Technical University Munich
Department of Mathematics
Chair of Mathematical Statistics

# Identifiability and Estimation of Recursive Max-Linear Models

Master's Thesis
by
Gian Luca Esposito

Supervisor: Prof. Dr. Claudia Klüppelberg
Advisor: M. Sc. Johannes Buck
Submission Date: April 30, 2019

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

Garching, April 30, 2019

# Acknowledgments

First and foremost, I would like to thank my supervisor Professor Claudia Klüppelberg for giving me the opportunity to write my thesis at the Chair of Mathematical Statistics in this very interesting and fascinating mathematical field. I very much appreciate her feedback offered and all the inspiring ideas and suggestions that she made. All the advice and comments has been a great help to always stay on the right track.
I would like to thank Johannes Buck for the outstanding advisory during the last six months. Without his guidance, continuous presence and persistent help this thesis would not have been possible.

Finally, I also would like to thank my parents for their unconditional love and unceasing support over the last years during my studies. You always believed in me also in challenging situations.

# Abstract

A recursive max-linear model is a structural equation model in which the dependence structure between the random variables is represented by a directed acyclic graph. In comparison to usual Gaussian structural equation models sums are replaced by maxima and the Gaussian distribution is replaced by the standard Fréchet-distribution. Hence, well-known estimation methods that uses conditional independence to infer the structure of the underlying unknown DAG cannot be applied anymore. In this thesis we develop a new Branch & Bound algorithm to estimate the topological order of the nodes of a recursive max-linear model with underlying unknown directed acyclic graph. We extend the recursive max-linear model and introduce multiplicative noise in two different ways, first in a recursive manner and then as Hadamard product in order to test the new algorithm also in situations that come close to real world scenarios. A simulation study shows that the new algorithm performs very well, if we have non-noisy observations as well as if we have noisy observations.

# Contents

# Chapter 1

# Introduction

In probabilistic graphical models each node in a graph represents a random variable $X_i,\ i \in \{1,\dots,d\},\ d \in \mathbb{N}$, of a random vector $\boldsymbol{X} = (X_1,\dots,X_d)$ with joint distribution $\mathcal{L}(\boldsymbol{X})$ (we use capital letters for random variables and bold letters for random vectors). Probabilistic graphical models are useful to express dependency structures between the random variables of a probabilistic model and one can easily read off model properties from the graph. If we have $n$ observations $\boldsymbol{X}^1,\dots,\boldsymbol{X}^n$ of a random vector $\boldsymbol{X}$ following some graphical model, we are interested in the estimation of the underlying graph in order to get an overview of relations between the random variables (cf. Lauritzen [1996]). In this thesis we assume that all variables are observable, i.e. that there are no hidden variables.

In the special case of a probabilistic graphical model with underlying directed acyclic graph (*directed graphical model*) all edges in the graph represent conditional independence relations between the random variables $\boldsymbol{X} = (X_1,\dots,X_d)$ of the probabilistic model. If the random vector $\boldsymbol{X}$ is multivariate Gaussian, we can estimate the directed acyclic graph (DAG) by exploiting the conditional independence property of the multivariate Gaussian distribution, and by identifying compatible independence structures. Then we can use the popular and well-known PC-algorithm (cf. Glymour and Spirtes [1991]) which estimates a DAG given conditional independence.

Directed graphical models allow for causal interpretations, and thus they are particularly suitable for probabilistic models where it is important to understand cause-effect relations. This is for instance of great relevance in probabilistic models that deal with extreme risks. In Gissibl and Klüppelberg [2018], the authors develop directed graphical models that are suitable to model such scenarios. In these models Gaussian distributions are replaced by extreme value distributions and hence for example the PC-algorithm, that exploits conditional independence, is not applicable anymore.
In this thesis, based on work in Gissibl and Klüppelberg [2018] and Gissibl et al. [2018], we introduce new methods and algorithms that estimate the underlying structure of the DAG for those models. We first develop methods for the usual non-noisy model and then we introduce two different noise models and apply the proposed techniques also on them. A simulation study for all three models shows that the proposed algorithms work very well in the non-noisy model and still well in the noisy-models, even in high dimensions.

Real-world examples for directed graphical models that deal with extreme risk can be found in many different areas of application. One very intuitive example are floods in river networks (cf. Asadi et al. [2015]). If there is a flood in a river, for instance due to a local heavy rainfall, what is the probability that the flood propagates into branches of the river?
Other real-world examples can be found in the assessment of financial risks (cf. Einmahl et al. [2018]) or in technical risk analysis such as the "runway-overrun" event of airplane landing (cf. Gissibl et al. [2017]).

The thesis is organized as follows. In Chapter 2 we give a short introduction to general graph theory and present the underlying directed graphical models. In Chapter 3 we consider the usual non-noisy model. First we present estimation methods for scenarios in which we already know the ancestral relations of the DAG (often, natural ancestral relations exist, e.g. in river networks). Secondly, we assume that we know the topological order of the nodes and present estimation methods to find the ancestral relations between the nodes. In the third setting, we assume that we do not know anything about the underlying DAG. We introduce new algorithms in order to infer the topological order of the unknown DAG.

In Chapter 4 we introduce two different noise models and develop estimation methods for them based on the new algorithm developed in Chapter 3.
In Chapter 5 we perform a simulation study on all proposed methods and algorithms. We compare the goodness of the introduced algorithms for many different dimensions and furthermore we compare the computation time.

# Chapter 2

# Preliminaries

## 2.1 Directed Graphical Models: Max-linear (ML) and recursive ML models

We first give a brief introduction to basics of graph theory based on Lauritzen [1996]. A tuple $\mathcal{D} = (V, E)$ denotes a graph where $V = \{1, \dots, d\},\ d \in \mathbb{N}$, is the set of nodes and $E \subseteq V \times V$ is the set of edges. For $i, j \in V$ we denote by $(i, j) \in E$ the edge from node $i$ to node $j$ in $\mathcal{D}$, if not stated otherwise. For $k_l \in V,\ l \in \{0, \dots, t\},\ t \in \mathbb{N}$, we denote by

$$p_{ij} = [i = k_0 \to k_1 \to \ldots \to k_t = j]$$

a path from node $i$ to node $j$ of length $t$ in $\mathcal{D}$ and by $P_{ij}$ we denote the set of all paths from node $i$ to node $j$.
We call an edge $(i, j) \in E$ *directed*, if $(i, j) \in E$ but $(j, i) \notin E$. A path is a *directed path*, if all edges of the path are directed. A directed path is called *cyclic*, if there is at least one node in the path which is a descendant of itself. A *directed acyclic graph* (DAG) is a graph $\mathcal{D} = (V, E)$ where all edges are directed and where no cycles occur.

We denote by $\mathrm{pa}(i)$, $\mathrm{ch}(i)$, $\mathrm{an}(i)$ and $\mathrm{de}(i)$ the sets which contain the *parents*, *children*, *ancestors* and the *descendants* of node $i \in V$. Furthermore we set,

$$\begin{aligned} \mathrm{Pa}(i) &= \mathrm{pa}(i) \cup \{i\}, \quad & \mathrm{An}(i) &= \mathrm{an}(i) \cup \{i\}, \\ \mathrm{Ch}(i) &= \mathrm{ch}(i) \cup \{i\}, \quad & \mathrm{De}(i) &= \mathrm{de}(i) \cup \{i\}. \end{aligned}$$

One possibility to construct a directed graphical model is a recursive structural equation model. It constructs a random vector on a DAG.

**Definition 2.1.** Given a DAG $\mathcal{D} = (V, E)$ with nodes $V = \{1, \dots, d\}$ and edges $E = \{(k, i) : i \in V, k \in \mathrm{pa}(i)\}$, a *recursive structural equation model* is a multivariate statistical model where every random variable $X_i, i = \{1, \dots, d\}$, of a random vector $\boldsymbol{X} \in \mathbb{R}^{1\times d}$, is expressed by

$$X_i = f_i(\boldsymbol{X}_{\mathrm{pa}(i)}, Z_i), \quad i \in \{1, \dots, d\}.$$

The functions $f_i$ are real-valued and measurable, the set $\mathrm{pa}(i)$ contains the parents of node $i$, the sequence $(Z_i)_{i=1}^d$ are independent and identically distributed (i.i.d.) random variables and $\boldsymbol{X}_{\mathrm{pa}(i)}$ denotes the vector of all components of $\boldsymbol{X}$ being a parent of $X_i$ in $\mathcal{D}$.

One possible specification in recursive structural equation models for the functions $f_i, i \in \{1, \dots, d\}$, are sums as defined in Pearl [2009], Section 1.4.1,

$$X_i = \sum_{k \in \mathrm{pa}(i)} c_{ki} X_k + Z_i, \quad i \in \{1, \dots, d\}, \tag{2.1}$$

with edge-weights $c_{ki} \in \mathbb{R} \setminus \{0\}$. If the random variables $(Z_i)_{i=1}^d$ are i.i.d. and follow a normal distribution $\mathcal{N}(0, \sigma^2)$ with equal variances $\sigma^2 > 0$, then Bühlmann and Peters [2014] showed that the underlying DAG is identifiable from the joint distribution $\mathcal{L}(\boldsymbol{X})$ of $\boldsymbol{X}$.
However, in a context of risk assessment, Gaussian distributions underestimate extreme risks. Therefore, Gissibl and Klüppelberg [2018] proposed another specification of the functions $f_i, i \in \{1, \dots, d\}$:

**Definition 2.2** (Recursive max-linear model)**.** A recursive max-linear (ML) model $\boldsymbol{X} = (X_1, \dots, X_d)$ on a DAG $\mathcal{D} = (V, E)$ with positive edge-weights $c_{ki}$, $k \in \mathrm{pa}(i)$, is specified by

$$X_i = \bigvee_{k \in \mathrm{pa}(i)} c_{ki} X_k \vee Z_i, \quad i \in \{1, \dots, d\}, \tag{2.2}$$

where $\mathrm{pa}(i)$ are the parents of node $i$ and $(Z_i)_{i=1}^d$ are i.i.d. random variables with positive support $(0, \infty)$ and atom-free distributions. We refer to them as *innovations*.

To simplify notation we write $\bigvee_{k=1}^d a_k$ for the maximum $\max\{a_k;\ 1 \le k \le d\}$ and $a_1 \vee a_2$ for $\max\{a_1, a_2\}$, where $a_k \in \mathbb{R}$.
In a recursive ML model, as defined in (2.2), sums are replaced by maxima. In such a setting natural candidates for the distribution of the i.i.d. innovations $(Z_i)_{i=1}^d$ are extreme value distributions (EVDs) or distributions in their domains of attraction. Then, such a model is suitable to examine extreme risk propagating through a network and the replacement of sums by maxima becomes natural. In order to model the occurrence of extreme events the Fréchet distribution is suitable, since it is heavy tailed and regularly varying at infinity with shape parameter $\nu$. Therefore,

$$\mathbb{P}(S_n > x) \sim c\,\mathbb{P}(\max\{Y_1, \dots, Y_n\} > x),$$

with $S_n = \sum_{i=1}^n Y_i$, $c \in \mathbb{R}$ being a constant and $(Y_i)_{i=1}^n$ are i.i.d. random variables following a Fréchet distribution. Hence,

$$\mathbb{P}\big(\textstyle\sum_{k \in \mathrm{pa}(i)} c_{ki} X_k + Z_i > x\big) \sim c\,\mathbb{P}\big(\textstyle\bigvee_{k \in \mathrm{pa}(i)} c_{ki} X_k \vee Z_i\big). \tag{2.3}$$

The innovations $(Z_i)_{i=1}^d$ in a recursive ML model can be interpreted as random shocks at each node (cf. Klüppelberg and Lauritzen [2019]). Each edge $(k, i) \in E,\ i \in V,\ k \in \mathrm{pa}(i)$, is associated with a positive weight $c_{ki} > 0$. The edge-weights amplify or attenuate the risk moving from one node to another, depending on whether they are strictly larger than one or strictly smaller than one. We summarize all edge weights from a recursive ML model in the matrix $C = (c_{ij})_{d\times d}$ and call it the *edge-weight matrix* of the recursive ML model $\boldsymbol{X}$. We set the diagonal entries $c_{ii}$ of this matrix equal to one, which can be considered as scalings for the innovations $Z_i, i \in \{1, \dots, d\}$.

**Definition 2.3** (Max-linear model)**.** A max-linear (ML) model $\boldsymbol{X} = (X_1, \dots, X_d)$ on a DAG $\mathcal{D} = (V, E)$ is given by

$$X_i = \bigvee_{k\in\mathrm{An(i)}} b_{ki} Z_k = \bigvee_{k=1}^{d} b_{ki} Z_k, \quad i \in \{1, \dots, d\}, \tag{2.4}$$

with innovations $(Z_i)_{i=1}^d$ as defined in (2.2) and $B = (b_{ij})_{d\times d}$ is a matrix with non-negative entries.

We refer to the matrix $B = (b_{ij})_{d\times d}$ as the *max-linear coefficient matrix* (ML coefficient matrix) of the ML model $\boldsymbol{X}$ and call the entries $b_{ij}$ *max-linear coefficients* (ML coefficients).
In order to define the ML coefficients we assign a path-weight $d(p_{ij})$ to each path $p_{ij} = [i = k_0 \to k_1 \to \dots \to k_t = j],\ t \in \mathbb{N}$, by multiplying the edge-weights along this path $p_{ij}$:

$$d(p_{ij}) = c_{k_0 k_1} \dots c_{k_{t-1} k_t} = \prod_{l=0}^{t-1} c_{k_l k_{l+1}}. \tag{2.5}$$

Then, the ML coefficients are given by

$$b_{ij} = \begin{cases} \bigvee_{p_{ij}\in P_{ij}} d(p_{ij}) & \text{if } i \in \mathrm{an}(j), \\ 1 & \text{if } i = j, \\ 0 & \text{if } i \in V \setminus \mathrm{An}(j). \end{cases} \tag{2.6}$$

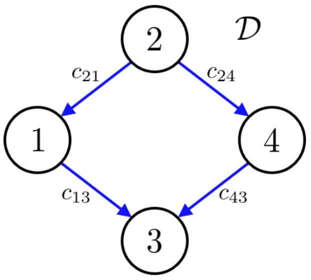
The following theorem states that every recursive ML model $\boldsymbol{X}$ has a ML representation (cf. Gissibl and Klüppelberg [2018], Theorem 2.2.).

**Theorem 2.4.** *Let $\boldsymbol{X}$ be a recursive ML model on a DAG $\mathcal{D} = (V, E)$ as defined in* (2.2) *and let $B = (b_{ij})_{d\times d}$ be the matrix with entries as defined in* (2.6)*. Then $\boldsymbol{X}$ can be rewritten as*

$$X_i = \bigvee_{k\in\mathrm{An(i)}} b_{ki} Z_k, \quad i \in \{1, \dots, d\}.$$

**Example 2.5** (Max-linear representation of a recursive ML model)**.** We consider a recursive ML model $\boldsymbol{X} = (X_1, X_2, X_3, X_4)$ on a DAG $\mathcal{D}$ with

$$\mathcal{D} = (V, E) = (\{1, 2, 3, 4\},\ \{(2, 4), (2, 1), (4, 3), (1, 3)\})$$



and edge-weight matrix given by

$$C = \begin{pmatrix} 1 & 0 & c_{13} & 0 \\ c_{21} & 1 & 0 & c_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & c_{43} & 1 \end{pmatrix} \in \mathbb{R}_+^{4\times 4}.$$

Starting with Definition 2.2 we obtain,

$$\begin{aligned} X_2 &= Z_2, \\ X_4 &= c_{24}X_2 \vee Z_4 = c_{24}Z_2 \vee Z_4, \\ X_1 &= c_{21}X_2 \vee Z_1 = c_{21}Z_2 \vee Z_1, \\ X_3 &= c_{43}X_4 \vee c_{13}X_1 \vee Z_3 = c_{43}(c_{24}Z_2 \vee Z_4) \vee c_{13}(c_{21}Z_2 \vee Z_1) \vee Z_3 \\ &= (c_{24}c_{43} \vee c_{21}c_{13})Z_2 \vee c_{13}Z_1 \vee c_{43}Z_4 \vee Z_3. \end{aligned}$$

The ML coefficient matrix $B$ is given by

$$B = \begin{pmatrix} 1 & 0 & c_{13} & 0 \\ c_{21} & 1 & c_{24}c_{43} \vee c_{21}c_{13} & c_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & c_{43} & 1 \end{pmatrix},$$

so indeed $\boldsymbol{X}$ satisfies representation (2.4).

For two matrices $F = (f_{ij})_{n\times m}$ and $G = (g_{jk})_{m\times p}$, $n, m, p \in \mathbb{N}$, with non-negative real entries, we define the matrix-operation $\odot$ by

$$\big(F \odot G\big)_{ik} = \bigvee_{t=1}^{m} f_{it}g_{tk},\ \ i = 1, \ldots, n,\ \ k = 1, \ldots, p. \tag{2.7}$$

In the same way as the usual matrix product we define for a matrix $A \in \mathbb{R}_+^{d\times d}$ powers recursively: $A^{\odot 0} = \mathrm{id}_{d\times d}$, $A^{\odot n} = A \odot A^{\odot(n-1)}, n \in \mathbb{N}$. Then we can rewrite equation (2.4) as $\boldsymbol{X} = B \odot \boldsymbol{Z}$, where $\boldsymbol{Z} = (Z_1, \ldots, Z_d)$.

**Remark 2.6.**

(i) A DAG $\mathcal{D}$ can be *well-ordered* in the sense that for any $k, i \in V$ and $k \in \text{pa}(i)$ it follows that $k < i$. The DAG in Example 2.5 is not well-ordered.

(ii) $\mathcal{D}$ is well-ordered if and only if the edge-weight matrix $C$ is in upper triangular form.

(iii) The edge-weight matrix $C$ is in upper triangular form if and only if the ML coefficient matrix $B$ is in upper triangular form.

(iv) Let $R = (r_{ij})_{d\times d}$ be the reachability matrix of $\mathcal{D}$ with entries defined by

$$r_{ij} = \begin{cases} 1 & \text{if there is a path } p_{ij} \text{ or if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Then it holds that,

$$R = \text{sgn}(B),$$

where sgn denotes the signum function and is taken componentwise. Hence, (iii) also holds, if we substitute $B$ by $R$ .

(v) The ML coefficient matrix $B$ is idempotent under the $\odot$-operation defined in (2.7) (meaning that $B \odot B = B$ or $b_{ij} = \bigvee_{k\in V} b_{ik}b_{kj}$ for all $i, j \in V$) if and only if $B$ corresponds to a DAG $\mathcal{D}$.

In order to compute the ML coefficient matrix $B$ we use (2.5) and (2.6). This algorithm is based on R-code provided by Nadine Gissibl.

**Algorithm 2.7** (Computation of the ML coefficient matrix $B$)**.**

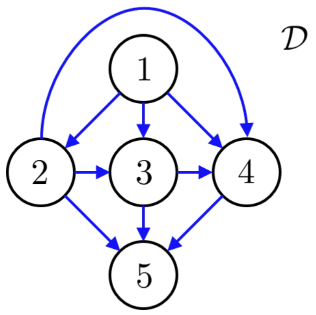
```
Input:  The edge-weight matrix $C = (c_{ij})_{d\times d}$.
Output: The ML coefficient matrix $B = (b_{ij})_{d\times d}$.

  1. Set $B = C$ and reorder the matrix $B$ such that the underlying DAG is
     well-ordered.

  2. Set all diagonal elements of $B$ equal to zero, i.e. for all $u \in V$ set
     $b_{uu} = 0$.

  3. For $k = 2, \dots, d-1$,
         for all $b_{ik}$ in the $k$-th column of $B$,
             for all $b_{kj}$ in the $k$-th row of $B$,

                 compute $z = b_{ik}b_{kj}$,
                 if $z > b_{ij}$, set $b_{ij} = z$.

             End for-loop.
         End for-loop.
     End for-loop.

  4. Set all diagonal elements of $B$ equal to one, i.e. for all $u \in V$ set
     $b_{uu} = 1$ and reorder $B$ according to its initial order.
```

## 2.2 The minimum ML directed acyclic graph (DAG)

The vector $\boldsymbol{X} = (X_1, \dots, X_d), d \in \mathbb{N}$, may satisfy representation (2.2) with respect to the underlying DAG $\mathcal{D}$ for different edge-weight matrices $C$. Furthermore, the whole vector $\boldsymbol{X}$ may satisfy (2.2) on subgraphs of $\mathcal{D}$. Therefore, in this section we introduce the DAG with the minimal number of edges such that every component of $\boldsymbol{X}$ satisfies representation (2.2) with unique edge-weights.

