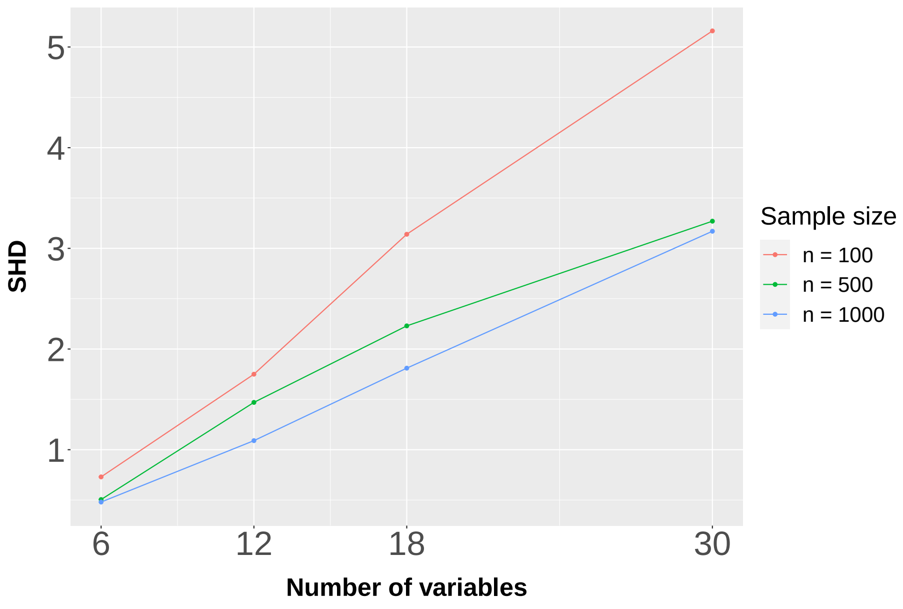


Figure 5.5: Average SHD between CPDAGs estimated by `pc-ph1` and `pc-ph2` by number of variables and sample size