**Example 2.8** (Compute the minimum ML DAG)**.** We consider a recursive ML model $\boldsymbol{X} = (X_1, \dots, X_5)$ on a DAG $\mathcal{D} = (V, E)$ with edge-weight matrix given by

$$C = \begin{pmatrix} 1 & 3/4 & 1/2 & 2/3 & 0 \\ 0 & 1 & 3/4 & 1/2 & 1/3 \\ 0 & 0 & 1 & 4/5 & 4/5 \\ 0 & 0 & 0 & 1 & 2/5 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



If we compute the ML coefficient matrix $B$, for example with Algorithm 2.7, we obtain,

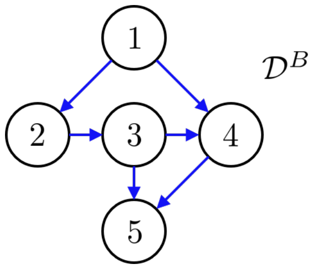
$$B = \begin{pmatrix} 1 & 3/4 & 9/16 & 2/3 & 9/20 \\ 0 & 1 & 3/4 & 3/5 & 3/5 \\ 0 & 0 & 1 & 4/5 & 4/5 \\ 0 & 0 & 0 & 1 & 2/5 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

In order to compute the minimum ML DAG denoted by $\mathcal{D}^B$ we set those edge-weights $c_{ij}$ in the matrix $C$ equal to zero for which it holds that

$$b_{ij} > c_{ij} > 0, \quad i, j \in V.$$

In our example we delete the edges $(2,4)$, $(2,5)$ and $(1,3)$. The resulting edge-weight matrix denoted by $C^B = (c_{ij}^B)_{d \times d}$ is given by

$$C^B = \begin{pmatrix} 1 & 3/4 & 0 & 2/3 & 0 \\ 0 & 1 & 3/4 & 0 & 0 \\ 0 & 0 & 1 & 4/5 & 4/5 \\ 0 & 0 & 0 & 1 & 2/5 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



The ML coefficient matrix of the minimum ML DAG $\mathcal{D}^B$ is again $B$ and in particular, $\mathcal{D}^B$ is the subgraph of $\mathcal{D}$ with the minimal number of edges such that the distribution $\mathcal{L}(\boldsymbol{X})$ of $\boldsymbol{X}$ is unchanged.

In Example 2.8 we deleted all edges $(i, j)$ with $b_{ij} > c_{ij} > 0$, since the risk will never pass through these edges, because there is a path $p_{ij}$ of larger weight. This leads to the following definition.

**Definition 2.9** (Max-weighted path)**.** Let $\boldsymbol{X}$ be a recursive ML model on a DAG $\mathcal{D} = (V, E)$ with path-weights defined in (2.5) and ML coefficient matrix $B$. A path $p_{ij}$ in $\mathcal{D}$ is called *max-weighted*, if

$$b_{ij} = d(p_{ij}), \quad i, j \in V,\ i \in \operatorname{an}(j).$$

**Definition 2.10** (Minimum max-linear DAG)**.** Let $\boldsymbol{X}$ be a recursive ML model on a DAG $\mathcal{D} = (V, E)$ with ML coefficient matrix $B = (b_{ij})_{d\times d}$ and edge-weight matrix $C = (c_{ij})_{d\times d}$. We call the DAG given by

$$\mathcal{D}^B = (V, E^B) = \Big(V, \Big\{(k, i) \in E :\ b_{ki} > \bigvee_{l \in \operatorname{de}(k) \cap \operatorname{pa}(i)} b_{kl} b_{li}\Big\}\Big)$$

the *minimum max-linear DAG* of $\boldsymbol{X}$. We denote by $C^B = (c^B_{ij})_{d\times d}$ the corresponding edge-weight matrix of the minimum ML DAG $\mathcal{D}^B$. We obtain this matrix $C^B$ by setting the entry $c_{ij}$ of the original edge-weight matrix $C$ equal to zero if and only if $b_{ij} > c_{ij} > 0$.

**Remark 2.11.** (i) The minimum ML DAG $\mathcal{D}^B = (V, E^B)$ is a subgraph of the initial DAG $\mathcal{D} = (V, E)$, i.e. it holds that $E^B \subseteq E$.

(ii) Since we delete all edges in $\mathcal{D}$ that are irrelevant for the distribution of $\boldsymbol{X}$, the DAGs $\mathcal{D}$ and $\mathcal{D}^B$ have the same ML coefficient matrix $B$, and hence also the same reachability matrix $\operatorname{sgn}(B)$.

(iii) The minimum ML DAG $\mathcal{D}^B$ has the same distribution $\mathcal{L}(\boldsymbol{X})$ of $\boldsymbol{X}$ as the DAG $\mathcal{D}$.

# Chapter 3

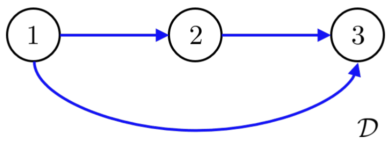
# Estimation of recursive ML models without noise

Before we start to describe estimation methods, we need to know which quantities in a recursive ML model $\boldsymbol{X}$ are generally identifiable from its distribution $\mathcal{L}(\boldsymbol{X})$. In Section 3.1 we start with considerations in this context mainly based on Gissibl et al. [2018], Section 3.

## 3.1 Identifiability of recursive ML models

Let $\boldsymbol{X} = (X_1, \dots, X_d)$ be a recursive ML model with underlying DAG $\mathcal{D}$. The question in this section is which quantities of $\boldsymbol{X}$ appearing in its definition can be identified from the distribution $\mathcal{L}(\boldsymbol{X})$ of $\boldsymbol{X}$.
From the following example we can see that we can generally neither identify the true underlying DAG $\mathcal{D}$ nor all edge-weights $c_{ki}$ of $\boldsymbol{X}$ in representation (2.2) from $\mathcal{L}(\boldsymbol{X})$.

**Example 3.1.** Let $\boldsymbol{X} = (X_1, X_2, X_3)$ be a recursive ML model with true underlying DAG $\mathcal{D}$ as in the figure below with edge-weights $c_{12}, c_{13}, c_{23}$.



By (2.2) the components of $\boldsymbol{X}$ have the following representation

$$X_1 = Z_1, \quad X_2 = c_{12}X_1 \vee Z_2, \quad X_3 = c_{13}X_1 \vee c_{23}X_2 \vee Z_3 \tag{3.1}$$

and by (2.4) they can be written as

$$X_1 = Z_1, \quad X_2 = c_{12}Z_1 \vee Z_2, \quad X_3 = (c_{12}c_{23} \vee c_{13})Z_1 \vee c_{23}Z_2 \vee Z_3.$$

Our main focus will be on the latter representation of $X_3$. If $c_{13} > c_{12}c_{23}$, then $\mathcal{D}$ is the only DAG with unique edge-weights that represents $\boldsymbol{X}$ in the sense of (2.2). That means that $\mathcal{D}$ and all edge-weights are identifiable from $\mathcal{L}(\boldsymbol{X})$ in this case.

However, if $c_{13} \leq c_{12}c_{23}$, then for all $c_{13}^* \in [0, c_{12}c_{23}]$ it holds that $b_{13} = c_{12}c_{23} \vee c_{13}^* = c_{12}c_{23}$. Hence, the edge $(1,3)$ does not have any influence on the distribution $\mathcal{L}(\boldsymbol{X})$ of $\boldsymbol{X}$ and we can write for $X_3$ in (3.1),

$$X_3 = c_{13}^* X_1 \vee c_{23}X_2 \vee Z_3 \quad \text{for all } c_{13}^* \in [0, c_{12}c_{23}].$$

That means that $\boldsymbol{X}$ follows a recursive ML model on the DAG $\mathcal{D}$ with edge-weights $c_{13}^*, c_{12}, c_{23}$ and also on the minimum ML DAG $\mathcal{D}^B$ with edge-weights $c_{12}, c_{23}$.

$$\begin{tikzcd} 1 \arrow[r] & 2 \arrow[r] & 3 \end{tikzcd}$$

$\mathcal{D}^B$

In this case the true underlying DAG $\mathcal{D}$, as well as the edge-weight $c_{13}$, is not identifiable from the distribution $\mathcal{L}(\boldsymbol{X})$.
Note however, that the ML coefficient $b_{13}$ is uniquely determined by $c_{13}$ in the first case and by $c_{12}c_{23}$ in the second case. Also the other ML coefficients are uniquely determined in both cases.

We have seen in Example 3.1 that it is generally not possible to identify all edge-weights of the underlying DAG $\mathcal{D}$ from $\mathcal{L}(\boldsymbol{X})$. Only edges $(i,j) \in E^B$ in the minimum ML DAG $\mathcal{D}^B$ are identifiable.
Therefore, the question is, whether it is possible to identify $B$ from the distribution $\mathcal{L}(\boldsymbol{X})$. Indeed, this is true in general as the following considerations show.

Using (2.2), we learn that for an edge between two nodes $i$ and $j$ it holds that

$$X_j = \bigvee_{k \in \mathrm{pa}(j)} c_{kj}X_k \vee Z_j \geq \bigvee_{k \in \mathrm{pa}(j)} c_{kj}X_k \geq c_{ij}X_i, \quad i,j \in V,\ i \in \mathrm{pa}(j). \tag{3.2}$$

Now assume there is a path $p_{ij} = [i = k_0 \to k_1 \to \ldots \to k_t = j]$ of length $t \in \mathbb{N},\ t \geq 2$, from $i$ to $j$. Then (3.2) can be applied iteratively in order to obtain,

$$X_{k_t} \geq c_{k_{t-1}k_t}X_{k_{t-1}} \geq c_{k_{t-1}k_t}c_{k_{t-2}k_{t-1}}X_{k_{t-2}} \geq \ldots \geq \prod_{l=0}^{t-1} c_{k_l k_{l+1}} X_{k_0},$$

which by (2.5) is equal to

$$X_j \geq d(p_{ij})X_i, \quad i,j \in V,\ i \in \mathrm{an}(j). \tag{3.3}$$

Since (3.3) holds for all paths from node $i$ to node $j$, it also holds for the max-weighted path, that is

$$X_j \geq \bigvee_{p_{ij} \in P_{ij}} d(p_{ij})X_j = b_{ij}X_i, \quad i,j \in V,\ i \in \mathrm{an}(j).$$

Reordering the last equation and taking into account that $X_i > 0$ for all $i \in V$, we finally obtain,

$$\frac{X_j}{X_i} \geq b_{ij}, \quad i,j \in V,\ i \in \mathrm{an}(j).$$

This means that for $i \in \mathrm{an}(j)$, the ratio $X_j/X_i$ is bounded from below by the ML coefficient $b_{ij} > 0$. In Gissibl et al. [2018], Lemma 3.2 (b), it is proved that the support of the ratio for $i \in$ an(j) is given by

$$\mathrm{supp}\left(\frac{X_j}{X_i}\right) = [b_{ij}, \infty), \quad i, j \in V, \ i \in \mathrm{an(j)}. \tag{3.4}$$

**Remark 3.2.** If we have $X_j/X_i = b_{ij}$, in a context of risk assessment in a network, this equality can be understood as a risk starting in node $i$ passing through a path $p_{ij}$ without being exceeded by the risk from another node in this path (which in turn either origins from the innovation at this node or by the risk from another path).
By Gissibl et al. [2018], Table 3.1, the set of all atoms of $X_j/X_i, i \in \mathrm{an}(j)$, is given by $\{b_{kj}/b_{ki}, k \in \mathrm{An}(i)\}$. In particular, it holds that

$$\mathbb{P}(X_j/X_i = b_{ij}) > 0, \quad \text{for } i, j \in V, i \in \mathrm{an}(j).$$

Taking into account that the edge-weights are strictly positive, we also learn from (2.2) in an analogously manner to (3.2) that for an edge between two nodes $j$ and $l$ it holds that

$$X_j \leq \frac{1}{c_{jl}} X_l, \quad j, l \in V, \ j \in \mathrm{pa}(l). \tag{3.5}$$

If we now consider a path $p_{jl} = [j = k_0 \to k_1 \to \ldots \to k_t = l]$ of length $t \in \mathbb{N}$, $t \geq 2$, from $j$ to $l$, we obtain analogously,

$$X_j \leq \frac{1}{\bigvee\limits_{p_{jl} \in P_{jl}} d(p_{jl})} X_l = \frac{1}{b_{jl}} X_l, \quad j, l \in V, \ l \in \mathrm{de}(j).$$

Reordering the last equation and taking into account that $X_l > 0$ for all $l \in V$, we finally obtain,

$$\frac{X_j}{X_l} \leq \frac{1}{b_{jl}}, \quad j, l \in V, \ l \in \mathrm{de}(j).$$

This means that for $l \in \mathrm{de}(j)$, the ratio $X_j/X_l$ is bounded from above by $\frac{1}{b_{jl}}$. Again by Gissibl et al. [2018], Lemma 3.2 (b), the support of the ratio is given by

$$\mathrm{supp}\left(\frac{X_j}{X_l}\right) = \left(0, \tfrac{1}{b_{jl}}\right], \quad j, l \in V, \ l \in \mathrm{de}(j), \tag{3.6}$$

and the set of all atoms for $X_j/X_l, l \in \mathrm{an}(j)$, is given by $\{b_{kj}/b_{kl}, k \in \mathrm{An}(j)\}$ (cf. Gissibl et al. [2018], Table 3.1).

As part of the definition of $\boldsymbol{X}$, in Gissibl et al. [2018] it is shown that we can also identify the distribution of the innovation vector $\boldsymbol{Z}$ from $\mathcal{L}(\boldsymbol{X})$.

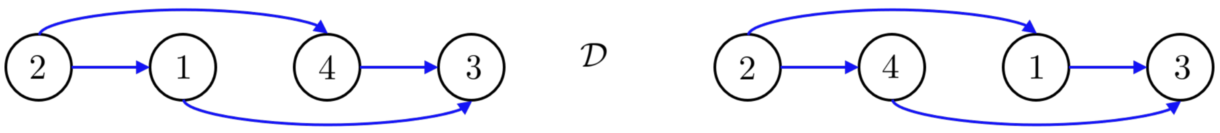
## 3.2 Topological order of the nodes of a DAG

We start this chapter with a definition adopted from Cormen et al. [2009], Section 22.4, p.612.

**Definition 3.3** (Topological order)**.** A *topological order* of a DAG $\mathcal{D} = (V, E)$ is a linear ordering of all its vertices such that if $\mathcal{D}$ contains an edge $(u, v) \in E$, then $u$ appears before $v$ in the ordering.

A topological order of a graph can be viewed as an ordering of its vertices along a horizontal line so that all directed edges go from left to right. A graph has a topological ordering if and only if it is a DAG, since only then a linear ordering is possible.

**Example 3.4.** Let us consider the recursive ML model $\boldsymbol{X} = (X_1, X_2, X_3, X_4)$ on the DAG $\mathcal{D} = \{\{1, 2, 3, 4\}, \{(2, 4), (2, 1), (4, 3), (1, 3)\}\}$ of Example 2.5.



The two possible topological orders of $\mathcal{D}$ are $(2, 1, 4, 3)$ and $(2, 4, 1, 3)$. We say that these two topological orders are *equivalent*, since both represent the structure of $\mathcal{D}$ in the sense of Definition 3.3.

Like in Example 3.4, a DAG $\mathcal{D} = (V, E)$, $|V| = d \in \mathbb{N}$, may have several topological orders of its nodes. Let $\Pi$ denote the set of all topological orders of size $d$. We define an equivalence relation $\sim$ on $\Pi$ by $x \sim y$ if and only if $x$ and $y$ represent the structure of $\mathcal{D}$ in the sense of Definition 3.3. As one can easily verify, the equivalence relation $\sim$ is reflexive, symmetric and transitive. The set of all topological orders induced by the reachability matrix $R = \operatorname{sgn}(B)$ of $\mathcal{D}$ forms an equivalence class $Q$ with respect to $\sim$.

The following pages are divided into three different initial settings. Let $\boldsymbol{X} = (X_1, \ldots, X_d)$ follow a recursive ML model on a DAG $\mathcal{D}$. In each setting we assume that we have given an i.i.d. sample $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^n,\ n \in \mathbb{N}$.

(1.) In the first setting we assume that we know the ancestral relations in the underlying DAG $\mathcal{D}$. The aim is to estimate the ML coefficient matrix $B$ (cf. Section 3.3).

(2.) In the second setting we assume that we know the topological order of the underlying DAG $\mathcal{D}$. The aim is to infer the minimum ML DAG $\mathcal{D}^B$, since we know from Section 3.1 that only $\mathcal{D}^B$ is identifiable and that it has the same distribution as the DAG $\mathcal{D}$. For this purpose, we first estimate the ML coefficient matrix $B$ and in a second step, we estimate the edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$ using the estimate of $B$ (cf. Section 3.4).

(3.) In the third setting we assume that we do not know anything about the underlying DAG $\mathcal{D}$. The aim is the estimation of a topological order of the nodes that belongs to the equivalence class $Q$ induced by $\mathcal{D}$ (cf. Section 3.5). Then, we estimate the minimum ML DAG $\mathcal{D}^B$ as in the second setting using the estimated topological order of the nodes.

When we start in the third setting where we assume to know nothing about the underlying DAG $\mathcal{D}$, we can go backward to setting two, i.e. we can estimate a topological order of the nodes and use this estimated topological order in setting two to estimate the ML coefficient matrix $B$ and the edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$.

In Section 3.5 we introduce four different algorithms to infer the underlying topological order of the nodes of an unknown DAG $\mathcal{D}$ with particular regard to computation time. The implementation of these methods and performing simulation studies to assess their small sample behavior is the first main part of this thesis.

## 3.3 Estimation of ML coefficients with known DAG

Let $\boldsymbol{X} = (X_1, \dots, X_d)$ follow a recursive ML model on a known DAG $\mathcal{D}$ and assume we have given an i.i.d. sample $\boldsymbol{X}^1, \dots, \boldsymbol{X}^n$. In this setting we want to estimate the ML coefficient matrix $B$. We have seen in Section 3.1 that it is possible to identify the ML coefficient matrix $B$ from the distribution of $\boldsymbol{X}$.

Let $B = (b_{ij})_{d\times d}, d \in \mathbb{N}$, be a matrix with non-negative entries and $b_{ii} = 1$ for all $i \in \{1, \dots, d\}$. We define $B_0 = \left(b_{ij}\mathbb{1}_{\mathrm{pa}(j)}(i)\right)_{d\times d}$ where $\mathbb{1}$ denotes the indicator function. Furthermore, denote by $\mathcal{B}(\mathcal{D})$ the class of possible ML coefficient matrices of all recursive ML models on a DAG $\mathcal{D}$. All matrices that belong to $\mathcal{B}(\mathcal{D})$ are idempotent with respect to the matrix multiplication $\odot$ defined in (2.7). By Theorem 4.2. in Gissibl and Klüppelberg [2018], it holds that $B \in \mathcal{B}(\mathcal{D})$, if and only if $[b_{ij} > 0 \Leftrightarrow i \in \mathrm{An}(j)]$ and additionally $B$ has to satisfy the fixed point equation $B = I_d \vee (B \odot B_0)$ where $I_d$ denotes the identity matrix.