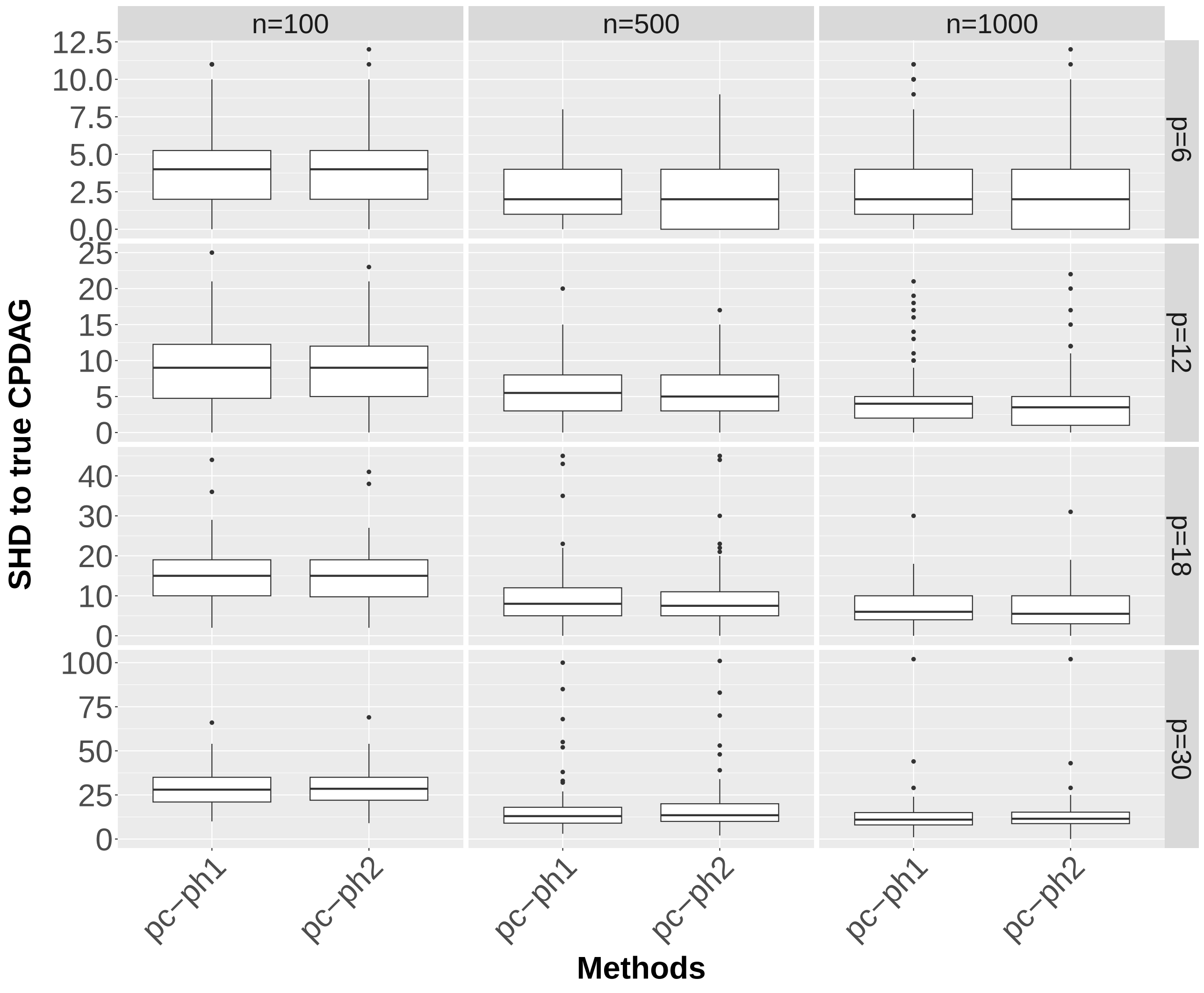


Figure 5.6: Box-Plots of SHD to true CPDAG by p and n in sparse graph setting with $\lceil p/3 \rceil$ +1 blocks. Comparison between estimated CPDAG by method in 4.4.1 after Phase I (`pc-ph1`) and estimated CPDAG after Phase II (`pc-ph2`).

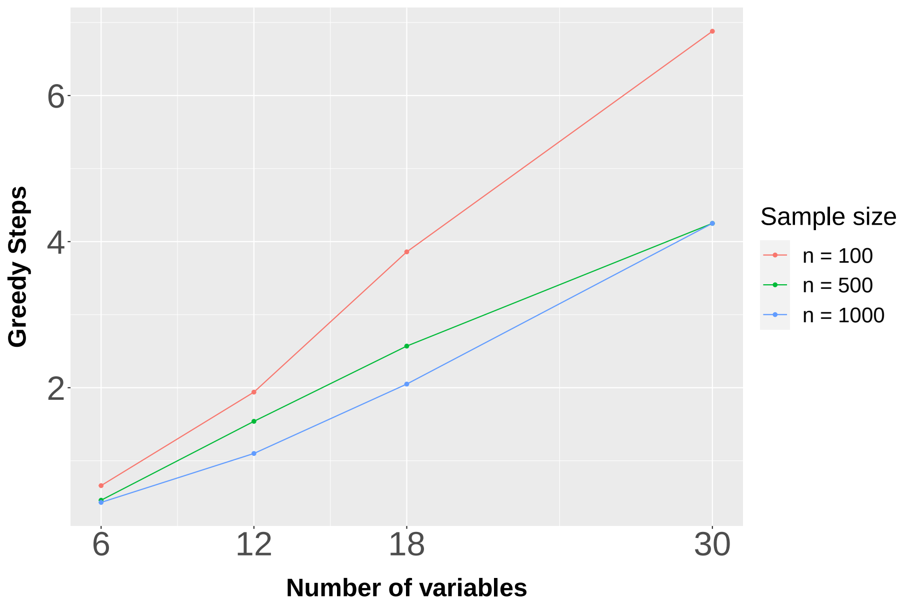


Figure 5.7: Average number of greedy steps (DAG + partition steps) in `pc-ph2` during Phase II by number of variables and sample size.