From (3.4), we know that the ratio $X_j/X_i, i \in \mathrm{an}(j)$, is bounded from below by $b_{ij}$. Using this fact, it seems reasonable to use the following estimator to estimate the ML coefficients if we know the underlying DAG $\mathcal{D}$:

$$\breve{b}_{ij} = \bigwedge_{s=1}^{n} \frac{X_j^s}{X_i^s} \quad \text{for } i \in \mathrm{an}(j), \quad \breve{b}_{ii} = 1 \quad \text{and} \quad \breve{b}_{ij} = 0 \quad \text{for } i \in V \setminus \mathrm{An}(j). \tag{3.7}$$

For a sufficiently large sample size $n$, we can expect to observe the atoms $b_{ij}, i \in \mathrm{an}(j)$, and thus to obtain exact estimates for the ML coefficients (cf. Gissibl et al. [2018], p. 7). However, there occurs a problem with this estimator, if the sample size is small. It may happen that we obtain a matrix $\breve{B}$ which does not belong to $\mathcal{B}(\mathcal{D})$, cf. Example 4.1. in Gissibl et al. [2018]. Then, $\breve{B}$ is no suitable estimate for $B$ anymore. In the following we extend the estimator in (3.7) such that we always obtain an estimate for $B$ such that it belongs to $\mathcal{B}(\mathcal{D})$.
We first estimate all ML coefficients $b_{ij}$ corresponding to edges in $\mathcal{D}$, i.e. we compute

$$\breve{B}_0 = (\breve{b}_{ij}\mathbb{1}_{\mathrm{pa}(j)}(i))_{d\times d}, \tag{3.8}$$

where $\breve{b}_{ij}$ is given by (3.7). Then we compute an estimate for the ML coefficient matrix $B$ based on Lemma 3.5 below, which corresponds to Lemma 4.2 in Gissibl et al. [2018].

**Lemma 3.5.** *Let $B_0 \in \mathbb{R}_+^{d\times d}$ be a matrix with $b_{ij} > 0$ if and only if $i \in \mathrm{pa}(j)$. A matrix $A \in \mathbb{R}_+^{d\times d}$ satisfies*

$$[a_{ij} > 0 \Leftrightarrow i \in \mathrm{An}(j)] \text{ and } A = I_d \vee (A \odot B_0),$$

*if and only if $A = (I_d \vee B_0)^{\odot(d-1)}$.*

Let us consider a path $p_{ij} = [i = k_0 \to k_1 \to \ldots \to k_u = j]$ of length $u \geq 2$ and a realization $\boldsymbol{X}^t = (X_1^t, \ldots, X_d^t)$ such that $\breve{b}_{ij} X_i^t = X_j^t, i \in \mathrm{an}(j)$. Then, we multiply the entries of $\breve{B}_0$ defined in (3.8) along this path and obtain,

$$\bigwedge_{s=1}^{n} \frac{X_{k_1}^s}{X_{k_0}^s} \bigwedge_{s=1}^{n} \frac{X_{k_2}^s}{X_{k_1}^s} \cdots \bigwedge_{s=1}^{n} \frac{X_{k_u}^s}{X_{k_{u-1}}^s} \leq \frac{X_{k_1}^t}{X_{k_0}^t} \frac{X_{k_2}^t}{X_{k_1}^t} \cdots \frac{X_{k_u}^t}{X_{k_{u-1}}^t} = \frac{X_{k_u}^t}{X_{k_0}^t} = \breve{b}_{ij} = \bigwedge_{s=1}^{n} \frac{X_j^s}{X_i^s}. \tag{3.9}$$

Thus, we define the new estimator $\widehat{B}$ by first computing the matrix $\breve{B}_0 = (\breve{b}_{ij} \mathbb{1}_{\mathrm{pa}(j)}(i))_{d\times d}$ and secondly computing the matrix product $\odot$:

$$\widehat{B} = (I_d \vee \breve{B}_0)^{\odot(d-1)}.$$

Then, by Lemma 3.5 $\widehat{B}$ always belongs to $\mathcal{B}(\mathcal{D})$. The entries of $\widehat{B}$ are explicitly defined by

$$\begin{aligned} &\widehat{b}_{ii} = 1, \quad \widehat{b}_{ij} = 0 \text{ for } i \in V \setminus \mathrm{An}(j), \quad \widehat{b}_{ij} = \bigwedge_{s=1}^{n} \frac{X_j^s}{X_i^s}, \text{ for } i \in \mathrm{pa}(j) \quad \text{ and} \\ &\widehat{b}_{ij} = \bigvee_{p_{ij} \in P_{ij}} \widehat{d}(p_{ij}), \text{ for } i \in \mathrm{an}(j) \setminus \mathrm{pa}(j), \end{aligned} \tag{3.10}$$

where $\widehat{d}(p_{ij}) = \prod_{l=0}^{u-1} \widehat{b}_{k_l k_{l+1}}$ for a path $p_{ij} = [i = k_0 \to k_1 \to \ldots \to k_u = j]$. By (3.4) it holds that $b_{ij} \leq \widehat{b}_{ij}$ for all $i, j \in V$ and by (3.9) it holds that $\widehat{b}_{ij} \leq \breve{b}_{ij}$ for all $i \in \mathrm{an}(j) \setminus \mathrm{pa}(j)$, so altogether,

$$b_{ij} \leq \widehat{b}_{ij} \leq \breve{b}_{ij}, \text{ for all } i, j \in V. \tag{3.11}$$

The inequality (3.11) shows that the estimator $\widehat{B}$ is more exact than the estimator $\breve{B}$ and thus $\widehat{B}$ is always preferable.

**Lemma 3.6.** *Let $\boldsymbol{X} = (X_1, \ldots, X_d)$ follow a recursive ML model on a DAG $\mathcal{D}$ and let $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^n$ be an i.i.d. sample. Furthermore, let $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ be defined as in (3.10). Then, for $i \in V$ and $i \in \mathrm{an}(j)$ it holds that $\widehat{b}_{ij} \overset{\text{a.s.}}{\to} b_{ij}$, as $n \to \infty$.*

*Proof.* By (3.11) it holds that $b_{ij} \leq \widehat{b}_{ij} \leq \breve{b}_{ij}$ for all $i, j \in V$ and in particular for $i \in \mathrm{an}(j)$. If we show that $\breve{b}_{ij} \overset{\text{a.s.}}{\to} b_{ij}$ for $n \to \infty$, we can conclude that $\widehat{b}_{ij} \overset{\text{a.s.}}{\to} b_{ij}$ for $n \to \infty$. Since $\breve{b}_{ij}$ is a decreasing sequence as $n$ increases, it suffices to show that $\breve{b}_{ij} \overset{\mathbb{P}}{\to} b_{ij}$ in order to

conclude that $\breve{b}_{ij} \overset{\text{a.s.}}{\to} b_{ij}$:
Let $t > 0$. Then it holds by the independence of the realizations and for $i \in \operatorname{an}(j)$ that

$$\mathbb{P}\Big(\breve{b}_{ij} > t\Big) = \mathbb{P}\Big(\bigwedge_{s=1}^{n} \frac{X_j^s}{X_i^s} > t\Big) = \mathbb{P}\Big(\frac{X_j^1}{X_i^1} > t, \dots, \frac{X_j^n}{X_i^n} > t\Big) = [1 - F_{X_j^1/X_i^1}(t)]^n,$$

where $F$ denotes the cumulative distribution function (c.d.f.). Now, $F_{X_j^1/X_i^1}(t) > 0$ if and only if $t > b_{ij}$. Hence, for all $t > b_{ij}$, it holds that $\mathbb{P}(\breve{b}_{ij} > t) = [1 - F_{X_j^1/X_i^1}(t)]^n \to 0$ as $n \to \infty$.
Now let $\varepsilon > 0$. We use the definition of convergence in probability and skip the absolute value, since $\breve{b}_{ij} \geq b_{ij}$ by (3.11). Then,

$$\mathbb{P}(\breve{b}_{ij} - b_{ij} > \varepsilon) = \mathbb{P}(\breve{b}_{ij} > \varepsilon + b_{ij}) = [1 - F_{X_j^1/X_i^1}(\varepsilon + b_{ij})]^n \to 0, \text{ as } n \to \infty,$$

which means that $\breve{b}_{ij} \xrightarrow{\mathbb{P}} b_{ij}$ and we obtain almost sure convergence of $\widehat{b}_{ij}$ with the above considerations. □

In Gissibl et al. [2018], Proposition 4.5, it is shown that $\widehat{b}_{ij}$ converges even exponentially fast to $b_{ij}$ for $n \to \infty$ and in Theorem 4.11. of the same authors it is proved that $\widehat{B}$ is an extension of the maximum-likelihood estimator (MLE). In fact, the matrix $\widehat{B}$ is a so-called *generalized maximum likelihood estimator* (GMLE) for the ML coefficient matrix $B$. The usual MLE is not applicable in our case, because it is not well-defined. In fact, the family of probability measures $\mathcal{P}(\mathcal{D})$ induced by $\boldsymbol{X}$ on $\mathcal{D}$ is not dominated.

## 3.4 Estimation of ML coefficients with known topol. order

Let $\boldsymbol{X} = (X_1, \dots, X_d)$ follow a recursive ML model on a DAG $\mathcal{D}$ from which we know the topological order $\pi \in Q$ and assume that we have given an i.i.d. sample $\boldsymbol{X}^1, \dots, \boldsymbol{X}^n$.

Assume we have given some specific topological order in a DAG with four nodes, for instance $\pi = (4, 2, 3, 1)$. By Definition 3.3 it is clear that there is no edge $(3, 2)$ in the minimum ML DAG $\mathcal{D}^B$, for instance. However, it is not clear whether there is an edge $(2, 3)$ in $\mathcal{D}^B$ or not.
The aim is to estimate the minimum ML DAG $\mathcal{D}^B$, since we know from Section 3.1 that only $\mathcal{D}^B$ is identifiable and that it has the same distribution as the DAG $\mathcal{D}$. The DAG $\mathcal{D}$ is not identifiable.
We denote by $\pi(i)$ the position of node $i$ in $\pi$. For example, in the topological order above we obtain $\pi(4) = 1,\ \pi(2) = 2,\ \pi(3) = 3$ and $\pi(1) = 4$.

We define the minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d \times d}$ by

$$\widehat{b}_{ij} = \bigwedge_{s=1}^{n} \frac{X_j^s}{X_i^s} \quad \text{for } i, j \in V, \tag{3.12}$$

which is the initial point for everything that follows in this section.

**Corollary 3.7.** *The minimum ratio estimator as defined in* (3.12) *converges almost surely to the true ML coefficient $b_{ij}$ for $i \in \operatorname{an}(j)$, and almost surely to zero for $i \in V \setminus \operatorname{An}(j)$, as $n \to \infty$.*

*Proof.* We have proved the almost sure convergence of $\widehat{b}_{ij}$ for the case $i \in \operatorname{an}(j)$ already in the proof of Lemma 3.6 and we computed the c.d.f. $F_{X_j^k/X_i^k}(t)$ already in there . Now let $i \in V \setminus \operatorname{An}(j)$. Then, it holds that $F_{X_j^k/X_i^k}(t) > 0$ for all $t > 0$. Thus, with the same arguments as in the proof of Lemma 3.6 it holds for all $\varepsilon > 0$ that

$$\mathbb{P}(\breve{b}_{ij} > \varepsilon) = [1 - F_{X_j^k/X_i^k}(\varepsilon)]^n \to 0, \text{ as } n \to \infty,$$

which proves the statement. □

In the following algorithm we estimate the ML coefficient matrix $B$ with given topological order of the nodes $\pi$ and afterwards in a second algorithm, we estimate the edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$.

**Algorithm 3.8** (Estimate $B$ with known topological order of the nodes)**.**

```
Input:  The minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as defined in (3.12) and the
        known topological order of the nodes $\pi$.
Output: An estimate for $B$.

  1. Set $S = \varnothing$.

  2. For all $\widehat{b}_{ij}$ with $\pi(i) > \pi(j)$, set $S = S \cup \{(i,j)\}$.

  3. For all $\widehat{b}_{ij}$ with $\pi(j) - \pi(i) = 1$ and $\widehat{b}_{ij} < c(n)$, set $S = S \cup \{(i,j)\}$.

  4. For $z = 2, \dots, d-1$,
         for all $\widehat{b}_{ij}$ with $\pi(j) - \pi(i) = z$ and $\widehat{b}_{ij} < c(n)$,
             for any $k \in V$ such that $\pi(i) < \pi(k) < \pi(j)$,
                 if $(i,k) \in S$ or $(k,j) \in S$ or $((i,k) \in S$ and $(k,j) \in S)$, set
     $S = S \cup \{(i,j)\}$.

  5. For all $(i,j) \in S$, set $\widehat{b}_{ij} = 0$.
```

The set $S$ in Algorithm 3.8 is a set of pairs $(i,j)$ such that for all $(i,j) \in S$ it hold that $\widehat{b}_{ij} < c(n)$ and the resulting matrix $\widehat{B}$ is still idempotent. Only the choice $c(n) \in \big(0, \bigvee_{i,j\in V} \widehat{b}_{ij}\big)$ makes sense, since $0 < \widehat{b}_{kl} < \bigvee_{i,j\in V} \widehat{b}_{ij}$ for all $k,l \in V$. Moreover, we want $c(n)$ to satisfy the following two conditions:

(i) $c(n) \to 0$ for $n \to \infty$ and

(ii) for all $i,j \in V$ such that $b_{ij} = 0$: $\frac{\widehat{b}_{ij}}{c(n)} \xrightarrow{\mathbb{P}} 0$ (in $o$-notation that means $\widehat{b}_{ij} = o_{\mathbb{P}}(c(n))$).

From Hartl [2015], Corollary 5.20, we know that the c.d.f. of the estimated ML coefficient $\widehat{b}_{ij}$ in case of Fréchet($\nu$)-distributed innovations $(Z_i)_{i=1}^d$ is given by

$$\mathbb{P}\big(\widehat{b}_{ij} < x\big) = 1 - \left(\frac{a}{x^\nu b + a}\right)^n, \quad x > 0.$$

If $b_{ij} = 0$, $a$ and $b$ are constants for $x \downarrow 0$. In the simulation study in Chapter 5 we consider Fréchet(1)-distributed innovations $(Z_i)_{i=1}^d$. Thus, for $\nu = 1$ and $x = \frac{1}{n}$, we obtain for $n \to \infty$,

$$\mathbb{P}\big(\widehat{b}_{ij} < \tfrac{1}{n}\big) = 1 - \left(\frac{a}{\frac{b}{n} + a}\right)^n \to 1 - \exp\{-\tfrac{b}{a}\},$$

however we demand convergence to 1.
Hence, for $x = n^{-1+\delta}, \delta > 0$, we obtain for $n \to \infty$,

$$\mathbb{P}\big(\widehat{b}_{ij} < n^{-1+\delta}\big) = 1 - \left(\frac{a}{bn^{-1+\delta} + a}\right)^n \to 1,$$

and condition (ii) above is satisfied for all $\delta > 0$. However, condition (i) *and* (ii) are satisfied, if and only if $0 < \delta < 1$. Therefore, it seems reasonable to choose

$$c(n) = \tfrac{k}{n}, \tag{3.13}$$

for some constant $k > 1$.

In step 2 of Algorithm 3.8 we set all entries in $\widehat{B}$ equal to zero that do not correspond to the known topological order. Furthermore, we need to go from small to big distance in order to preserve the idempotence during step 4.

After we have estimated $B$, in a second step we search for critical paths in the estimate for $B$ to obtain an estimate for the edge-weight matrix $C^B$ corresponding to the minimum ML DAG $\mathcal{D}^B$.

**Algorithm 3.9** (Estimate the minimum ML DAG).

```
Input:  The matrix $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as output from Algorithm 3.8.
Output: An estimate for the edge-weight matrix $C^B = (c_{ij}^B)_{d\times d}$ of the minimum ML
        DAG $\mathcal{D}^B$.

  1. Set $\widehat{C}^{\widehat{B}} = \widehat{B}$.

  2. For all $i,j,k \in V$ with $\pi(i)<\pi(k)<\pi(j)$,
         if $|\widehat{b}_{ij} - \bigvee_k \widehat{b}_{ik}\widehat{b}_{kj}| = 0$, set $\widehat{c}_{ij}^{\widehat{B}} = 0$.
```

Note, that we could omit the absolute value in step 2., since for all $i,j,k \in V$ with $\pi(i) < \pi(k) < \pi(j)$ it always holds that $\widehat{b}_{ij} \geq \widehat{b}_{ik}\widehat{b}_{kj}$.

## 3.5 Estimation of the topol. order

Let $\boldsymbol{X} = (X_1, \dots, X_d)$ follow a recursive ML model on a DAG $\mathcal{D}$ which is assumed to be unknown and assume we have given an i.i.d. sample $\boldsymbol{X}^1, \dots, \boldsymbol{X}^n$. The aim in this setting is to infer the topological order the nodes.
We start our considerations with the minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as defined in (3.12), which is the initial point for everything that follows in this section.
The first idea is that we search for critical paths $\widehat{b}_{ij} = \widehat{b}_{ik}\widehat{b}_{kj}$ to find ancestral relations and conclude that $\pi(i) < \pi(k) < \pi(j)$. However, this method is not very reliable as the following example shows.

**Example 3.10.** Assume that the true underlying DAG $\mathcal{D}$ is given by the graph depicted below.

$$\mathcal{D}$$

$$\begin{tikzcd} X_j \arrow[r] & X_i \arrow[r] & X_k \end{tikzcd}$$

In the following we prove that the observation $\widehat{b}_{ij} = \widehat{b}_{ik}\widehat{b}_{kj}$ has positive probability. However, we cannot conclude that $\pi(i) < \pi(k) < \pi(j)$.
It holds that

$$\widehat{b}_{kj} = \bigwedge_{s=1}^{n} \frac{X_j^s}{X_k^s} = \bigwedge_{s=1}^{n} \frac{1}{\frac{X_k^s}{X_j^s}} = \frac{1}{\bigvee_{s=1}^{n} \frac{X_k^s}{X_j^s}},$$

and therefore we can rewrite the equation $\widehat{b}_{ij} = \widehat{b}_{ik}\widehat{b}_{kj}$ as

$$\bigvee_{s=1}^{n} \frac{X_k^s}{X_j^s} = \bigwedge_{s=1}^{n} \frac{X_k^s}{X_i^s} \bigvee_{s=1}^{n} \frac{X_i^s}{X_j^s}. \tag{3.14}$$

Now assume, that the maximum on the left-hand side of (3.14) is attained for some index $t \in \{1, \dots, n\}$, i.e.

$$\bigvee_{s=1}^{n} \frac{X_k^s}{X_j^s} = \frac{X_k^t}{X_j^t}. \tag{3.15}$$

For this index $t$ it directly follows from the definition of recursive ML models that $X_k^t = X_i^t b_{ik} \vee Z_k^t \geq X_i^t b_{ik}$. It holds that

$$\mathbb{P}(X_k^t = X_i^t b_{ik}) > 0, \tag{3.16}$$

because

$$\begin{aligned} &\mathbb{P}\big(b_{jk}Z_j \vee b_{ik}Z_i \vee Z_k = (Z_j b_{ji} \vee Z_i)b_{ik}\big) \\ =&\mathbb{P}\big(b_{jk}Z_j \vee b_{ik}Z_i \vee Z_k = b_{jk}Z_j \vee b_{ik}Z_i\big) \\ >&\, 0. \end{aligned}$$

Therefore, we may assume that

$$X_k^t = X_i^t b_{ik}. \tag{3.17}$$

For $n \to \infty$ it holds that $\widehat{b}_{ik} = \bigwedge_{s=1}^n X_k^s / X_i^s \to b_{ik}$ exponentially fast (cf. Gissibl et al. [2018], Proposition 4.5). Thus, we may assume that $\bigwedge_{s=1}^n X_k^s/X_i^s = b_{ik}$. The maximum on the right-hand side of (3.14) must be attained for the same index $t$, i.e. $\bigvee_{s=1}^n X_i^s/X_j^s = X_i^t/X_j^t$. Otherwise, there would exist a $t^* \in \{1, \dots, n\}, t^* \neq t$, such that

$$\frac{X_i^{t^*}}{X_j^{t^*}} > \frac{X_i^t}{X_j^t}.$$

However, this is a contradiction to (3.15), since

$$\frac{X_k^{t^*}}{X_j^{t^*}} \geq \frac{X_i^{t^*} b_{ik}}{X_j^{t^*}} > \frac{X_i^t b_{ik}}{X_j^t} = \frac{X_k^t}{X_j^t},$$

by assumption (3.17). For this reason we have

$$\bigvee_{s=1}^t \frac{X_i^s}{X_j^s} = \frac{X_i^t}{X_j^t} \quad \text{and} \quad \bigvee_{s=1}^t \frac{X_k^s}{X_j^s} = \frac{X_k^t}{X_j^t}.$$

Therefore,

$$\bigvee_{s=1}^n \frac{X_k^s}{X_j^s} = \frac{X_k^t}{X_j^t} = b_{ik}\frac{X_i^t}{X_j^t} = b_{ik}\bigvee_{s=1}^n \frac{X_i^s}{X_j^s},$$

which is equivalent to $X_k^t = b_{ik} X_i^t$ and has positive probability given $\mathcal{D}$.

Another reason, why we should not look for critical paths to infer the topological order of the nodes is that, if we analyze real data sets, we will only rarely observe critical paths, since we cannot expect to observe atoms.
Due to these facts, we need other ideas to infer the topological order of unknown DAGs.

### 3.5.1 Greedy algorithm to infer the topol. order

Let $Q$ denote the equivalence class of all topological orders of the nodes induced by the true underlying DAG $\mathcal{D}$ (cf. Section 3.2). The aim is to infer a topological order $\widehat{\pi} \in Q$. The only information we have at hand is the minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d \times d}$ as defined in (3.12).

The first idea we have is based on the following consideration: it holds that $\widehat{\pi} \notin Q$ if and only if there is a pair of distinct nodes $i, j \in V$ such that $b_{ij} > 0$ and $\widehat{\pi}(j) < \widehat{\pi}(i)$. This leads to the following optimization problem in order to infer a topological order of the nodes of an unknown DAG.

**Optimization Problem 3.11.** Find the topological order $\pi^*$ of $(1, \dots, d)$ such that,

$$\pi^* = \arg\min_{\pi \in \Pi} \max_{\substack{(i,j) \in V \times V: \\ \pi(j) < \pi(i)}} \widehat{b}_{ij}, \tag{3.18}$$

where $\Pi$ denotes the set of all topological orders of length $d$.

In other words, we want to minimize the maximal estimated ML coefficient $\widehat{b}_{ij}, i, j \in V$, whose pair of indices $(i, j)$ does not belong to a topological order in $Q$. Hence, we want to find $\pi \in Q$ such that the maximal $\widehat{b}_{ij}$ with $\pi(j) < \pi(i)$ is minimized.

We know from Corollary 3.7 that $\widehat{b}_{ij} \overset{\text{a.s.}}{\to} 0$ for $i \in V \setminus \operatorname{An}(j)$ and $\widehat{b}_{ij} \overset{\text{a.s.}}{\to} b_{ij}$ for $i \in \operatorname{An}(j)$. Therefore, it also holds that

$$\lim_{n \to \infty} \max_{\substack{(i,j) \in V \times V: \\ \pi(j) < \pi(i)}} \widehat{b}_{ij} = 0 \quad \text{for all } \pi \in Q,$$

and

$$\lim_{n \to \infty} \max_{\substack{(i,j) \in V \times V: \\ \pi(j) < \pi(i)}} \widehat{b}_{ij} > 0 \quad \text{for all } \pi \in \Pi \setminus Q.$$

Hence, it follows that

$$\lim_{n \to \infty} \mathbb{P}\Big( \arg\min_{\pi \in \Pi} \max_{\substack{(i,j) \in V \times V: \\ \pi(j) < \pi(i)}} \widehat{b}_{ij} \in Q \Big) = 1.$$

In order to use these facts systematically we apply a so-called *Greedy algorithm* to solve Optimization Problem 3.11. The Greedy algorithm always makes the choice that is optimal at the current step of the algorithm without considering the steps that follow (cf. Cormen et al. [2009], p. 414). It therefore "hopes" that this choice will lead to a globally optimal solution. The advantage of this algorithm is that it is fast. However, the Greedy algorithm does not always yield an optimal solution, but for many problems it does.

In our situation, the Greedy algorithm does not solve Optimization Problem 3.11, if we generate a cycle. We ensure not to obtain a cycle by introducing a reachability matrix $R = (r_{ij})_{d \times d}$ starting as zero matrix. In each iteration we check whether there is already a path from node $j$ to node $i$ in $R$ and if this is not the case we set $r_{ij} = 1$, otherwise we continue.

**Algorithm 3.12** (Greedy Algorithm)**.**

```
Input:  The minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d \times d}$ as defined in (3.12).
Output: A topological order $\widehat{\pi}$.

  1. Sort all $\widehat{b}_{ij}$ by size from large to small. Remove all diagonal entries
     $\widehat{b}_{ii} = 1$.
```

```
2. Set $R = (r_{ij})_{d\times d} = (\mathbb{0})_{d\times d}$.

3. For all $\widehat{b}_{ij}$ beginning with the largest,
       if $r_{ji} \neq 1$,
           set $r_{ij} = 1$ and update $R$ to a reachability matrix with
           Algorithm A.3;
       else, continue with $\widehat{b}_{ij}$ next in size.
   End for-loop.

4. Use a Depth-First Search (DFS) for graphs to recover a topological order
   from the reachability matrix $R$. An algorithm for a DFS for graphs is
   given in Cormen et al. [2009], Section 22.3., p. 604.
```

The worst-case running time to solve Optimization Problem 3.11 with the Greedy algorithm is $\mathcal{O}(d^4)$:

(i) The cost to sort the entries $\widehat{b}_{ij}$ by size using a standard sorting algorithm, for instance a merge sort, is $\mathcal{O}(d\log(d))$ (cf. Knuth [1973], Section 5.2.4). Since we have $d^2$ entries as input we obtain $\mathcal{O}(d^2\log(d))$.

(ii) The main for-loop in step 3 takes $d(d-1)$ steps into account. In each iteration we check in constant time whether there is a already path from node $j$ to $i$ in $R$. If we make a decision, i.e. if we set $r_{ij} = 1$, we update the matrix $R$ to a reachability matrix. This has a cost of $\mathcal{O}(d^2)$ due to the nested for-loops in step 4 of Algorithm A.3. Altogether we obtain,

$$\begin{aligned}\mathcal{O}\Big(\sum_{k=1}^{d(d-1)}\Big[\Big(\sum_{k=1}^{(d/2)-1}\sum_{k=1}^{(d/2)-1} 1\Big]\Big) &= \mathcal{O}\Big(\sum_{k=1}^{d^2} d^2\Big)\\ &= \mathcal{O}\Big(d^4\Big).\end{aligned}$$

(iii) The DFS for graphs in step 4 of the Greedy algorithm has cost of $\mathcal{O}(|V| + |E|) = \mathcal{O}(d^2)$ (cf. Cormen et al. [2009], Section 22.3., p.606).

Stating the worst-case running time of $\mathcal{O}(d^4)$ for the Greedy algorithm is a rather pessimistic bound. It is more reasonable to state the *amortized running time*. That is the average performance of each operation in the worst case (Cormen et al. [2009], chapter 17, p. 451).
Theorem 4.3 in Italiano [1986] shows, that deciding whether there is a path from node $j$ to node $i$ and setting an edge in a directed graph, i.e. setting $r_{ij} = 1$, can both be solved in $\mathcal{O}(d)$ amortized time.
Hence, the Greedy algorithm (Algorithm 3.12) has an amortized running of $\mathcal{O}(d^3)$.

**Lemma 3.13.** *The Greedy algorithm (Algorithm 3.12) solves Optimization Problem 3.11.*

*Proof.* The output $\widehat{\pi} \in Q$ from the Greedy algorithm solves (3.18), if step 3 in the Greedy algorithm does not result in any cycles and at the same time if there is no $\pi^* \in Q$ such that

$$\max_{\substack{(i,j)\in V\times V:\\ \pi^*(j)<\pi^*(i)}} \widehat{b}_{ij} < \max_{\substack{(i,j)\in V\times V:\\ \widehat{\pi}(j)<\widehat{\pi}(i)}} \widehat{b}_{ij}. \tag{3.19}$$

First assume that we can generate a cycle in step 3. Since we start with an empty reachability matrix, we create a cycle in some iteration of step 3 by adding a path from node $j$ to node $i$ in the graph (which is equivalent to setting $r_{ji} = 1$). Therefore, the cycle must contain the path from $j$ to $i$ and hence there is also a path from $i$ to $j$ which is a contradiction.
Now assume that there is a $\pi^*$ such that (3.19) holds. Let $(j', i')$ denote the first pair of nodes in Algorithm 3.12 where we could not set $r_{j'i'} = 1$, since there is already a path from $i'$ to $j'$ (i.e. $r_{i'j'} = 1$). Then,

$$\max_{\substack{(i,j)\in V\times V:\\ \widehat{\pi}(j)<\widehat{\pi}(i)}} \widehat{b}_{ij} = \widehat{b}_{j'i'}.$$

If (3.19) holds, then $\pi^*(j) < \pi^*(i)$ for all entries with $\widehat{b}_{ji} > \widehat{b}_{j'i'}$ and $\pi^*(j') < \pi^*(i')$. This leads to a cycle which contradicts our assumption that $\pi^*$ is a topological order. $\square$

### 3.5.2 First extension of the Greedy algorithm to infer the topol. order

In the following we extend the Greedy algorithm to make it more robust to infer a topological order $\pi$ of the nodes that belongs to the equivalence class $Q$ induced by the true underlying DAG $\mathcal{D}$.
Assume that there is a path from node $i$ to node $j$ in the true DAG $\mathcal{D}$, i.e. it holds that $b_{ij} > 0$, and that we estimated the ML coefficients

$$\widehat{b}_{ij} \approx \widehat{b}_{ji},$$

i.e. the values are close to each other. It might hold that the wrong estimated ML coefficient $\widehat{b}_{ji}$ is larger than the correct one $\widehat{b}_{ij}$, i.e. that $\widehat{b}_{ji} > \widehat{b}_{ij}$. Then, the Greedy algorithm might infer a wrong direction of the path. It first checks, whether it has already inferred a path from node $j$ to node $i$. If this is not the case, it decides to infer the path from $j$ to $i$ which is the wrong direction. In one of the subsequently iterations it considers the estimated ML coefficient $\widehat{b}_{ij}$ and notices that a path from $j$ to $i$ was already inferred.

This example shows that the Greedy algorithm might not be very resistant for estimated ML coefficients with $\widehat{b}_{ij} \approx \widehat{b}_{ji}$. For that reason, we extend the Greedy algorithm in order to make it more robust for those values.

**Algorithm 3.14** (Extended greedy algorithm).

```
Input:  The minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as defined in (3.12).
Output: Reachability matrix $R = (r_{ij})_{d\times d}$.

  1. Sort all entries $\widehat{b}_{ij}$ by size from large to small. Remove all diagonal
     entries $\widehat{b}_{ii} = 1$.

  2. Set $R = (r_{ij})_{d\times d} = (\mathbb{0})_{d\times d}$.

  3. For each $\widehat{b}_{ij}$ beginning with the largest,
         if $\widehat{b}_{ij}/\widehat{b}_{ji} > \varepsilon_1(n)$ and $\max(\widehat{b}_{ij}, \widehat{b}_{ji}) > \varepsilon_2(n)$ and $r_{ji} \neq 1$,
             set $r_{ij} = 1$ and update $R$ to a reachability matrix with
             Algorithm A.3;
         else, continue with $\widehat{b}_{ij}$ next in size.
```

**Remark 3.15.** (i) Observe that Algorithm 3.14 does not lead to an unique topological order of the nodes compared to Algorithm 3.12. The output is the reachability matrix $R$ which saves preliminary decisions. Algorithm 3.14 leaves out uncertain pairs $(i, j)$. Later on, we use the reachability matrix $R$ to reduce the set of feasible topological orders of the nodes and subsequently infer an unique one.

(ii) If we choose the thresholds $\varepsilon_1(n), \varepsilon_2(n)$ such that $\varepsilon_1(n), \varepsilon_2(n) = 0$ we are back to the Greedy algorithm. That is why Algorithm 3.14 can be considered as an extension of Algorithm 3.12.

With the condition

$$\max\{\widehat{b}_{ij}, \widehat{b}_{ji}\} > \varepsilon_2(n), \tag{3.20}$$

we identify pairs of nodes that are connected by a path with high probability. With the second condition

$$\frac{\widehat{b}_{ij}}{\widehat{b}_{ji}} > \varepsilon_1(n) \tag{3.21}$$

we infer the correct direction of the path with high probability. If we choose $\varepsilon_1(n) > 1$, the extended Greedy is more robust for values $\widehat{b}_{ij} \approx \widehat{b}_{ji}$ than before. How should we choose $\varepsilon_2(n)$? We know that at least $d(d-1)/2$ values of the true ML coefficient matrix $B$ are equal to zero. Hence, it seems reasonable to choose $\varepsilon_2(n)$ depending on the $d(d-1)/2$ smallest values of all estimated ML coefficients in the minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as defined in (3.12).

### 3.5.3 Second extension of the Greedy algorithm to infer the topol. order

#### Critical paths to find ancestral relations

For any critical path $p_{ij} = [i = k_0 \to ... \to k_u = j]$ between two nodes $i$ to $j$ and for any $l \in \{0, ..., u\}$ we have the equality $b_{ij} = b_{ik_l} b_{k_l j}$. Considering that and bearing in mind that $\mathbb{P}(X_j/X_i = b_{ij}) > 0$ whenever $i \in \mathrm{An}(j)$ (cf. Remark 3.2) it seems reasonable to search for critical paths by checking for triples of vertices $(i, k, j)$ such that

$$|\widehat{b}_{ij} - \widehat{b}_{ik}\widehat{b}_{kj}| = 0. \tag{3.22}$$

Correctly identifying such a triple would mean that $i \perp\!\!\!\perp j | k$ which could be used later to estimate ancestral relations of the DAG $\mathcal{D}$. Also, checking (3.22) could be done efficiently and easily be extended to other algorithms, e.g. the first extension of the Greedy algorithm (cf. Algorithm 3.14). However, we have seen in Example 3.10 that observing such an equality does not mean that there is really a critical path from $i$ to $j$ over $k$.
When we have noisy observations (cf. Chapter 4), we generally do not observe any equality as seen in (3.22). However, it holds that

$$|\widehat{b}_{ij} - \widehat{b}_{ik}\widehat{b}_{kj}| \to 0 \tag{3.23}$$

for $n \to \infty$. We could also use this to infer whether a path exists or not. But also here, it is almost impossible because by Corollary 3.7 we know that $\widehat{b}_{ij} \xrightarrow{a.s.} 0$ if $b_{ij} = 0$. By the idempotence it naturally holds that $\widehat{b}_{ij} - \widehat{b}_{ik}\widehat{b}_{kj} \geq 0$ and therefore (3.23) also holds even if there is no path from $i$ to $j$.
Therefore, the idea to search for critical paths to recover the structure of the DAG is inherently flawed and we omit it.

#### Concentration around the minimum value

As we have discussed earlier it holds that $\mathbb{P}(X_j/X_i = b_{ij}) > 0$, if $\boldsymbol{X}$ follows a recursive ML-model (cf. Remark 2.6). Therefore, for an i.i.d. sample $\boldsymbol{X}^1, ..., \boldsymbol{X}^n$ and defining $Y_{ij}^k = X_j^k / X_i^k$ for $k \in \{1, ..., n\}$, we will naturally have different observations of $Y_{ij}$ that realize the minimum, if $b_{ij} > 0$ and $n$ is large enough, since the probability to realize the minimum $r$-times is negative binomial (cf. Proposition 4.5 in Gissibl et al. [2018]).
We therefore want to use the concentration around the minimum by checking the ratio of order statistics of a sample $Y_{ij}^1, \dots, Y_{ij}^n$. For simplicity, we only consider the ratio of the two smallest values from the sample, i.e. $Y_{ij}^{(1)}$ and $Y_{ij}^{(2)}$.

**Lemma 3.16.** *Let $\boldsymbol{X}$ follow a recursive ML model as defined in* (2.2) *and let $\boldsymbol{X}^1, \dots, \boldsymbol{X}^n$ be an i.i.d. sample. Let $Y_{ij}^{(1)}, Y_{ij}^{(2)}$ be the ratio of the two smallest values from a sample $Y_{ij}^1, \dots, Y_{ij}^n$ with $Y_{ij}^k = X_j^k / X_i^k$ for $k \in \{1, ..., n\}$. Then, it holds that*

$$Y_{ij}^{(1)} / Y_{ij}^{(2)} \xrightarrow{\mathbb{P}} 1 \quad \textit{if and only if } b_{ij} > 0,$$

*for $n \to \infty$.*

*Proof.* First, let $b_{ij} > 0$, i.e. $i \in \mathrm{an}(j)$. By Lemma 3.2.(b) in Gissibl et al. [2018], the support of $Y_{ij}$ in this case is given by,

$$\mathrm{supp}(Y_{ij}) = [b_{ij}, \infty)$$

and we know that $\mathbb{P}(Y_{ij} = b_{ij}) > 0$. Hence, for $n \to \infty$ it holds that

$$\frac{Y_{ij}^{(1)}}{Y_{ij}^{(2)}} \overset{\text{a.s.}}{\to} 1.$$

For the other direction we first show that $Y_{ij}$ is regularly varying, i.e.

$$\lim_{t \downarrow 0} F(tx)/F(t) = x^{\alpha} \text{ for some } \alpha > 0, \tag{3.24}$$

whenever $b_{ij} = 0$ where $F$ is the distribution function of $Y_{ij}$. To do so, observe that the two sets

$$\mathrm{An}(i)_x^j = \{k \in \mathrm{An}(i) : b_{kj}/b_{ki} \le x\} \text{ and } \mathrm{An}(j) \setminus \mathrm{An}(i)_x^j$$

are non-empty for $x \downarrow 0$ and approach the sets $\mathrm{An}(j) \setminus \mathrm{An}(i)$ respectively $\mathrm{An}(j) \setminus (\mathrm{An}(i) \setminus \mathrm{An}(j)) = \mathrm{An}(j)$ which are both non-empty given that $b_{ij} = 0$.
Using Theorem 5.12. in Hartl [2015] we have,

$$\lim_{t \downarrow 0} \mathbb{P}(X_j/X_i \le tx)/\mathbb{P}(X_j/X_i \le t) = \lim_{t \downarrow 0} \frac{\frac{t^{\nu}x^{\nu}c_1}{t^{\nu}x^{\nu}c_1 + c_2}}{\frac{t^{\nu}c_1}{t^{\nu}c_1 + c_2}} = \lim_{t \downarrow 0} \frac{x^{\nu}(t^{\nu}c_1 + c_2)}{t^{\nu}x^{\nu}c_1 + c_2} = \frac{x^{\nu}c_2}{c_2} = x^{\nu}.$$

**Remark:** $Y_{ij}$ is regularly varying, because $X_i$ is regularly varying at infinity. The result, therefore, remains true whenever innovations are regularly varying at infinity.

Now, let $b_{ij} = 0$. For continuous distributions, the joint density for the two smallest elements is given by,

$$f_{Y_{ij}^{(1)}, Y_{ij}^{(2)}}(x, y) = n(n-1) f_{Y_{ij}}(x) f_{Y_{ij}}(y)(1 - F_{Y_{ij}}(y))^{n-2} \quad \text{for } x \le y$$

and defining $x = ry$ we have,

$$f_{Y_{ij}^{(1)}, Y_{ij}^{(2)}}(ry, y) = n(n-1) f_{Y_{ji}}(ry) f_{Y_{ij}}(y)(1 - F_{Y_{ij}}(y))^{n-2} \quad \text{for } r \in [0, 1].$$

By Corollary 3.7 we know that $Y_{ij}^{(1)} \xrightarrow{a.s.} 0$ and $Y_{ij}^{(2)} \xrightarrow{a.s.} 0$, if $b_{ij} = 0$.
Therefore, since $Y_{ij}$ is continuous on some interval $(0, x]$ with $x > 0$, we might assume that the distribution is continuous. Since $Y_{ij}$ is regularly varying at zero with exponent $\nu$, we can write $F_{Y_{ij}}(x) = x^{\nu} l(x)$ with $l(x)$ being some slowly varying function at zero. Therefore,

$$\mathbb{P}(Y_{ij}^{(1)}/Y_{ij}^{(2)} \le r) = \mathbb{P}(Y_{ij}^{(1)} \le rY_{ij}^{(2)}) = \int_0^c \int_0^r f_{Y_{ij}^{(1)}, Y_{ij}^{(1)}}(ry, y) dr dy$$

$$= \int_0^c n(n-1) F_{Y_{ij}}(ry) f_{Y_{ij}}(y)(1 - F_{Y_{ij}}(y))^{n-2} dy = \int_0^c n(n-1)\nu r^{\nu} y^{2\nu - 1}(1 - y^{\nu})^{n-2} l(y) dy,$$

for any $c > 0$, since $Y_{ij}^{(2)}$ becomes arbitrarily small for $n \to \infty$. Using $c = 1$ and Proposition 1.5.8 in Bingham et al. [1989] we get

$$\mathbb{P}(Y_{ij}^{(1)}/Y_{ij}^{(2)} \leq r) \sim c_2\, r^{\nu},$$

with $c_2$ being some constant. So the ratio is independent of the sample size $n$ and in particular it holds for any $\nu > 0$ and any $r \leq 1$ that $\mathbb{P}(Y_{ij}^{(1)}/Y_{ij}^{(2)} \leq r) > 0$. □

**Remark 3.17.** We can see that the probability for $r < 1$ gets smaller the bigger $\nu$ is. However, the probability stays positive, showing that the distribution function must converge to zero slower than by a polynomial.