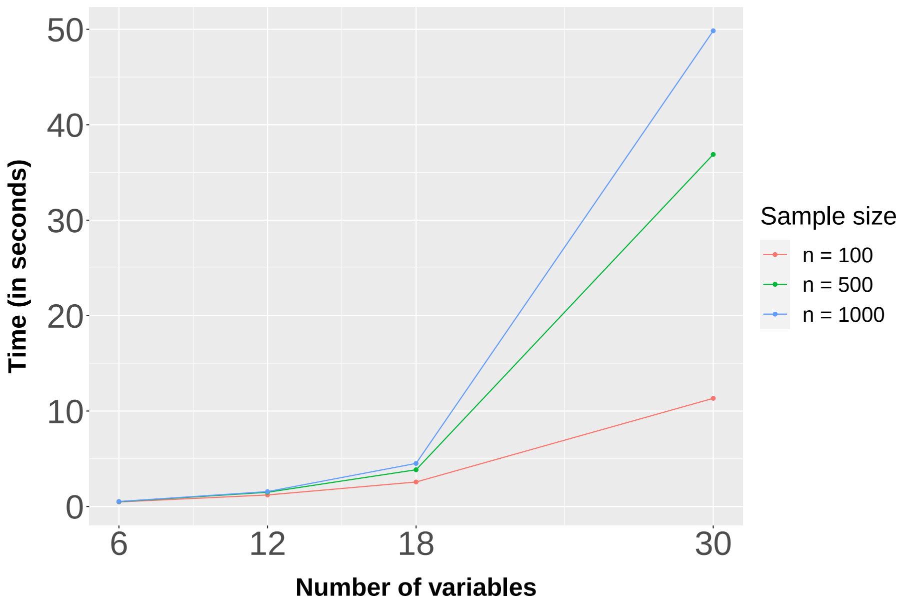
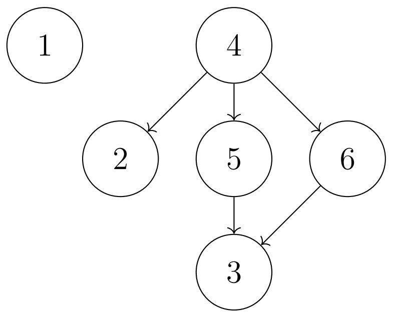
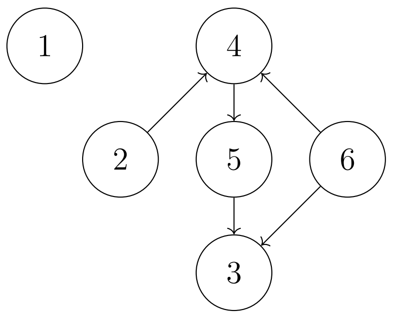


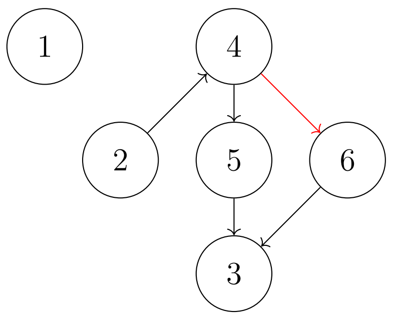
Figure 5.8: Average runtime for `pc-ph2` by number of variables and sample size.