In practice, we summarize the above theory for the non-noisy model in the following algorithm. The extended Greedy algorithm (Algorithm 3.14) is expanded and pooled with the new property. Due to this new property, we weaken the condition in step 3 of the the extended Greedy algorithm. Therefore, we now also make decisions for those pairs of indices, for which the extended Greedy algorithm alone does not make a decision due to uncertainties. Therefore, the reachability matrix $R$, which we obtain as output, is more complete than the reachability matrix which is the output from the first extension of the Greedy algorithm. As a consequence, we reduce the set of feasible topological orders much more.

**Algorithm 3.18** (Expansion of Algorithm 3.14).

```
Input:  The minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as defined in (3.12) and the
        i.i.d. sample $\boldsymbol{X}^1, \dots, \boldsymbol{X}^n$.
Output: Reachability matrix $R$.

  1. For all $i \in V$,
         for all $j \in V$,
            for all $s \in \{1, \dots, n\}$,
               compute the ratios $Y_{ij}^s = X_i^s / X_j^s$;
            end for-loop;
         end for-loop;
     end for-loop;

  2. Sort all entries $\widehat{b}_{ij}$ by size from large to small. Remove all diagonal
     entries $\widehat{b}_{ii} = 1$.

  3. Set $R = (r_{ij})_{d\times d} = (\mathbb{0})_{d\times d}$.

  4. For each $\widehat{b}_{ij}$ beginning with the largest,
         set $\lambda_1 = Y_{ij}^{(1)}/Y_{ij}^{(2)}$ and $\lambda_2 = Y_{ji}^{(1)}/Y_{ji}^{(2)}$,
         if $r_{ji} \neq 1$ and $((\lambda_1 = 1$ and $\lambda_2 > 1)$ or $(\widehat{b}_{ij}/\widehat{b}_{ji} > \varepsilon_1(n)$ and $\max(\widehat{b}_{ij}, \widehat{b}_{ji}) > \varepsilon_2(n)))$,
            set $r_{ij} = 1$ and update $R$ to a reachability matrix
            with Algorithm A.3;
         else, continue with $\widehat{b}_{ij}$ next in size.
```

**Ignore estimated ML coefficients with $\widehat{b}_{ij}, \widehat{b}_{ji} \approx 0$**

We now address those pairs of indices $(i, j)$ with $\widehat{b}_{ij}, \widehat{b}_{ji} \approx 0$. We use this as an extension of Algorithm 3.18 to fill the reachability matrix even more and to reduce the set of feasible topological orders also even more. If $r_{ij} = r_{ji} = 0$, we set a random direction of the pair of nodes $i, j$ (i.e. either $(i, j)$ or $(j, i)$). However, we cannot do this in the beginning, i.e. before Algorithm 3.18, as the following example shows.

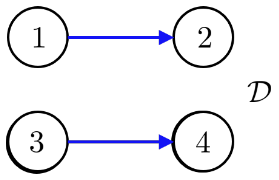


Figure 3.1: The pairs 1, 3 and 1, 4 and 2, 3 and 2, 4 are not connected.

**Example 3.19.** Consider the DAG in Figure 3.1. Assume that we correctly infer that the nodes 2, 3 and the nodes 1, 3 are not connected by any path. Then we arbitrarily set 2 before 3 and 3 before 1 and obtain the topological order

$$(\dots, 2, \dots, 3, \dots, 1, \dots),$$

which does not belong to the equivalence class induced by the true underlying DAG in Figure 3.1.

Since in Algorithm 3.18 we sort the entries by size, this example cannot occur anymore, if we use it after Algorithm 3.18.

**Algorithm 3.20** (Continuation of Algorithm 3.18).

```
Input:  Reachability matrix $R = (r_{ij})_{d \times d}$ as output from Algorithm 3.18 and
        the minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d \times d}$ as defined in (3.12).
Output: Modified reachability matrix $R$.

  1. For all $\widehat{b}_{ij} > 0$,
         if $\max\{\widehat{b}_{ij}, \widehat{b}_{ji}\} < \varepsilon_3(n)$ and $r_{ji} = 0$ and $r_{ij} = 0$,
             if $\widehat{b}_{ij} > \widehat{b}_{ji}$,
                 set $r_{ij} = 1$ and update $R$ to a reachability matrix with
                 Algorithm A.3;
             else,
                 set $r_{ji} = 1$ and update $R$ to a reachability matrix with
                 Algorithm A.3;
         else, continue with the next $\widehat{b}_{ij} > 0$.
```

How should we choose the threshold $\varepsilon_3(n)$? We propose to set $\varepsilon_3(n)$ equal to the geometric mean of the $d(d-1)/2$ smallest estimated ML coefficients in $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as defined in (3.12), i.e.

$$\varepsilon_3(n) = \left( \prod_{k=1}^{d(d-1)/2} \widehat{b}_{ij}^{(k)} \right)^{1/(d(d-1)/2)}, \tag{3.25}$$

where $\widehat{b}_{ij}^{(k)}$ denotes the $k$-th smallest estimated ML coefficient.
Setting a direction for pairs of estimated ML coefficients with $\max\{\widehat{b}_{ij}, \widehat{b}_{ji}\} < \varepsilon_3$ (i.e. setting $r_{ij} = 1$ or $r_{ji} = 1$) improves running time.

If an entry $r_{ij}$ of the reachability matrix $R$ which we obtain as output from Algorithm 3.20 is equal to one, i.e. $r_{ij} = 1$, we made a decision for this pair of indices. Therefore, it does not change anything, if we set the transposed entry $\widehat{b}_{ji}$ of the minimum ratio estimator equal to zero, i.e. $\widehat{b}_{ji} = 0$. We will see later on that this simplifies further calculations.

**Algorithm 3.21** (Continuation of Algorithm 3.20)**.**

```
Input:  Reachability matrix $R = (r_{ij})_{d\times d}$ and the minimum ratio estimator
        $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ as defined in (3.12).
Output: Modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$.

  1. For all $i, j \in V$ with $r_{ij} = 1$,
         set $\widehat{b}_{ji} = 0$.
```

## 3.5.4 Maximize the sum of ML coefficients to infer the topol. order

After applying Algorithms 3.18, 3.20 and 3.21 what we have at hand is a reachability matrix $R = (r_{ij})_{d\times d}$ corresponding to a DAG and a modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$. We now use both of them to infer a topological order of the nodes.

The set of feasible topological orders was reduced already by Algorithms 3.18 and 3.20, however there might be still more than one feasible topological order left due to uncertain pairs of indices $i, j \in V$. Thus, it seems robust to maximize the sum of estimated ML coefficients of the modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ obtained as output from Algorithm 3.21 to infer an unique topological order of the nodes.

**Optimization Problem 3.22.** Given a reachability matrix $R = (r_{ij})_{d\times d}$ corresponding to a DAG and the modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$, find the topological order $\pi^*$ of the nodes $(1, \dots, d)$ such that,

$$\pi^* = \arg\max_{\pi\in\Pi} \sum_{\pi(i)<\pi(j)} \widehat{b}_{ij}, \qquad \text{subject to } \pi^* \in Q_R, \tag{3.26}$$

where $Q_R$ denotes the equivalence class of topological orders induced by the reachability matrix $R$, $\Pi$ is the set of all topological orders of size $d$ and $\pi(i)$ denotes the position of $i$ in the topological order $\pi$.

**Lemma 3.23.** *Optimization Problem 3.22 is NP-hard.*

*Proof.* Follows from Theorem 17 in Mishra and Sikdar [2004]. □

One reason why we set estimated ML coefficients in Algorithm 3.21 equal to zero for which we make a decision is that we do not want to obtain a contradiction with extensions of the algorithm that follow in the next subsection (Algorithm 3.25). Another reason becomes clear now: if we make a decision for a pair of nodes $i, j \in V$, for instance we decided to set $(i, j)$, then we do not want the wrong estimated ML coefficient $\widehat{b}_{ji}$ to influence the value of the sum in (3.26) negatively.

A first idea to solve Optimization Problem 3.22 is a Brute-Force search, where we compute the sum in (3.26) for all topological orders of size $d$. The topological order $\pi$ that returns the maximal value of the sum in (3.26) and for which holds that $\pi \in Q_R$, is an estimate for the true topological order of the nodes of the unknown DAG $\mathcal{D}$.
The problem of this method is its running time. Assume that we have a DAG $\mathcal{D} = (V, E)$ with $|V| = d \in \mathbb{N}$ nodes. If we compute the sum in (3.26) for all topological orders, we obtain a running time of $\mathcal{O}(d!)$, since there are $d!$-many topological orders. The aim for the next subsection is to develop an algorithm that keeps the set of feasible solutions small and thus becomes much faster than the Brute-Force search.

**Algorithm 3.24** (Brute-Force).

```
Input:  The modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ and the
        reachability matrix $R = (r_{ij})_{d\times d}$.
Output: An almost surely maximizing topological order for
        Optimization Problem 3.22.

  1. Generate all topological orders of size $d$.

  2. For each of these topological orders $\pi$, compute the sum

                    $\sum_{\pi(i)<\pi(j)} \widehat{b}_{ij}$.

     If it is larger than the sum of the previous topological order and if this
     topological order corresponds to the reachability matrix $R = (r_{ij})_{d\times d}$, it
     is a potential solution of Optimization Problem 3.22.
```

Since the reachability matrix $R = (r_{ij})_{d\times d}$ corresponds to a DAG, Algorithm 3.24 solves Optimization Problem 3.22.

### 3.5.5 The Branch & Bound algorithm to infer the topol. order

The Branch & Bound algorithm has its origin in the field of combinatorial optimization. The general idea is to split the set of all possible solutions into subsets (branches). Using appropriate bounds, the aim is to delete specific branches which do not satisfy the bounds and can therefore not be optimal (cf. Korte and Vygen [2018], Section 21.6). The computation time compared to the Brute-Force method is reduced significantly.

What we are going to use to delete specific branches is the modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ and the reachability matrix $R = (r_{ij})_{d\times d}$ corresponding to a DAG. For the components of the modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ it either holds that

$$\widehat{b}_{ij} < \widehat{b}_{ji} \quad \text{or} \quad \widehat{b}_{ij} > \widehat{b}_{ji} \quad \text{or} \quad \widehat{b}_{ij} = \widehat{b}_{ji}$$

for any pair of distinct nodes $i, j \in V$.
For a fixed pair of distinct nodes $k, l \in V$ we now assume that we observe $\widehat{b}_{kl} > \widehat{b}_{lk}$. Then, it holds for any $\pi \in \Pi$ with $|\pi(k) - \pi(l)| = 1$ that

$$\widehat{b}_{kl} + \sum_{\substack{\pi(i)<\pi(j)\\(i,j)\neq(k,l)}} \widehat{b}_{ij} \quad > \quad \widehat{b}_{lk} + \sum_{\substack{\pi(i)<\pi(j)\\(i,j)\neq(l,k)}} \widehat{b}_{ij},$$

since the second term on both sides of the inequality has the the same value. Thus, the component $\widehat{b}_{lk}$ does not contribute to the maximum of the sum in (3.26) and we exclude all topological orders $\pi$ from $\Pi$ where the sequence $(l, k)$ is immediately consecutive, i.e. all topological orders with $(l, k, \ldots)$, $(\ldots, l, k, \ldots)$ or $(\ldots, l, k)$. We should *not* exclude any topological order where $l$ is not immediately consecutive before $k$, i.e. $(\ldots, l, \ldots, k, \ldots)$, since then we might not maximize the sum in (3.26).
The case $\widehat{b}_{lk} > \widehat{b}_{kl}$ with the strict inequality in the other direction is analogously to the first one.
If we observe that $\widehat{b}_{ij} = \widehat{b}_{ji}$, then it does not matter whether we exclude all topological orders with $(k, l)$ or with $(l, k)$.

We introduce a permutation tree which we use to exclude topological orders in a systematical and efficient manner. Each branch represents one topological order.

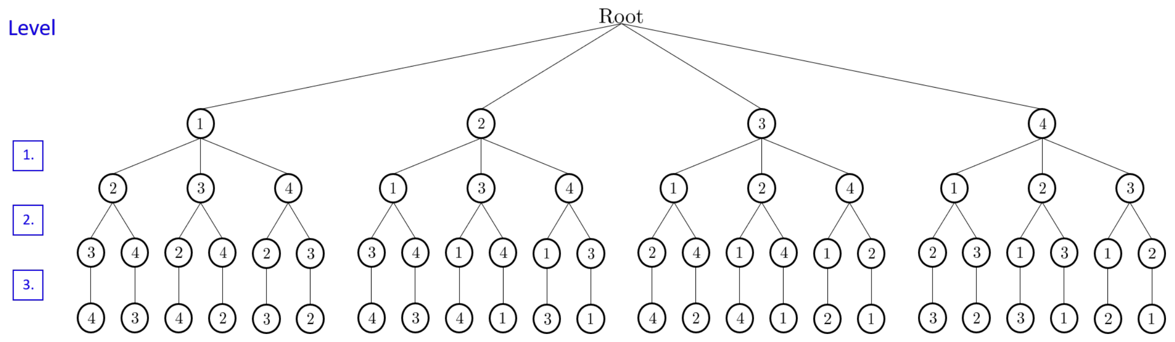


Figure 3.2: A permutation tree for a DAG with $d = 4$ nodes.

If we cut one branch in the first level of the tree, we prune the complete subtree below and thus we remove $\frac{d!}{d(d-1)} = (d-2)!$ topological orders from $\Pi$. If we cut one branch in the second level, we again prune the complete subtree below the second level and remove another $\frac{d!}{d(d-1)(d-2)} = (d-3)!$ topological orders from $\Pi$ and if we cut one branch in the $k$-th level, we remove $\frac{d!}{d(d-1)\ldots(d-k)} = (d-k-1)!$ topological orders from $\Pi$ by pruning the complete subtree below the $k$-th level.

**Algorithm 3.25** (Pure Branch & Bound)**.**

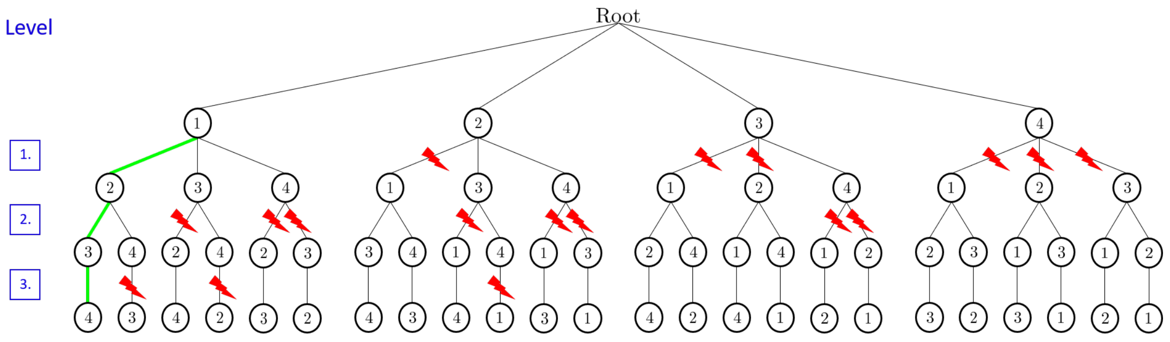
```
Input:  The modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$.
Output: A matrix $\alpha = (\alpha_{ij})_{d\times d}$ which saves the deleted branches.
  1. Set $\alpha = (\alpha_{ij})_{d\times d} = (\mathbb{0})_{d\times d}$.
  2. Start with the first level which represents the first vertex in the
     topological order. For an edge $(i,j)$ delete the branch if and only if
     $\widehat{b}_{ji} > \widehat{b}_{ij}$ and $\widehat{b}_{ji} > 0$, i.e. we set $\alpha_{ij} = 1$.
     Go to the next level.
  3. For level $k \in \{2,\ldots,d-1\}$ and a given partial topological order $(i_1,\ldots,i_k)$:
     repeat step 2. for all existing branches.
```

**Remark 3.26.** In Algorithm 3.25 we start with an empty matrix $\alpha = (\alpha_{ij})_{d\times d} = (\mathbb{0})_{d\times d}$. Deleting a branch means that we set the corresponding entry in the matrix $\alpha$ equal to 1.

**Example 3.27** (Pure Branch & Bound)**.** Consider the permutation tree in Figure 3.2 ($d = 4$) and assume that we have the modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ after Algorithm 3.20 at hand. If we apply the Brute-Force method introduced in Subsection 3.5.4, we compute the sum in (3.26) for all possible $4! = 24$ topological orders. Now assume that we observe,

$$\widehat{b}_{ji} < \widehat{b}_{ij} \text{ and } \widehat{b}_{ij} > 0$$

for all pairs $i,j \in \{1,2,3,4\}$ with $i < j$. Corresponding to the pure Branch & Bound algorithm we exclude branches that contain a topological order where $j$ directly occurs before $i$. In the first level of the tree these are the branches starting with $(2,1,\ldots)$, $(3,1,\ldots)$, $(3,2,\ldots)$, $(4,1,\ldots)$, $(4,2,\ldots)$, $(4,3,\ldots)$, so we already exclude six branches and prune its subtrees, such that we already exclude twelve topological orders after the first level. In the second level of the tree we exclude another eight branches and prune its subtrees, such that we exclude another eight topological orders. In the third level we exclude another three branches and thus three more topological orders:



We are left with the branch $(1, 2, 3, 4)$ which is the unique solution.

When we speak of the *Branch & Bound* algorithm in the following, we always mean not only Algorithm 3.25, but we mean the composition of Algorithms 3.18, 3.20, 3.21, 3.25 and 3.28.

We implement the Branch & Bound algorithm by using a *Depth-First Search* for trees (cf. Cormen et al. [2009], Section 22.3). Using that procedure, we start in the root of the tree and explore each branch into the depth as far as possible. If on a certain level of the tree we cut the branch and thus prune the subtree below, we track back to that node which starts a new branch that we have not explored yet. Moreover, if we run over a whole branch and successfully end, we also track back to that node which starts a new branch that we have not explored yet.

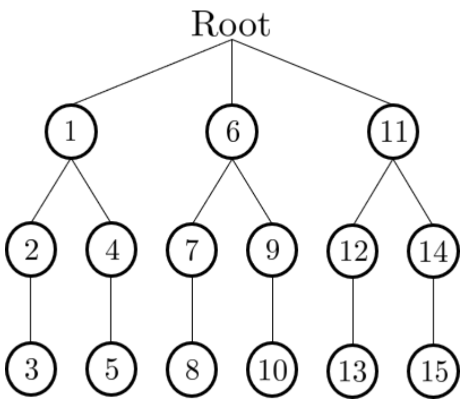


Figure 3.3: Order in which the nodes are visited performing a Depth-First Search.

For a better understanding, we present the pseudo code for the Depth-First Search adapted to our situation. The crucial steps where we cut branches from the permutation tree are lines 6 and 8.

**Algorithm 3.28:** Depth-First Search

**input :** The matrix $\alpha = (\alpha_{ij})_{d\times d}$ as output from Algorithm 3.25, the reachability matrix $R = (r_{ij})_{d\times d}$ and the modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$.

**output:** Reduced set of feasible solutions for Optimization Problem 3.22.

```
1 function DFS($\alpha$,$\widehat{B}$,$R$):
      // a set with all nodes from 1 to $d$
2     set $M = \{1, \dots, d\}$
      // $\pi$ starts as empty tuple, but it grows in recursion
3     set $\pi = ()$
4     function Inner($\pi$):
5         set $l = \text{length}(\pi)$ // current length of the topological order
```

```
6          if $l > 1$ and $\alpha_{\pi(l-1),\pi(l)} = 1$ then
7              return( )
8          else if $l \geq 1$ and if there is a first $k \in \{1, \dots, d\}$ such that $r_{k,\pi(l)} = 1$
                           and $k$ is not already in $\pi$ then
9              return( )
           // As soon as $\pi$ is a full topological order that was not rejected, we
           return it.
10         if $l = d$ then
11             return($\pi$)
           // If $\pi$ is not full yet, and was also not rejected yet, we append to it
           all possible indices that are not already in there and recurse
12         return($rec$ for each $i$ in $(M \setminus \pi)$ and
13                         for $rec$ in Inner($\pi \cup \{i\}$))
14     return(Inner())
```

For the condition in line 6 the Branch & Bound condition has to be satisfied for the last two elements that have been added to the partial topological order $\pi$, i.e. $\alpha_{\pi(l-1)\pi(l)} = 1$. Then, we reject the current partial topological order and prune its subtree out of the search space by taking the set difference $M \setminus \pi$ in the recursive part in lines 12 and 13.
To explain what happens in line 8 assume it holds that $r_{ij} = 1$. There has to be at least one element in the partial topological order. Now, two cases may occur. In the first case assume, that the vertex $j$ is added to the topological order on first position. Then, we immediately delete all topological orders that start with $j$ (since $r_{ij} = 1$) and prune its subtrees out of the search space. In the second case assume, that $j$ is added to the topological order not on first position but on a position after the first and that $i$ was not added before. Then, we immediately delete all topological orders with $\pi(j) < \pi(i)$ and prune its subtrees out of the search space.

**Theorem 3.29.** *Let $S$ denote the set of feasible solutions for Optimization Problem 3.22 obtained from Algorithm 3.28. Then it holds that $S \neq \varnothing$.*

*Proof.* Since the reachability matrix $R = (r_{ij})_{d \times d}$ corresponds to a DAG $\mathcal{D}$, it is enough to show that there is no contradiction between pure Branch & Bound (Algorithm 3.25) and Algorithms 3.18, 3.20 and 3.21.
Assume we have given a topological order $\pi = (1, 2, \dots, d)$ such that $\pi$ does not violate $R$ and assume that $\pi \notin S$. Then there is a pair of indices $(i, i+1)$ such that $\widehat{b}_{i+1,i} < \widehat{b}_{i,i+1}$. Now observe that flipping the two vertices $i$ and $i+1$, i.e. $\pi_2 = (1, \dots, , i-1, i+1, i, i+2, \dots, d)$ does not violate $R$. The reason is that for all $k \in \{1, \dots, d\} \setminus \{i, i+1\}$ it holds that

$$\pi(k) < \pi(i) \text{ if and only if } \pi_2(k) < \pi_2(i) \text{ and}$$
$$\pi(k) > \pi(i+1) \text{ if and only if } \pi_2(k) > \pi_2(i+1),$$

and $\widehat{b}_{i+1,i} = 0$ whenever $r_{i,i+1} = 1$. Therefore, we can flip neighboring pairs of vertices that contradict each other until we get a non-contradictory topological order, which still satisfies $R$. □

### 3.5.6 Dynamic programming to infer the topol. order

In this section we introduce another method to infer the topological order of the nodes of an unknown DAG. This method also solves Optimization Problem 3.22. We do this in order to have a comparison of running times with the Branch & Bound algorithm and to see that we gained an improvement. We follow Cormen et al. [2009], chapter 15.
The dynamic programming method divides an optimization problem into recurring subproblems and saves the subsolutions in a systematic manner in order to use them to solve the original problem. This method is in particular useful, if the original optimization problem consists of many subproblems of the same kind.
Adapted to our situation, where the aim is to find a topological order $\pi^*$ that maximizes the sum Optimization Problem 3.22 with $\pi^* \in Q_R$, where $Q_R$ denotes the equivalence class of topological orders induced by $R$, we divide it into recurring subproblems as follows.

**Algorithm 3.30** (Dynamic Programming).

Input: The reachability matrix $R = (r_{ij})_{d\times d}$ and the modified minimum ratio estimator $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$.
Output: An unqiue topological order $\widehat{\pi}$.

1. In the first iteration ($k = 2$) we have $\binom{d}{2}$ many subproblems: for all subsets $\{i, j\}$ of the powerset $\mathcal{P}\big(\{1, \dots, d\}\big)$ with two elements find the $\binom{d}{2}$ topological orders that maximize the sum and satisfy the reachability matrix $R$, that is either $(i, j)$ or $(j, i)$. We obtain $\binom{d}{2}$ subsolutions in the first iteration.
   Save the values of the maximized sums and the corresponding topological orders of size two.

2. For $k = 3, \dots, d$ we have to solve $\binom{d}{k}$ subproblems based on the topological orders of size $k - 1$ and the corresponding values of the sums found in the previous iteration.
   We want to find the sum-maximizing topological order $\pi_S$ of the subset $S = \{i_1, \dots, i_k\}$ which satisfies $R$. For this purpose, we divide $S$ into $k$ subsets each of size $k - 1$,
$$S_1 = \{i_2, \dots, i_k\}, S_2 = \{i_1, i_3, \dots, i_k\}, \dots, S_k = \{i_1, \dots, i_{k-1}\}.$$
   Assume that $\pi_l : \pi^{-1}(1), \pi^{-1}(2), \dots, \pi^1(k-1)$, $l \in \{1, \dots, k\}$, is the sum-maximizing topological order for the subset $S_l$ with corresponding value of the maximized sum equal to $\Sigma_l$ and that $\pi_l$ satisfies $R$. Then, for the missing value $i_l$, the following sum is computed for each set $S_l$:
$$\Sigma_l + \sum_{l=1}^{k-1} \widehat{b}_{\pi(l)i_l}.$$
   The topological order that generates the maximal value, i.e.
$$\max_{l\in\{1,\dots,k\}} \Sigma_l + \sum_{l=1}^{k-1} \widehat{b}_{\pi(l)i_l},$$

```
    and satisfies $R$, is saved. Also the value of the corresponding sum is
    saved for the next iteration.
```

Since the reachability matrix $R = (r_{ij})_{d\times d}$ corresponds to a DAG, Algorithm 3.30 solves Optimization Problem 3.22.

Algorithm 3.30 has to save

$$2\binom{d}{k-1} + 2\binom{d}{k}$$

elements at the same time in iteration $k \in \{1,\dots,d\}$. In detail these are $\binom{d}{k-1}$ values of the maximized sums together with the corresponding topological orders in iteration $k-1$ and the $\binom{d}{k}$ values of the maximized sums with the corresponding topological orders in iteration $k$. This needs far more additional memory at one step than the Brute-Force method uses. But it is also one of the main advantages of the dynamic programming method, since it makes use of a so-called *time-memory trade-off*. This means it uses additional memory to save computation time.
The worst-case running time to solve Optimization Problem 3.22 is $\mathcal{O}\big(d^2 2^d\big)$. In each iteration $k$ we consider $(k-1)$ sub-topological orders. For each of these sub-topological orders, we calculate the sum. This leads to a total number of calculations of

$$\sum_{k=2}^{d} \binom{d}{k}(k-1)^2,$$

and thus the worst case-running time is $\mathcal{O}\big(d^2 2^d\big)$. It is therefore less than $\mathcal{O}(\,d!\,)$, which was the worst-case running time of the Brute-Force method.

# Chapter 4

# Estimation of recursive ML models with noise

In this chapter we extend the recursive ML model introduced in Section 2.1 by adding multiplicative noise terms. We apply the estimation methods and algorithms to real world data in future. However, in real data sets it is unreasonable to expect to observe atoms. Therefore, we add noise terms to make it more realistic.
The implementation of these methods and performing simulations in order to assess their behavior with noise is the second main part of this thesis.

We use noise terms in two different models, and for both models we investigate the three different settings.

(1.) In the first setting we know the underlying DAG and estimate the ML coefficients $B = (b_{ij})_{d \times d}$.

(2.) In the second setting we assume that we know the the topological order of the nodes. The aim is to infer the minimum ML DAG $\mathcal{D}^B$. For this purpose, we first estimate the ML coefficient matrix $B$ and in a second step, we estimate the edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$ using the estimate of $B$.

(3.) In the third setting we do not know anything about the underlying DAG $\mathcal{D}$. The aim is the estimation of a topological order of the nodes that belongs to the equivalence class $Q$ induced by the true DAG $\mathcal{D}$. Then, we estimate the minimum ML DAG $\mathcal{D}^B$ as in the second setting using the estimated topological order of the nodes.

## 4.1 Estimation of ML coefficients in the recursive noise model

In the first noise model we define the noise terms in a recursive manner. This is a very natural approach, since in a regular additive noise model $X = Yc + \varepsilon$ it is done in the same way.

Without loss of generality we consider a well-ordered DAG and define:

$$\begin{aligned} \widetilde{X}_1 &= Z_1\varepsilon_1, \\ \widetilde{X}_i &= \Big(\bigvee_{k\in \mathrm{pa}(i)} c_{ki}\widetilde{X}_k \vee Z_i\Big)\varepsilon_i, \quad i\in\{2,\dots,d\}, \end{aligned} \tag{4.1}$$

where the sequence $(Z_i)_{i=1}^d$ are i.i.d. innovations with positive support $(0,\infty)$ and atom-free distributions, $c_{ki}$ are positive edge-weights as in Definition 2.2 and $(\varepsilon_i)_{i=1}^d$ are i.i.d. noise terms.

We mentioned in Chapter 2 that a recursive ML model is a specific recursive structural equation model, which is suitable to assess extreme risk propagating through a network and where sums are replaced by maxima. Therefore, it is natural to choose EVDs with positive support for the i.i.d. innovations $(Z_i)_{i=1}^d$. This is the case due to (2.3) and when we extend the recursive ML model from Section 2.1 and consider noisy observations, we want to preserve this property. Hence, we choose for the distribution of the i.i.d. noise terms $(\varepsilon_i)_{i=1}^d$ any continuous distribution such that,

$$\mathrm{supp}(\varepsilon_i) = [1,\infty), \quad i\in\{1,\dots,d\}. \tag{4.2}$$

With a result from Buck and Klüppelberg [2019] it holds that $\widetilde{X}_i \geq X_i$ for all $i\in\{1,\dots,d\}$ and for the support of the ratio of two components of $\widetilde{\boldsymbol{X}}$ it holds that

$$\mathrm{supp}\bigg(\frac{\widetilde{X}_j}{\widetilde{X}_i}\bigg) = [b_{ij},\infty), \quad i,j\in V,\ i\in \mathrm{an}(j), \tag{4.3}$$

and

$$\mathrm{supp}\bigg(\frac{\widetilde{X}_j}{\widetilde{X}_l}\bigg) = \big(0,\tfrac{1}{b_{jl}}\big] \quad j,l\in V,\ l\in \mathrm{de}(j). \tag{4.4}$$

### 4.1.1 Estimation of ML coefficients with known DAG

We assume that we are in the first setting where we know the underlying DAG $\mathcal{D}$. Let $\widetilde{\boldsymbol{X}} = (\widetilde{X}_1,\dots,\widetilde{X}_d)$ follow a recursive ML model on a DAG $\mathcal{D}$ with recursive noise as defined in (4.1). Assume we have given an i.i.d. sample $\widetilde{\boldsymbol{X}}^1,\dots,\widetilde{\boldsymbol{X}}^n$. The support of the ratio of two components of $\widetilde{\boldsymbol{X}}$ (cf. (4.3) and (4.4)) is equal to the support of the ratio of two components of $\boldsymbol{X}$ in the non-noisy model (cf. (3.4) and (3.6)). Therefore, at first sight we could use the estimator similar to the one as defined in the non-noisy recursive ML model (cf. (3.7)),

$$\breve{b}^*_{ij} = \bigwedge_{s=1}^n \frac{\widetilde{X}_j^s}{\widetilde{X}_i^s} \quad \text{for } i\in\mathrm{an}(j), \quad \breve{b}^*_{ii}=1 \quad \text{and} \quad \breve{b}^*_{ij}=0 \quad \text{for } i\in V\setminus \mathrm{An}(j). \tag{4.5}$$

However, the same problem as in the non-noisy model occurs. For a small sample size $n$ we may obtain an estimate for the ML coefficient matrix $B$, which does not belong to

$\mathcal{B}(\mathcal{D})$, the class of ML coefficient matrices of all recursive ML models on $\mathcal{D}$. Therefore, we exploit the fact that we know the underlying DAG $\mathcal{D}$ and apply Lemma 3.5 in order to compute an estimate for the ML coefficient matrix $B$.
Analogously to the non-noisy model we first compute all ML coefficients corresponding to edges in the DAG $\mathcal{D}$, i.e.,

$$\breve{B}_0^* = (\breve{b}_{ij}^* \mathbb{1}_{\mathrm{pa}(j)}(i))_{d\times d},$$

where $\breve{b}_{ij}^*$ is given by (4.5). Then, we consider a path $p_{ij} = [i = k_0 \to k_1 \to \ldots \to k_u = j]$ of length $u \geq 2$ and a realization $\widetilde{\boldsymbol{X}}^t = (\widetilde{X}_1^t, \ldots, \widetilde{X}_d^t)$ such that $\breve{b}_{ij}^* \widetilde{X}_i^t = \widetilde{X}_j^t, i \in \mathrm{an}(j)$, and multiply the entries of $\breve{B}_0^*$ along this path. Thus, the new estimator $\widehat{B}^*$ is given by

$$\widehat{B}^* = (I_d \vee \breve{B}_0^*)^{\odot(d-1)}.$$

Then, by Lemma 3.5 the estimate $\widehat{B}^*$ always belongs to $\mathcal{B}(\mathcal{D})$. The entries $\widehat{b}_{ij}^*$ of $\widehat{B}^*$ are explicitly defined by

$$\begin{aligned} &\widehat{b}_{ii}^* = 1, \quad \widehat{b}_{ij}^* = 0 \text{ for } i \in V \setminus \mathrm{An}(j), \quad \widehat{b}_{ij}^* = \bigwedge_{s=1}^{n} \frac{\widetilde{X}_j^s}{\widetilde{X}_i^s}, \text{ for } i \in \mathrm{pa}(j) \quad \text{ and} \\ &\widehat{b}_{ij}^* = \bigvee_{p_{ij} \in P_{ij}} \widehat{d}^*(p_{ij}), \text{ for } i \in \mathrm{an}(j) \setminus \mathrm{pa}(j), \end{aligned} \tag{4.6}$$

where $\widehat{d}^*(p_{ij}) = \prod_{l=0}^{u-1} \widehat{b}_{k_l k_{l+1}}^*$ for a path $p_{ij} = [i = k_0 \to k_1 \to \ldots \to k_u = j]$.
Furthermore, we obtain the inequality

$$b_{ij} \leq \widehat{b}_{ij}^* \leq \breve{b}_{ij}^*, \quad \text{for all } i, j \in V. \tag{4.7}$$

The matrix $\widehat{B}^*$ is an estimate for the ML coefficient matrix $B$ and due to the inequality (4.7), $\widehat{B}^*$ is always preferable to $\breve{B}^*$.

**Corollary 4.1.** *Let $\widetilde{\boldsymbol{X}} = (\widetilde{X}_1, \ldots, \widetilde{X}_d)$ be given by a recursive ML model on a DAG $\mathcal{D}$ with recursive noise as defined in* (4.1) *and let $\widetilde{\boldsymbol{X}}^1, \ldots, \widetilde{\boldsymbol{X}}^n$ be an i.i.d. sample of $\widetilde{\boldsymbol{X}}$. Furthermore, let $\widehat{B}^* = (\widehat{b}_{ij}^*)_{d\times d}$ be defined as in* (4.6). *Then, for $i \in V$ and $i \in \mathrm{an}(j)$, it holds that $\widehat{b}_{ij}^* \overset{\text{a.s.}}{\to} b_{ij}^*$, as $n \to \infty$.*

*Proof.* Analogously to the proof of Lemma 3.6. □

## 4.1.2 Estimation of ML coefficients with known topol. order

If we are in the second setting where we assume to know the topological order of the nodes, the starting point is the minimum ratio estimator defined by

$$\widehat{b}_{ij}^* = \bigwedge_{s=1}^{n} \frac{\widetilde{X}_j^s}{\widetilde{X}_i^s}, \quad i, j \in V. \tag{4.8}$$

**Corollary 4.2.** *The minimum ratio estimator as defined in* (4.8) *converges almost surely to the true ML coefficient $b_{ij}$ for $i \in \operatorname{an}(j)$ and it converges to zero for $i \in V \setminus \operatorname{An}(j)$.*

*Proof.* Analogously to the proof of Corollary 3.7. □

The aim is to estimate the minimum ML DAG $\mathcal{D}^B$, since we know from Section 3.1 that only $\mathcal{D}^B$ is identifiable and that it has the same distribution as the DAG $\mathcal{D}$. The DAG is $\mathcal{D}$ is not identifiable.
First, we estimate the ML coefficient matrix $B = (\widehat{b}_{ij})$ with Algorithm 3.8. In a second step, we infer which edges in the minimum ML DAG $\mathcal{D}^B$ exist by applying Algorithm 3.9. However, we need a modification in step 2, since we are in the noise model and do not observe any atoms. Thus, we do not observe an equality $\widehat{b}^*_{ij} = \bigvee_k \widehat{b}^*_{ik}\widehat{b}^*_{kj}$. For that reason, we choose a threshold $\varepsilon(n)$ depending on the sample size $n$ such that $\varepsilon(n) \to 0$ as $n \to \infty$ and

$$\frac{|\widehat{b}^*_{ij} - \widehat{b}^*_{ik}\widehat{b}^*_{kj}|}{\varepsilon(n)} \xrightarrow{\mathbb{P}} 0, \quad i, k, j \in V,$$

as $n \to \infty$. If the absolute value of the difference between $\widehat{b}^*_{ij}$ and $\bigvee_k \widehat{b}^*_{ik}\widehat{b}^*_{kj}$ is smaller than this threshold, we set $\widehat{c}^{\widehat{B}^*}_{ij} = 0$.

**Algorithm 4.3** (Estimate the minimum ML DAG $\mathcal{D}^B$ in the recursive noise model).

```
Input:  The matrix $\widehat{B}^* = (\widehat{b}^*_{ij})_{d\times d}$ as output from Algorithm 3.8.
Output: An estimate for the edge-weight matrix $C^B$ of the minimum ML DAG.

  1. Set $\widehat{C}^{\widehat{B}^*} = \widehat{B}^*$.

  2. For all $i, j, k \in V$ with $\pi(i)<\pi(k)<\pi(j)$,
         if $|\widehat{b}^*_{ij} - \bigvee_k \widehat{b}^*_{ik}\widehat{b}^*_{kj}| < \varepsilon(n)$, set $\widehat{c}^{\widehat{B}^*}_{ij} = 0$.
```

## 4.1.3 Estimation of the topol. order

If we are in the third setting where we assume to know nothing about the underlying DAG and where we want to estimate a topological order $\pi \in Q$ of the nodes, we start with the minimum ratio estimator as defined in (4.8). The equivalence class $Q$ is induced by the true underlying DAG $\mathcal{D}$ (cf. Section 3.2).
We proceed, as we proceeded in the non-noisy model, i.e. we apply the Branch & Bound algorithm (composition of Algorithms 3.18, 3.20, 3.25 and 3.28). However, we adjust Algorithm 3.18 as follows, since we do not observe any atoms in the noise model. In the if-condition in step 4. we replace the condition

$$(r_1 = 1 \texttt{ and } r_2 > 1)$$

by

$$\frac{(r_2 - 1)}{(r_1 - 1)} > \varepsilon,$$

for some $\varepsilon > 1$.