(a) True DAG

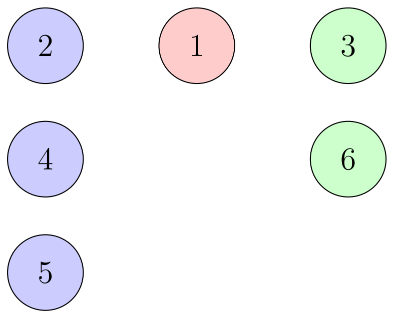


(b) DAG estimated by `pc-ph1`

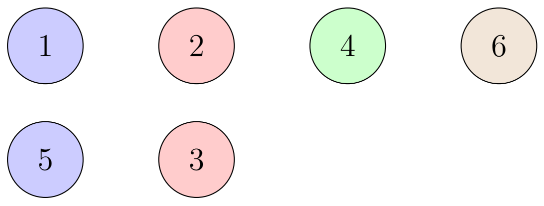


(c) DAG estimated by `pc-ph2`

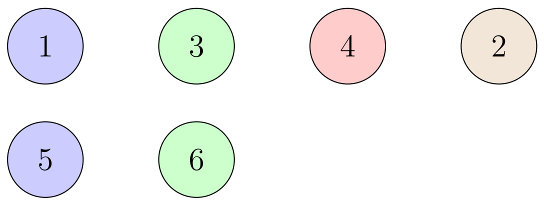
Figure 5.9: Comparison of true DAG with DAGs estimated by `pc-ph1` and `pc-ph2` for one simulation where $p = 6$ and $n = 1000$ in sparse graph setting. One can see that the edge $4 \to 6$ (in red) is correctly reversed during Phase II in `pc-ph2`.



(a) True partition



(b) Partition estimated by `pc-ph1`



(c) Partition estimated by `pc-ph2`

Figure 5.10: Comparison of true partition with partitions estimated by `pc-ph1` and `pc-ph2` for one simulation where $p = 6$ and $n = 1000$ in sparse graph setting. After Phase I in `pc-ph1` no true partition block is learned, but fine-tuning in Phase II leads to one correct block ($\{3,6\}$). Partition blocks are indicated by different colors.

Simultaneous DAG and partitions steps

Next, we observe the results of estimating the true CPDAG using simultaneous DAG and partition steps (Section 4.4.2), where both empty and random initialization are compared. We refer to these as `sim-empty` and `sim-random` respectively. Moreover, we restarted the random initialization five times and chose the result with the highest score. When dealing with a large number of variables, the size of the local neighborhood of a state becomes enormous. Due to the limitation of computational resources for this exhaustive greedy search, only the following configurations were considered :

- $n \in \{100, 500, 1000\}$,
- $p \in \{6, 12\}$, where the corresponding number of partition blocks is $\lceil p/3 \rceil + 1$, i.e., the number of blocks increases with number of variables.
- sparse graphs.

The simulations run 100 times for each configuration. Figure 5.11 displays the Box-Plots of the SHD between true CPDAG and CPDAG estimated by empty and random initialization of the method with PC and GES. As one can see, for high sample size $n = 1000$, both `sim-empty` and `sim-random` underperform across $p = 6$ and $p = 12$. However for $n = 100$ and $n = 500$ with $p = 6$, `sim-empty` performs the best, while for $p = 12$, `sim-random` shows the best results. Table 5.2 shows the average runtime per simulation for $n = 1000$. One can see the extreme increase from $p = 6$ to $p = 12$ for both initialization methods. These results can be explained by the exponential growth of the local neighborhood for DAG and partition when increasing the number of variables. As expected, the number of greedy steps increases from $p = 6$ to $p = 12$ (see Table 5.1).

Table 5.1: Average number of greedy steps by number of variables p and by initialization of method in 4.4.2; $n = 1000$.