By Corollary 4.1 we know that the minimum ratio estimator $\widehat{B}^* = (\widehat{b}^*_{ij})_{d\times d}$ as defined in (4.8) converges almost surely to the true ML coefficient $b_{ij}$ for $i \in \mathrm{an}(j)$ and by Corollary 4.2 we know that it converges to zero for $i \in V \setminus \mathrm{An}(j)$ . Therefore, each $\widehat{b}^*_{ij}$ in the sum of Optimization Problem 3.22 converges almost surely and thus also in the recursive noise model it holds that

$$\lim_{n\to\infty} \mathbb{P}\Big( \arg\max_{\pi\in\Pi} \sum_{\pi(i)<\pi(j)} \widehat{b}^*_{ij} \in Q \Big) = 1.$$

## 4.2 Estimation of ML coefficients in the Hadamard noise model

In this section we define noise in the recursive ML model from Section 2.1 in a different way. The random vector $\widetilde{\boldsymbol{X}}$ is now given by

$$\widetilde{\boldsymbol{X}} = \begin{pmatrix} \widetilde{X}_1 \\ \vdots \\ \widetilde{X}_d \end{pmatrix} = \begin{pmatrix} X_1 \\ \vdots \\ X_d \end{pmatrix} \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_d \end{pmatrix} = \boldsymbol{X}\boldsymbol{\varepsilon}, \tag{4.9}$$

where $\boldsymbol{X} = (X_1, \ldots, X_d)$ is generated by (2.2) and the noise vector $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_d)$ with $(\varepsilon_i)_{i=1}^d$ being i.i.d., is multiplied componentwise to $\boldsymbol{X}$ (Hadamard product). Furthermore, we choose the distribution of $\varepsilon_i$ such that its support is a compact interval, i.e.,

$$\mathrm{supp}(\varepsilon_i) = [a, b], \quad 0 < a < b < \infty, \tag{4.10}$$

where $a$ and $b$ are unknown. Note, that the support of the ratio of two noise variables $\varepsilon_i, \varepsilon_j, i \neq j$ is then given by

$$\mathrm{supp}\Big(\frac{\varepsilon_j}{\varepsilon_i}\Big) = \big[\tfrac{a}{b}, \tfrac{b}{a}\big].$$

Thus, considering the ratio $\widetilde{X}_j/\widetilde{X}_i$ it holds with (3.4) that

$$\frac{\widetilde{X}_j}{\widetilde{X}_i} = \frac{X_j\,\varepsilon_j}{X_i\,\varepsilon_i} \geq b_{ij}\,\frac{a}{b} \quad i, j \in V,\ i \in \mathrm{an}(j),$$

what we will use to construct an estimate for the ML coefficient matrix $B$ in the following. The idea is to find a scaling factor, that estimates the value of the unknown quantity $b/a$ and then to multiply this scaling factor with the minimum ratio estimate $\widehat{b}_{ij}(a/b)$ to obtain an estimate for $b_{ij}$.

**Lemma 4.4.** *For all $i \in V$, let $X_i$ follow a recursive ML model and let the innovations $Z_k$, $k \in \mathrm{an}(i)$, be* Fréchet$(\nu > 0, s\!=\!1, m\!=\!0)$*-distributed, where $\nu$ is the shape parameter, $s$ the scale parameter and $m$ the location parameter.*
*Then, $X_i$ is* Fréchet$(\nu > 0, s\!=\!(\sum_{k\in\mathrm{An}(i)} b_{ki}^{\nu})^{1/\nu}, m\!=\!0)$*-distributed.*

*Proof.* Let $Z_k \sim$ Fréchet$(\nu > 0, s{=}1, m{=}0)$. Due to the independence of the innovations and with Definition 2.3 it holds for $x > 0$ that

$$\begin{aligned}
\mathbb{P}\big(X_i \le x\big) &= \mathbb{P}\big(\textstyle\bigvee_{k\in \mathrm{An}(i)} b_{ki} Z_k \le x\big) = \mathbb{P}\big(Z_k \le \frac{x}{b_{ki}} \text{ for all } k \in \mathrm{An}(i)\big) \\
&= \textstyle\prod_{k\in \mathrm{An}(i)} \mathbb{P}\big(Z_k \le \frac{x}{b_{ki}}\big) = \prod_{k\in \mathrm{An}(i)} \exp\big\{ - \big(\frac{x}{b_{ki}}\big)^{-\nu}\big\} \\
&= \exp\{-\big(\textstyle\sum_{k\in \mathrm{An}(i)} \big(\frac{b_{ki}}{x}\big)^{\nu}\big)\} = \exp\{-x^{-\nu} \sum_{k\in \mathrm{An}(i)} b_{ki}^{\nu}\} \\
&= \exp\Big\{ - \Big(\frac{x}{(\sum_{k\in \mathrm{An}(i)} b_{ki}^{\nu})^{1/\nu}}\Big)^{-\nu}\Big\},
\end{aligned}$$

which is the c.d.f. of a Fréchet$(\nu > 0, s{=}(\sum_{k\in \mathrm{An}(i)} b_{ki}^{\nu})^{1/\nu}, m{=}0)$-distribution. □

**Definition 4.5.** Let $X$ be a continuous random variable with non-negative support. We define the *geometric mean* of $X$ by

$$\mathrm{GM}(X) = \exp\{\mathbb{E}[\ln(X)]\}.$$

**Lemma 4.6.** *Let $X \sim$ Fréchet$(\nu > 0, s > 0, m = 0)$. Then,*

$$\mathrm{GM}(X) = s\ \exp\Big\{\frac{\gamma}{\nu}\Big\},$$

*where $\gamma = -\int_0^\infty \ln(t)\mathrm{e}^{-t}dt$ is the* Euler-Masceroni constant*.*

*Proof.* We use the substitution

$$t = \Big(\frac{x}{s}\Big)^{-\nu}, \quad \frac{dt}{dx} = -\frac{\nu}{s}\Big(\frac{x}{s}\Big)^{-\nu-1}, \quad x = st^{-\nu^{-1}}$$

and the identity

$$\gamma = -\int_0^\infty \ln(t)\mathrm{e}^{-t}dt.$$

Then, it holds that

$$\begin{aligned}
\mathbb{E}[\ln(X)] &= \int_{\mathbb{R}_+} \ln(x) dF(x) = \int_0^\infty \ln(x)\ \exp\{-\big(\tfrac{x}{s}\big)^{-\nu}\}\ \tfrac{\nu}{s}\ \big(\tfrac{x}{s}\big)^{-\nu-1}dx \\
&= -\int_\infty^0 \ln(st^{-\nu^{-1}})\mathrm{e}^{-t}dt = \int_0^\infty [\ln(s) - \ln(t^{\nu^{-1}})]\mathrm{e}^{-t}dt \\
&= \ln(s)\int_0^\infty \mathrm{e}^{-t}dt - \nu^{-1}\int_0^\infty \ln(t)\mathrm{e}^{-t}dt \\
&= \ln(s) + \frac{\gamma}{\nu}.
\end{aligned}$$

Applying the exponential function on both sides, we obtain the desired result. □

**Corollary 4.7.** *Let* $X_i \sim$ Fréchet$(\nu > 0, s_i = (\sum_{k\in\mathrm{An}(i)} b_{ki}^{\nu})^{1/\nu}, m = 0)$ *for all* $i \in V$. *Then, the geometric mean of* $X_i$ *is given by*

$$\mathrm{GM}(X_i) = \exp\Big\{\frac{\gamma}{\nu}\Big\}\Big(\sum_{k\in\mathrm{An}(i)} b_{ki}^{\nu}\Big)^{1/\nu},$$

*where* $\gamma$ *is the Euler-Masceroni-constant defined in Lemma 4.6.*

**Corollary 4.8.** *For all* $i \in V$, *let* $X_i \sim$ Fréchet$(\nu > 0, s_i = (\sum_{k\in\mathrm{An}(i)} b_{ki}^{\nu})^{1/\nu}, m = 0)$ *and assume that the i.i.d. noise variables* $(\varepsilon_i)_{i=1}^{d}$ *have geometric mean* $\mathrm{GM}(\varepsilon_i) = 1$. *Then the geometric mean of* $\widetilde{X}_i = X_i\varepsilon_i$ *is given by*

$$\mathrm{GM}(\widetilde{X}_i) = \exp\Big\{\frac{\gamma}{\nu}\Big\}\Big(\sum_{k\in\mathrm{An}(i)} b_{ki}^{\nu}\Big)^{1/\nu}.$$

*Proof.* For the geometric mean of $\widetilde{X}_i$ it holds by definition and with the linearity of the expectation that

$$\begin{aligned}\mathrm{GM}(\widetilde{X_i}) &= \exp\{\mathbb{E}[\ln(\widetilde{X_i})]\} = \exp\{\mathbb{E}[\ln(X_i) + \ln(\varepsilon_i)]\}\\ &= \exp\{\mathbb{E}[\ln(X_i)]\}\exp\{\mathbb{E}[\ln(\varepsilon_i)]\}\\ &= \mathrm{GM}(X_i)\mathrm{GM}(\varepsilon_i).\end{aligned}$$

If we assume that $\mathrm{GM}(\varepsilon_i) = 1$, we obtain the desired result with Corollary 4.7 . □

In Corollary 4.8 we assumed that the geometric mean of the i.i.d. noise variables is equal to one. This implies that

$$\mathbb{E}[\ln(\varepsilon_i)] = 0, \text{ for all } i \in V. \tag{4.11}$$

In the noise model $\widetilde{X}_i = X_i\varepsilon_i$ it is natural to assume that $\mathrm{GM}(\varepsilon_i) = 1$. In a regular additive noise model $X = Yc + \varepsilon$ we demand that $\mathbb{E}[\varepsilon] = 0$. Applying the exponential function on both sides, we obtain $e^X = e^{Yc}e^{\varepsilon}$ and it holds that

$$0 = \mathbb{E}[\varepsilon] = \mathbb{E}[\ln(e^{\varepsilon})],$$

which is equivalent to

$$e^{\mathbb{E}[\ln(e^{\varepsilon})]} = \mathrm{GM}(e^{\varepsilon}) = 1.$$

Hence, in the following we choose the distribution for the noise variables $\varepsilon_i, i \in V$, such that properties (4.10) and (4.11) are satisfied. In the simulation study in Chapter 5 we consider the truncated lognormal distribution. For this distribution we first choose the probability density function of a conventional lognormal distribution and specify the mean $\mu = 0$ and the standard deviation $\sigma > 0$. Moreover, we choose a truncation range, which is a compact interval $[a, b]$ with $0 < a < b < \infty$.
Afterwards, the probability density function corresponding to the conventional lognormal

distribution is adjusted by setting values outside the interval $[a, b]$ to zero. The values inside are uniformly scaled such that,

$$\int_a^b f(x)dx = 1.$$

Another distribution which satisfies properties (4.10) and (4.11) is the loguniform distribution, i.e. $\varepsilon_i = \exp\{U_i\}$, where $U_i \sim U([\ln(a), \ln(1/a)]), 0 < a < 1$. Then, the geometric mean of $\varepsilon_i$ is given by

$$\mathrm{GM}(\varepsilon_i) = \exp\{\mathbb{E}[\ln(\varepsilon_i)]\} = \exp\{\mathbb{E}[U_i]\} = \exp\{\tfrac{1}{2}(\ln(a) + \ln(\tfrac{1}{a}))\} = 1.$$

Under the assumptions of Corollary 4.8 and using its result, we define the *scale estimator* for $s_i = (\sum_{k \in \mathrm{An}(i)} b_{ki}^{\nu})^{1/\nu}$ by,

$$\widehat{s}_i(n) = \exp\Big\{-\frac{\gamma}{\nu}\Big\}\Big(\prod_{l=1}^{n} \widetilde{X}_l\Big)^{1/n}, \tag{4.12}$$

where $(\widetilde{X}_l)_{l=1}^n = (X_l \varepsilon_l)_{l=1}^n$ are i.i.d. copies of $\widetilde{X}_i$.

**Theorem 4.9.** *Let $\widehat{s}_i(n)$ be given as in (4.12). Then $\widehat{s}_i(n)$ converges almost surely to the true $s_i = (\sum_{k \in \mathrm{An}(i)} b_{ki}^{\nu})^{1/\nu}$, i.e. it holds for $n \to \infty$ that $\widehat{s}_i(n) \overset{\text{a.s.}}{\to} s_i$.*

*Proof.* By taking the logarithm on both sides of (4.12), we obtain with (4.9),

$$\ln(\widehat{s}_i(n)) = -\frac{\gamma}{\nu} + \frac{1}{n}\sum_{l=1}^{n} \ln(X_l) + \frac{1}{n}\sum_{l=1}^{n} \ln(\varepsilon_l).$$

Since $(X_l)_{l=1}^n$ and $(\varepsilon_l)_{l=1}^n$ are i.i.d. and independent sequences with $\mathbb{E}[|\ln(X_i)|] < \infty$ and $\mathbb{E}[|\ln(\varepsilon_i)|] < \infty$, the strong law of large numbers holds for $(\ln(X_l))_{l=1}^n$ and $(\ln(\varepsilon_l))_{l=1}^n$, i.e.,

$$\begin{aligned}\ln(\widehat{s}_i(n)) &= -\frac{\gamma}{\nu} + \frac{1}{n}\sum_{l=1}^{n} \ln(X_l) + \frac{1}{n}\sum_{l=1}^{n} \ln(\varepsilon_l)\\ &\overset{\text{a.s.}}{\to} -\frac{\gamma}{\nu} + \mathbb{E}[\ln(X_i)] + \mathbb{E}[\ln(\varepsilon_i)] = -\frac{\gamma}{\nu} + \Big(\ln(s_i) + \frac{\gamma}{\nu}\Big) + 0 = \ln(s_i),\end{aligned}$$

where we computed the expectation of $\ln(X_i)$ already in the proof of Lemma 4.6 and the expectation of $\ln(\varepsilon_i)$ is given by (4.11). Since $x \mapsto \exp\{x\}$ is continuous, we obtain

$$\widehat{s}_i(n) = \exp\{\ln(\widehat{s}_i(n))\} \overset{\text{a.s.}}{\to} \exp\{\ln(s_i)\} = s_i,$$

which is the desired result. □

As already mentioned above, we consider noise distributions such that the support of a noise variable $\varepsilon_i$ is given by $\mathrm{supp}(\varepsilon_i) = [a, b]$, $0 < a < b < \infty$, for all $i \in V$. With Lemma 3.2.(b) in Gissibl et al. [2018] we obtain,

$$\mathrm{supp}\Big(\frac{\widetilde{X}_j}{\widetilde{X}_i}\Big) = \mathrm{supp}\Big(\frac{X_j\,\varepsilon_j}{X_i\,\varepsilon_i}\Big) = [b_{ij}\,\frac{a}{b}, \infty) \quad \text{for } i \in \mathrm{an}(j).$$

Since we chose $0 < a < b < \infty$, it holds that $0 < (a/b) < 1$, and thus

$$b_{ij}\,\frac{a}{b} < b_{ij}. \tag{4.13}$$

This means, if we use the minimum ratio estimator

$$\widehat{b}^*_{ij} = \bigwedge_{s=1}^{n} \frac{\widetilde{X}^s_j}{\widetilde{X}^s_i} = \bigwedge_{s=1}^{n} \frac{X^s_j \varepsilon^s_j}{X^s_i \varepsilon^s_i}, \qquad i, j \in V, \tag{4.14}$$

we always underestimate the true ML coefficient $b_{ij}$ by the factor $a/b$. Therefore, the main idea of the new estimator is to scale up uniformly the minimum ratio estimate $\widehat{b}^*_{ij}$ by an appropriate *scaling factor* $\Delta > 1$, which is an estimate for the unknown value $b/a$.

**Corollary 4.10.** *The minimum ratio estimator $\widehat{b}^*_{ij}$ in the Hadamard noise model as defined in (4.14) converges almost surely to $b_{ij}(a/b)$ for $i \in \operatorname{an}(j)$ as $n \to \infty$.*

*Proof.* Follows from Lemma 3.6. □

## 4.2.1 Estimation of ML coefficients with known DAG

Let $\widetilde{\boldsymbol{X}} = (\widetilde{X}_1, \dots, \widetilde{X}_d)$ follow a recursive ML model on a known DAG $\mathcal{D}$ with noise as Hadamard product as defined in (4.9). Assume we have given an i.i.d. sample $\widetilde{\boldsymbol{X}}^1, \dots, \widetilde{\boldsymbol{X}}^n$. The initial point is the minimum ratio estimator $\widehat{B}^* = (\widehat{b}^*_{ij})_{d\times d}$ with entries defined by

$$\widehat{b}^*_{ij} = \bigwedge_{s=1}^{n} \frac{\widetilde{X}^s_j}{\widetilde{X}^s_i} = \bigwedge_{s=1}^{n} \frac{X^s_j \varepsilon^s_j}{X^s_i \varepsilon^s_i}, \qquad \text{for } i, j \in V. \tag{4.15}$$

The important step in the following algorithm is the computation of the scaling factor $\Delta_i$, $i \in V$. This scaling factor estimates the unknown value of $b/a$ and consequently is appropriate to adjust the minimum ratio estimate as defined in (4.15). Without adjustment, the minimum ratio estimate $\widehat{b}^*_{ij}$ underestimates the true $b_{ij}$ by $a/b$ as we have seen in (4.13).
In order to compute the scaling factor $\Delta_i$ we need the scale estimator $\widehat{s}_i(n)$ as defined in (4.12), since by Theorem 4.9 it converges almost surely to the true $(\sum_{k\in \operatorname{An}(i)} b^\nu_{ki})^{1/\nu}$ for $n \to \infty$. Bear in mind, that the scale parameter $\nu > 0$ is known. For the true shape parameter $s_i$ it holds by Lemma 4.4 that

$$s_i = \Big(\sum_{k\in \operatorname{An}(i)} b^\nu_{ki}\Big)^{1/\nu},$$

which is the same as

$$s^\nu_i = 1 + \sum_{k\in \operatorname{an}(i)} b^\nu_{ki}.$$

The 1 corresponds to the coefficient $\widehat{b}^*_{ii} = \bigwedge_{s=1}^n X_i^s\varepsilon_i^s / X_i^s\varepsilon_i^s = 1$ on the diagonal which we do not need to scale. For the scale estimator $\widehat{s}_i(n)$ as defined in (4.12) it holds that

$$(\widehat{s}_i(n))^\nu = \exp\{-\gamma\}\Big(\prod_{l=1}^n \widetilde{X}_l\Big)^{\nu/n}.$$

Therefore, it is natural to set

$$\Delta_i \sum_{k\in \mathrm{an}(i)} (\widehat{b}^*_{ki})^\nu + 1 \overset{!}{=} \exp\{-\gamma\}\Big(\prod_{l=1}^n \widetilde{X}_l\Big)^{\nu/n}, \tag{4.16}$$

and to solve this equation for $\Delta_i$.

**Algorithm 4.11** (Estimation of ML coefficients with noise as Hadamard product).

```
Input:  The minimum ratio estimator $\widehat{B}^* = (\widehat{b}^*_{ij})_{d\times d}$ as defined in (4.15).
Output: An estimate for the ML coefficient matrix $B = (b_{ij})_{d\times d}$.

    For all $i \in V$,

 1. compute $\widehat{s}_i(n)$ as defined in (4.12).

 2. If $\mathrm{an}(i) \neq \varnothing$,
        set $(\widehat{s}_i(n))^\nu \overset{!}{=} \Delta_i \sum_{k\in\mathrm{an}(i)} (\widehat{b}^*_{ki})^\nu + 1$ (cf. (4.16)) and solve for $\Delta_i$, i.e.,

            $\Delta_i = \frac{(\widehat{s}_i(n))^\nu - 1}{\sum_{k\in\mathrm{an}(i)} (\widehat{b}^*_{ki})^\nu}$,

        where $\Delta_i$ is the scaling factor for node $i$.
    Else,
        set $\Delta_i = 1$.

 3. Scale up the minimum ratio estimate $\widehat{b}^*_{ki}$ by the scaling factor $\Delta_i$ for all
    $k \in \mathrm{an}(i)$. The minimum ratio estimate $\widehat{b}^*_{ki}$ estimates $(a/b)b_{ki}$ and the scaling
    factor $\Delta_i$ estimates $b/a$. For all $k \in \mathrm{an}(i)$ set,

            $\breve{b}^{\mathtt{new}}_{ki} = \Delta_i \widehat{b}^*_{ki}$.

    End for-loop.

 4. For all $k \in V \setminus \mathrm{An}(i)$, set $\breve{b}^{\mathtt{new}}_{ki} = 0$.

 5. Proceed like in Section 3.3,
    compute $\breve{B}^{\mathtt{new}}_0 = (\breve{b}^{\mathtt{new}}_{ij} \mathbb{1}_{\mathrm{pa}(j)}(i))_{d\times d}$. Then, an estimate for $B$ is given by

            $\widehat{B}^{\mathtt{new}} = (I_d \vee \breve{B}^{\mathtt{new}}_0)^{\odot(d-1)}$.
```

### 4.2.2 Estimation of ML coefficients with known topol.

The starting point in this setting is the minimum ratio estimator $\widehat{B}^* = (\widehat{b}^*_{ij})_{d\times d}$ with entries as defined in (4.15).
Based on this matrix $\widehat{B}^*$ and the known topological order of the nodes, we infer the minimum ML DAG $\mathcal{D}^B$.
The explanations from the beginning in the first setting also apply here (cf. Subsection 4.2.1). The difference is, that we do not know the set of ancestors $\mathrm{an}(i)$ of node $i$ to compute the sum

$$\sum_{k\in \mathrm{an}(i)} (\widehat{b}^*_{ki})^\nu. \tag{4.17}$$

Therefore, we first estimate them by applying Algorithm 3.8 with the known topological order of the nodes as input. We obtain the set of estimated ancestors $\widehat{\mathrm{an}}(i)$ and compute the sum in (4.17) with this estimated set.