	$p = 6$	$p = 12$
sim-empty	5.55	13.14
sim-random	6.65	26.03

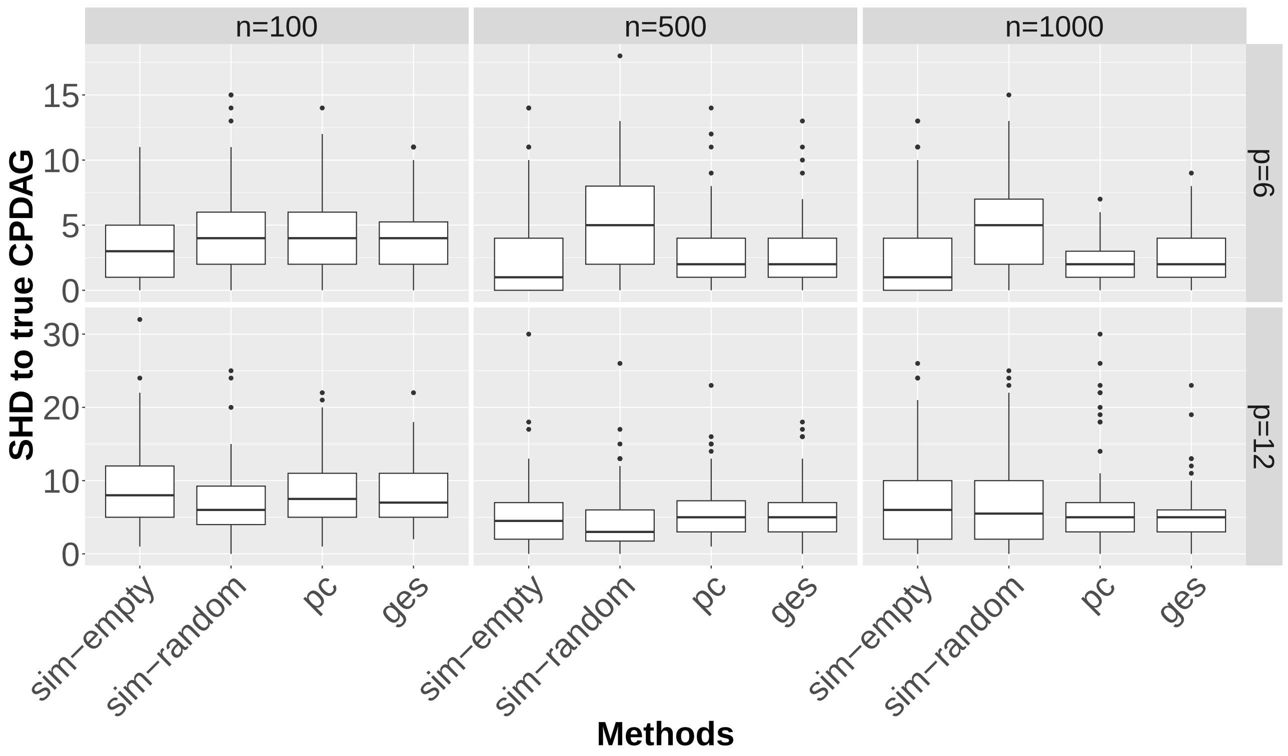


Figure 5.11: Box-Plots of SHD to true CPDAG by p and n for sparse graph setting with $\lceil p/3 \rceil + 1$ blocks. Estimated CPDAG by method in 4.4.2 for empty and random initialization (`sim-empty`, `sim-random`), compared to GES (`ges`) and PC (`pc`).

Table 5.2: Average runtime (in seconds) by number of variables and by initialization for method in 4.4.2, compared to PC and GES; $n = 1000$. For `sim-random` and PC, the runtime is calculated as the cumulative value of all restarts in each simulation.

	$p = 6$	$p = 12$
sim-empty	0.29	21.1
sim-random	1.53	228
pc	0.5	1.49
ges	0.002	0.003

Alternating DAG and partition search

At last, we evaluate the performance of alternating DAG and partition search (Section 4.4.3). Here, we also compare empty initialization with random initialization. We refer to these as `alt-empty` and `alt-random` respectively. For random initialization, we choose the result with the highest score among the five restarts. Similar to the last simulation, the greedy search is exhaustive for a high number of variables. Therefore, we look at the following configurations:

- $n \in \{100, 500, 1000\}$,
- $p \in \{6, 12, 18\}$, where corresponding number of partition blocks is $\lceil p/3 \rceil + 1$, i.e., the number of blocks increases with the number of variables.
- sparse graphs.

The simulations run 100 times for each configuration. Figure 5.12 displays the Box-Plots of the SHD between true CPDAG and CPDAG estimated by empty and random initialization with PC and GES. It is easy to see that `alt-random` achieves remarkable results and performs better than all other methods across all configurations of p and n. On the contrary, `alt-empty` performs worst across all numbers of variables and sample sizes. Table 5.3 gives additional information on the average number of greedy steps. One can see that `alt-random` nearly executes twice as many greedy steps. This supports Table 5.4, where `alt-random` has longer runtime than `alt-empty` , heavily exceeding PC and GES for $p = 12$ and $p = 18$ with $n = 1000$ as expected. The results from Table 5.3 and Table 5.4 indicate that `alt-empty` rather quickly stucks at a local optimum, a possible explanation for the poor performance.

Table 5.3: Average number of greedy steps by number of variables p and by initialization for method in 4.4.3; $n = 1000$. In each simulation, the number of every greedy step of all greedy DAG and greedy moving searches has been summed up.

	$p = 6$	$p = 12$	$p = 18$
alt-empty	7.73	18.15	29.78
alt-random	13.51	35.5	58.07

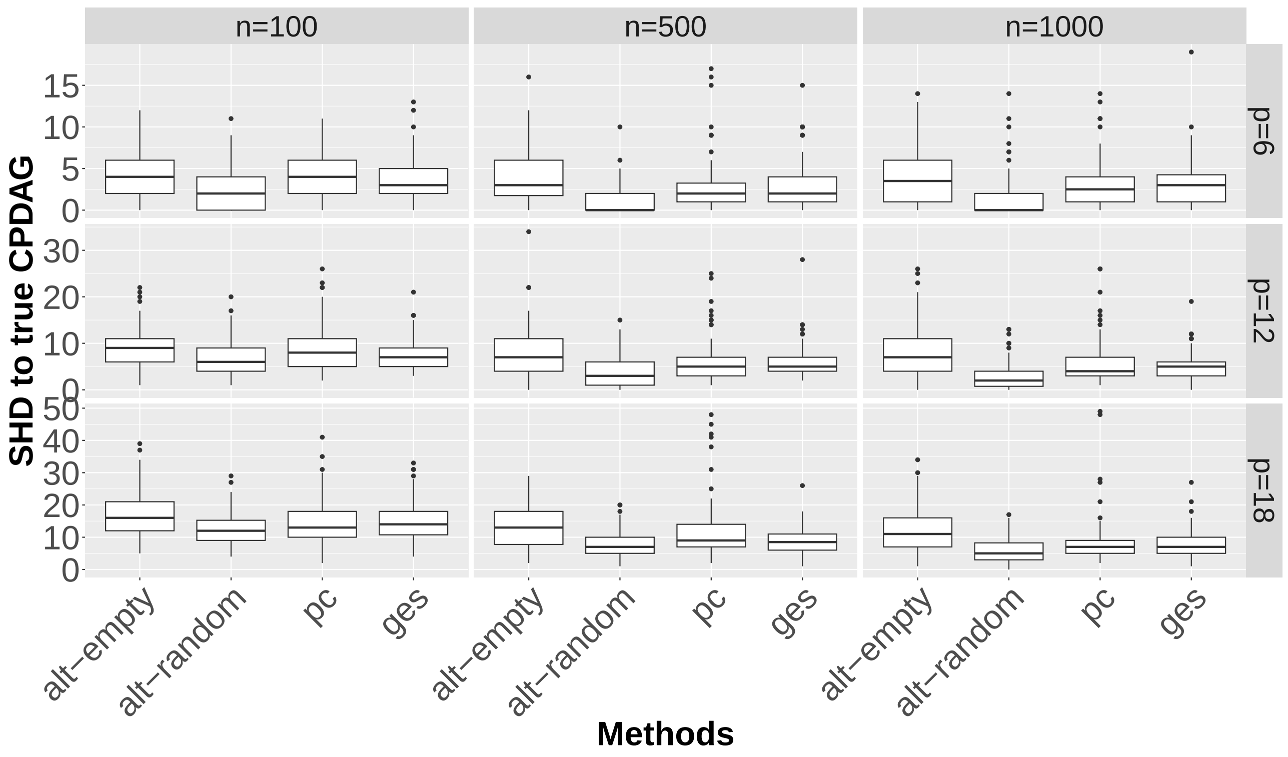


Figure 5.12: Box-Plots of SHD to true CPDAG by p and n for sparse graphs setting with $\lceil p/3 \rceil + 1$ blocks. Estimated CPDAG by method in 4.4.3 for empty and random initialization (`alt-empty`, `alt-random`), compared to GES (`ges`) and PC (`pc`).

Table 5.4: Average runtime (in seconds) by number of variables and by initialization for method in 4.4.3, compared to PC and GES; $n = 1000$. For `alt-random` and PC, the runtime is calculated as the cumulative value of all restarts in each simulation.

	$p = 6$	$p = 12$	$p = 18$
alt-empty	0.027	0.38	1.93
alt-random	0.27	5.54	30.36
pc	0.52	1.37	4.24
ges	0.002	0.003	0.005

6 Conclusion

In this thesis, we examined the partially homoscedastic linear Gaussian SEM. A newly studied setup which falls between the classical case of linear Gaussian SEM with arbitrary error variances and the recently considered case of equal error variances. One theoretical key finding for this new model is the exhibition of algebraic constraints due to the equality among error variances in the same partition block. Consequently, a new type of model equivalence was developed, refining the classical Markov equivalence classes. To illustrate this, we displayed some examples for a small number of variables.

We then proposed three different methods for learning this new type of equivalence class from data. Each method utilizes a greedy score-based approach for estimating DAG and partition, varying on the combination of different searching schemes. In simulation studies, we tested the performance on the task of correctly estimating the CPDAG under partition information. Initializing DAG and partition randomly, followed by alternating multiple greedy DAG and partition steps, as described in Section 4.4.3, appears to be the most promising approach. It confirmed our expectation that exploiting partition information in the search algorithm can lead to better results in estimating CPDAG under partition information than standard methods like PC or GES. However, our approach of moving through DAG and partition space simultaneously (Section 4.4.2) underperformed on most configurations of sample size and number of variables inspected. Additionally, the runtime increased enormously with a larger number of variables.

One possible way to avoid inefficient search spaces is to move between equivalence classes. Assuming that the partition Π (or equivalently, background knowledge) of the variable set is known, one might extend the approach of Chickering (2002) to Π-model equivalence classes. There, a set of equivalence-class operators are introduced, which can be applied to a CPDAG and create a new CPDAG. This prevents moving within an equivalence class, and the set of local neighbors of a current state are scored locally, further increasing the computational efficiency.

To conclude, this thesis described two schemes of moving across partition space. Exploring more ways of moving across a partition lattice might further improve the methods presented. Another interesting future work would be to evaluate the techniques for estimating DAG and partition on possible future real-world datasets.

References

Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). “A characterization of Markov equivalence classes for acyclic digraphs”. In: *Ann. Statist.* Volume 25.(2), pp. 505–541.

Andres, B. (2020). *Lecture in Machine Learning I.* Machine Learning for Computer Vision TU Dresden `https://mlcv.inf.tu-dresden.de/courses/wt20/ml1/07-partitioning-handout.pdf`.

Becker, H. W. and Riordan, J. (1948). “The arithmetic of Bell and Stirling numbers”. In: *Amer. J. Math.* Volume 70, pp. 385–394.

Chickering, D. M. (2002). “Learning equivalence classes of Bayesian-network structures”. In: *J. Mach. Learn. Res.* Volume 2.(3), pp. 445–498.

Chickering, D. M. (2003). “Optimal structure identification with greedy search”. In: *J. Mach. Learn. Res.* Volume 3, pp. 507–554.

Drton, M. (2018). “Algebraic problems in structural equation modeling”. In: *The 50th anniversary of Gröbner bases.* Vol. 77. Adv. Stud. Pure Math. Math. Soc. Japan, pp. 35–86.

Drton, M., Sturmfels, B., and Sullivant, S. (2009). *Lectures on algebraic statistics.* Vol. Volume 39. Oberwolfach Seminars. Birkhäuser Verlag.

Duncan, O. D. (1975). *Introduction to structural equation models.* Academic Press [Harcourt Brace Jovanovich, Publishers].

Geiger, D. and Pearl, J. (1990). “On the logic of causal models”. In: *Uncertainty in artificial intelligence, 4.* Vol. 9. Mach. Intelligence Pattern Recogn. North-Holland, pp. 3–14.

Gillispie, S. B. and Perlman, M. D. (2001). “Enumerating Markov Equivalence Classes of Acyclic Digraph Models”. In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence.* Morgan Kaufmann Publishers Inc., pp. 171–177.

Haavelmo, T. (1944). "The probability approach in econometrics". In: *Econometrica* Volume 12, S, pp. iii–115.

Harris, N. and Drton, M. (2013). "PC algorithm for nonparanormal graphical models". In: *J. Mach. Learn. Res.* Volume 14, pp. 3365–3383.

Hoyer, P., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. (2008). "Nonlinear causal discovery with additive noise models". In: *Advances in Neural Information Processing Systems*. Vol. 21.

Kıcıman, E., Ness, R., Sharma, A., and Tan, C. (2023). *Causal Reasoning and Large Language Models: Opening a New Frontier for Causality.* Preprint at https://arxiv.org/abs/2305.00050.

Koopmans, T. (1945). "Statistical Estimation of Simultaneous Economic Relations". In: *Journal of the American Statistical Association* Volume 40.

Meek, C. (1995). "Causal Inference and Causal Explanation with Background Knowledge". In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 403–410.

OEIS Foundation Inc. (2023). Entry A000110 (Number of partitions) in The On-Line Encyclopedia of Integer Sequences, `https://oeis.org/A000110`.

OEIS Foundation Inc. (2023). Entry A003024 (Number of DAGs) in The On-Line Encyclopedia of Integer Sequences, `https://oeis.org/A003024`.

Pearl, J. (2009). *Causality: Models, Reasoning and Inference, 2nd Edition.* Cambridge University Press.

Pearl, J. and Mackenzie, D. (2018). *The book of why : The new science of cause and effect.* Basic Books.

Peters, J. and Bühlmann, P. (2014). "Identifiability of Gaussian structural equation models with equal error variances". In: *Biometrika* Volume 101.(1), pp. 219–228.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference.* Adaptive Computation and Machine Learning. MIT Press.

Peters, J., Mooij, J., Janzing, D., and Schölkopf, B. (2011). "Identifiability of Causal Graphs using Functional Models". In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence.* UAI'11. AUAI Press, pp. 589–598.

Radhakrishnan, A., Solus, L., and Uhler, C. (2018). "Counting Markov equivalence classes for DAG models on trees". In: *Discrete Appl. Math.*, pp. 170–185.

Richardson, T. and Spirtes, P. (2002). "Ancestral graph Markov models". In: *Ann. Statist.* Volume 30.(4), pp. 962–1030.

Robinson, R. W. (1973). "Counting labeled acyclic digraphs". In: *New directions in the theory of graphs (Proc. Third Ann Arbor Conf., Univ. Michigan, Ann Arbor, Mich., 1971).* Academic Press, pp. 239–273.

Shimizu, S., Hoyer, P. O., Hyvärinen, A., and Kerminen, A. (2006). "A linear non-Gaussian acyclic model for causal discovery". In: *Journal of Machine Learning Research (JMLR)* Volume 7, pp. 2003–2030.

Spirtes, P. (2013). "Directed Cyclic Graphical Representations of Feedback Models". In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence.*

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, prediction, and search, 2nd Edition.* Adaptive Computation and Machine Learning. MIT Press.

Tsamardinos, I., Brown, L., and Aliferis, C. (2006). "The max-min hill-climbing Bayesian network structure learning algorithm". In: *Machine learning* Volume 65.(1), pp. 31–78.

Verma, T. and Pearl, J. (1992). "An Algorithm for Deciding If a Set of Observed Independencies Has a Causal Explanation". In: *Proceedings of the Eighth International Conference on Uncertainty in Artificial Intelligence.* UAI'92. Morgan Kaufmann Publishers Inc., pp. 323–330.

Verma, T. and Pearl, J. (1990). "Equivalence and Synthesis of Causal Models". In: *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 255–270.

Wright, S. (1921). "Correlation and causation". In: *Journal of agricultural research* Volume 20.

Wu, J. and Drton, M. (2023). *Partial Homoscedasticity in Causal Discovery with Linear Models.* Preprint at https://arxiv.org/abs/2308.08959.