**Algorithm 4.12.**

```
Input:  The matrix $\widehat{B}^* = (\widehat{b}^*_{ij})_{d\times d}$ with entries as defined in $(4.15)$ and the
        known topological order of the nodes.
Output: An estimate for the ML coefficient matrix $B$ and for the edge-weight
        matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$.

  1. Using $\widehat{B}^* = (\widehat{b}^*_{ij})_{d\times d}$ and the known topological order as input, perform Al-
     gorithm 3.8. For each node $i \in V$, we obtain the set of estimated ancestors
     $\widehat{\mathrm{an}}(i)$ of $i$.

     For all $i \in V$,

  2. compute $\widehat{s}_i(n)$ as defined in $(4.12)$.

  3. If $\widehat{\mathrm{an}}(i) \neq \varnothing$,

        set $(\widehat{s}_i(n))^\nu \overset{!}{=} \Delta_i \sum_{k\in\widehat{\mathrm{an}}(i)} (\widehat{b}^*_{ki})^\nu + 1$ (cf. $(4.16)$) and solve for $\Delta_i$, i.e.,

            $\Delta_i = \frac{(\widehat{s}_i(n))^\nu - 1}{\sum_{k\in\widehat{\mathrm{an}}(i)} (\widehat{b}^*_{ki})^\nu}$,

        where $\Delta_i$ is the scaling factor for node $i$.

     Else,

        set $\Delta_i = 1$.

  4. Scale up the minimum ratio estimate $\widehat{b}^*_{ki}$ by the scaling factor $\Delta_i$ for
     all $k \in \mathrm{an}(i)$. Here, $\widehat{b}^*_{ki}$ is the minimum ratio estimate of $b_{ij}$ after the
     application of Algorithm 3.8. The minimum ratio estimate $\widehat{b}^*_{ki}$ estimates
     $(a/b)b_{ki}$ and the scaling factor $\Delta_i$ estimates $b/a$. For all $k \in \widehat{\mathrm{an}}(i)$ set,

            $\breve{b}^{\mathrm{new}}_{ki} = \Delta_i \widehat{b}^*_{ki}$.
```

```
    End for-loop.
5. Using the estimate $\widehat{B}^{\text{new}} = (\widehat{b}_{ij}^{\text{new}})_{d\times d}$ for the ML coefficients $B$ as input,
   proceed with Algorithm 4.3, in order to infer which edges in the minimum
   ML DAG $\mathcal{D}^B$ exist.
```

### 4.2.3 Estimation of the topol. order

Initial point in this setting is the matrix $\widehat{B}^* = (\widehat{b}_{ij}^*)_{d\times d}$ with entries as defined in (4.15). We infer the topological order of the nodes of the unknown DAG by applying the Branch & Bound algorithm (composition of Algorithms 3.18, 3.20, 3.21, 3.25 and 3.28) from Chapter 3. However, we have to adjust Algorithm 3.18 a little bit.

**Lemma 4.13.** *Let $\widetilde{Y}_{ij} = \widetilde{X}_i/\widetilde{X}_j = X_i\varepsilon_i/X_j\varepsilon_j$. Then it holds that*

$$\widetilde{Y}_{ij}^{(1)}/\widetilde{Y}_{ij}^{(2)} \overset{\text{a.s.}}{\to} 1$$

*if and only if $b_{ij} > 0$ for $n \to \infty$.*

*Proof.* First, let $b_{ij} > 0$, i.e. $i \in \text{an}(j)$. Then it holds that

$$\widetilde{Y}_{ij} \geq b_{ij}\frac{a}{b}$$

and by Corollary 4.10 also that $\widetilde{Y}_{ij}^{(1)}, \widetilde{Y}_{ij}^{(2)} \overset{\text{a.s.}}{\to} b_{ij}\frac{a}{b}$, as $n \to \infty$. Hence, for $n \to \infty$ it holds that

$$\widetilde{Y}_{ij}^{(1)}/\widetilde{Y}_{ij}^{(2)} \overset{\text{a.s.}}{\to} 1.$$

For the other direction we show that the c.d.f. of $\widetilde{Y}_{ij}$ goes to zero polynomial. First, it holds that

$$\mathbb{P}(\widetilde{X}_j/\widetilde{X}_i \leq x) \leq \mathbb{P}(X_j/X_i \leq x\tfrac{b}{a}) \quad \text{and}$$
$$\mathbb{P}(\widetilde{X}_j/\widetilde{X}_i \leq x) \geq \mathbb{P}(X_j/X_i \leq x\tfrac{a}{b}).$$

With these two inequalities we obtain an upper bound,

$$\begin{aligned}\frac{F(tx)}{F(t)} &= \frac{\mathbb{P}(\widetilde{X}_j/\widetilde{X}_i \leq tx)}{\mathbb{P}(\widetilde{X}_j/\widetilde{X}_i \leq t)} \leq \frac{\mathbb{P}(X_j/X_i \leq t\frac{a}{b}\frac{b}{a}\frac{b}{a}x)}{\mathbb{P}(X_j/X_i \leq t\frac{a}{b})} \\ &= \frac{\mathbb{P}(X_j/X_i \leq \tilde{t}\ \tilde{x}\ )}{\mathbb{P}(X_j/X_i \leq \tilde{t}\ )},\end{aligned} \tag{4.18}$$

where $\tilde{t} = t\frac{a}{b}$ and $\tilde{x} = \frac{b}{a}\frac{b}{a}x$.
For $\tilde{t} \downarrow 0$, and thus for $t \downarrow 0$, we obtain,

$$\lim_{t\downarrow 0}\frac{F(tx)}{F(t)} \leq \tilde{x}^{\alpha}\big(\tfrac{b^2}{a^2}\big)^{\alpha} \text{ for some } \alpha > 0.$$

Analogously to (4.18), we state a lower bound,

$$\lim_{t\downarrow 0} \frac{F(tx)}{F(t)} \geq \tilde{x}^{\alpha}\left(\tfrac{a^2}{b^2}\right)^{\alpha} \text{ for some } \alpha > 0.$$

Since $a$ and $b$ are real constants, we obtain,

$$\lim_{t\downarrow 0} \frac{F(tx)}{F(t)} \sim x^{\alpha},$$

for some $\alpha > 0$. Now we use the same arguments as in Lemma 3.16 to finish the proof. □

We have to adjust Algorithm 3.18, since we do not observe any atoms in the noise model. In the if-condition in step 4. we replace the condition

$$(r_1 = 1 \texttt{ and } r_2 > 1)$$

by

$$\frac{(r_2 - 1)}{(r_1 - 1)} > \varepsilon,$$

for some $\varepsilon > 0$.

Using the estimated topological order of the nodes and the modified minimum ratio estimator, we proceed with Algorithm 4.12 in order to the minimum ML DAG $\mathcal{D}^B$.

# Chapter 5

# Simulation Study

This chapter is dedicated to experimental studies on all algorithms and methods which we presented in Chapters 3 and 4 in order to assess practical performances and the power and restrictions of the methods. To do so, we are going to use simulated data.
The running times of the different algorithms is one important subject of investigation. Furthermore, we investigate how accurate the Branch & Bound algorithm (composition of Algorithms 3.18, 3.20, 3.21, 3.25 and 3.28) infers the unknown topological order compared to the Greedy algorithm (Algorithm 3.12) and how accurate we estimate the ML coefficients and the minimum ML DAG $\mathcal{D}^B$ knowing only the topological order of the DAG.
All these experiments are first performed in the non-noisy model and secondly in both noise models. All simulations are done with the programming language Python.

## 5.1 Randomly generated DAGs

We generate a DAG in the following way.

Step 1) Simulate a topological order $\pi$ by drawing integers from 1 to $d$ without replacement.

Step 2) The maximal number of connecting edges between pairs of distinct nodes in a DAG is $d(d-1)/2$ (if we set more than $d(d-1)/2$ edges, we directly obtain cycles respectively undirected edges). Thus, we simulate an adjacency matrix in upper triangular form (cf. Remark 2.6).
For each pair of distinct nodes make a random choice whether or not to connect them by an edge. The choices are made independently from each other. We denote by $0 < p < 1$ the success probability for connecting two distinct nodes by an edge. Then, this choice-process is Bernoulli($p$)-distributed. This procedure was introduced by Erdös and Rényi [1960].

Step 3) According to the topological order $\pi$ generated in Step 1, reorder the adjacency matrix.

Step 4) Assign a weight to each edge. The weights are uniformly distributed on an interval $\mathcal{I} \subset (0, \infty)$.

In order to compute the true underlying ML coefficient matrix $B$ corresponding to the simulated DAG $\mathcal{D}$, we perform Algorithm 2.7. For the purpose of generating synthetic data corresponding to a recursive ML model on a DAG $\mathcal{D}$ we make use of Definition 2.2. All these steps are performed for all simulations that follow.

Depending on the specific algorithm there is a different number of parameters that have to be tuned. However, for all algorithms there are some parameters that are always relevant. That is,

- the sample size $n$,
- the number of nodes $d$,
- the interval $\mathcal{I} \in (0, \infty)$ from which the edge-weights are drawn uniformly,
- the probability $0 < p < 1$ for setting an edge in the true DAG,
- the number of replications $m$, and
- the distribution of the i.i.d. innovations $(Z_i)_{i=1}^d$.

In all simulations we choose the interval $\mathcal{I} = [0, 1]$ from which the edge-weights are drawn uniformly and we fix the distribution of the i.i.d. innovations $(Z_i)_{i=1}^d$ to be standard Fréchet. We differ the number of nodes $d$, the probability for setting an edge $p$ and the sample size $n$. Unless stated otherwise, we fix the number of replications to $m = 300$ and average the results over these replicates.

Concerning the proportion between the sample size and the number of nodes, we always choose the sample size larger than the number of nodes. Otherwise, there is a dimension flaw (cf. Champion et al. [2017], Section 1).

## 5.2 Key performance measures

We use different performance measures to assess the accuracy of the estimates.

- In the first setting, where we assume to know the ancestral relations of the underlying DAG, we first compare the *maximum measurement error* MME between the estimated ML coefficients $\widehat{B}^{(k)} = (\widehat{b}_{ij}^{(k)})_{d\times d}$ in replication $k \in \{1, \dots, m\}$ and the true ML coefficients $B^{(k)} = (b_{ij}^{(k)})_{d\times d}$ averaged over all $m$ replications,

$$\text{MME} = \frac{1}{m}\sum_{k=1}^{m} \max_{i,j\in V} \big|\widehat{b}_{ij}^{(k)} - b_{ij}^{(k)}\big|. \tag{5.1}$$

  Secondly, we consider the *average measurement error* AME between $\widehat{B}^{(k)}$ and $B^{(k)}$, i.e.,

$$\text{AME} = \frac{1}{m}\sum_{k=1}^{m} \frac{1}{|S|} \sum_{\substack{i,j\in V:\\ i\neq j,\\ b_{ij}^{(k)}>0}} \big|\widehat{b}_{ij}^{(k)} - b_{ij}^{(k)}\big|, \tag{5.2}$$

where $S = \{b_{ij}^{(k)} : b_{ij}^{(k)} > 0 \text{ and } i \neq j\}$.

- In the second setting, where we assume to know the topological order of the underlying DAG, and where we estimate ML coefficients and infer the minimum ML DAG $\mathcal{D}^B$, we use the *Hamming distance* for graphs to assess the accuracy of recovery. The Hamming distance was first proposed by Richard Hamming in 1950 (cf., Hamming [1950]) in the field of coding theory. In this field, it measures the minimum number of substitutions required to change one string into the other, where the two strings are of equal length.
  Later on, people also used the Hamming distance as a measure of graph recovery: in Bu and Lederer [2017], Subsection 2.2, it is defined as follows. Let $\widehat{E}$ denote the estimated edge set and $E$ the true edge set. Then, the Hamming distance is defined by
  $$\mathrm{d_H}(\widehat{E}, E) = |\{(i,j) : (i,j) \in \widehat{E}, (i,j) \notin E\} \cup \{(i,j) : (i,j) \notin \widehat{E}, (i,j) \in E\}|,$$
  i.e. it is the sum of edges that were inferred by mistake (*false positive error*) and of edges that were not recognized, although there is an edge in the true graph (*false negative error*). The lower the Hamming distance, the better the graph recovery. In order to be comparable, we define the normed Hamming distance, i.e.,
  $$\overline{\mathrm{d_H}}(\widehat{E}, E) = \frac{\mathrm{d_H}(\widehat{E}, E)}{d(d-1)/2}, \tag{5.3}$$
  where $d$ denotes the number of nodes in the DAG. There are at most $d(d-1)/2$ edges in a DAG. Observe that we generally cannot identify edges of the true underlying DAG $\mathcal{D}$ and therefore we only consider the edges in the minimum ML DAG $\mathcal{D}^B$.
  We use this measure to assess the accuracy of the recovery of ML coefficients as well. For this purpose, we substitute the set of inferred edges $\widehat{E}$ by the set of inferred paths $\widehat{P}$ and we substitute the set of existing edges $E$ by the set of existing paths $P$. Identifying ML coefficients is equivalent to inferring the reachability matrix $R = \mathrm{sgn}(B)$.

  For the sake of completeness, we say that an inferred edge or path is a *true positive edge/path*, if in the true underlying minimum ML DAG $\mathcal{D}^B$ or in the true DAG $\mathcal{D}$ there is an edge or a path and it is correctly inferred. Moreover, we say that an inferred edge or path is a *true negative edge/path*, if in the true underlying minimum ML DAG $\mathcal{D}^B$ or in the true DAG $\mathcal{D}$ there is no edge or no path and it is correctly *not* inferred.

- In the third setting, where we want to infer a topological order of the nodes of the unknown DAG, we first compare the *overall success rate*. That means, if the estimated topological order $\widehat{\pi}_k$, in replication $k \in \{1, \dots, m\}$, belongs to the equivalence class $Q$ induced by the true underlying DAG $\mathcal{D}$ (cf. Section 3.2), then we mark it as success. If it does not belong to this equivalence class, we mark it as failure. Replicating this procedure for $k = 1, \dots, m$, we obtain the overall success rate OSR defined by

$$\mathrm{OSR} = \frac{1}{m}\sum_{k=1}^{m} \mathbb{1}_{\{\widehat{\pi}_k \in Q\}}, \tag{5.4}$$

where $\mathbb{1}$ denotes the indicator function. This overall success rate can also be interpreted as the empirical probability for inferring a topological order $\widehat{\pi} \in Q$.

Secondly, when we infer a wrong topological order, we would like to assess how good or bad this estimate still is. Therefore, we introduce the *ancestor success rate* ASR defined by

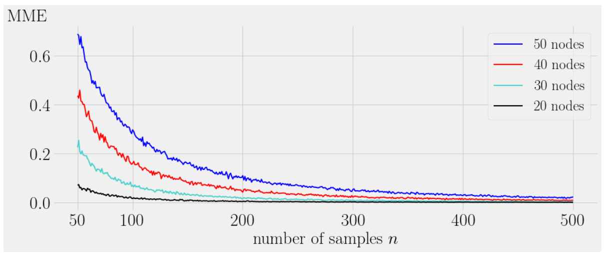
$$\mathrm{ASR} = \frac{1}{m}\sum_{k=1}^{m} \frac{\#\{\text{correctly predicted ancestral relations in } \widehat{\pi}_k\}}{\#\{\text{existing ancestral relations in total in } \pi_k\}}. \tag{5.5}$$

## 5.3 Experimental results

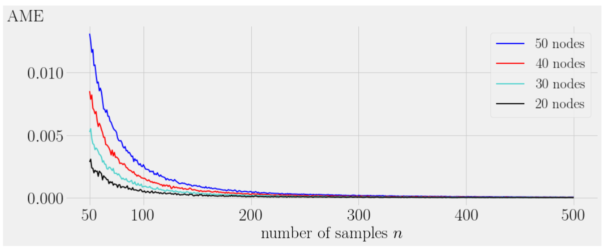
### 5.3.1 No-noise model

**First setting: GMLE estimate of $B$**

We start with simulations on the GMLE for ML coefficients on randomly generated DAGs of different sizes (cf. Section 3.3). We observed, that the choice of the probability $p$ for setting an edge has only limited effect on the plots in this setting. Therefore, we fix it to $p = 0.2$. The only difference is that the higher the probability $p$, the higher the measurement errors. With increasing $p$ the number of paths in $\mathcal{D}$ increases and therefore the probability for making errors increases as well, since a node $j$ has more ancestors and therefore the probability $\mathbb{P}(X_j/X_i = b_{ij})$ decreases for $i \in \mathrm{an}(j)$.



(a) MME as in (5.1)



(b) AME as in (5.2)

Figure 5.1: We observe the exponential decay of the MME and the AME with increasing sample size (cf. Proposition 4.5 in Gissibl et al. [2018]) for different sizes of randomly generated DAGs.

In Figure 5.1 we see the exponential rate of convergence of the MME and the AME. Furthermore, we see in Figure 5.1(b) that already at a sample size of $n = 300$ for all numbers of nodes convergence is reached.

**Second setting: estimate ML coefficients and find the minimum ML DAG $\mathcal{D}^B$**

In this setting we apply Algorithm 3.8 and Algorithm 3.9 with a simulated topological order as input. We compute the normed Hamming-distance as defined in (5.3) to assess the goodness of the recovery of ML coefficients and of edges in the minimum ML DAG $\mathcal{D}^B$.
For the ML coefficient matrix $B$ we substitute the set of inferred edges $\widehat{E}$ by the set of inferred paths $\widehat{P}$ and the set of existing edges $E$ by the set of existing paths $P$. We use the edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$ as the set of existing edges $E$, since we generally cannot identify all edges of the true underlying DAG $\mathcal{D}$.
We first plot the normed Hamming distance on heatmaps for different probabilities $p = 0.60, 0.50, 0.40, 0.30$ for setting an edge in the true DAG and for all sample sizes between $n = 50$ and $n = 200$. We do this for various number of nodes.
Secondly, we state the absolute number of false negative errors and false positive errors (cf. Section 5.2) for selected sample sizes and selected number of nodes.

Let us consider the heatmaps below. As we expected, the larger the sample size $n$, the better the recovery of ML coefficients and of edges in the minimum ML DAG $\mathcal{D}^B$.
For all probabilities for setting an edge $p$, the normed Hamming distance is in a range between 0 and 0.25, i.e. a maximum of only 25% of ML coefficients or edges in $\mathcal{D}^B$ have been inferred wrong, what confirms the good performance of both algorithms (with sample sizes in a range between $n = 50$ and $n = 200$).
Furthermore, we see that the larger the probability $p$ for setting an edge in the true DAG $\mathcal{D}$, the better the recovery of ML coefficients and the better the recovery of edges in the true minimum ML DAG $\mathcal{D}^B$, respectively.

If we fix the number of nodes, with increasing $p$ the average number of incoming paths into a node $j$ increases, too. Thus, the value of the random variable $X_j$ is large and the minimum of the ratio $X_j/X_i$, $i \in \mathrm{an}(j)$, is large, as well. Therefore, it is easier to identify the path from node $i$ to node $j$ with increasing $p$.
The recovery of edges in the minimum ML DAG $\mathcal{D}^B$ becomes better with increasing $p$, too. If $p$ is small, the probability for observing $X_j = X_i b_{ij}$, $i \in \mathrm{pa}(j)$, is large. Hence, we make less false negative errors, but more false positive errors (i.e. we make more errors where in $C^B$ is no edge, but we inferred an edge in $\widehat{C^B}$). The threshold $\varepsilon = \frac{k}{n}$ (cf. (3.13)) is the asymptotic behavior, but it is unreliable for small sample sizes. For larger $p$ there are fewer nodes that are not connected by an edge and therefore there are less possibilities to make a false positive error.

If we fix the probability $p$, we observe in all plots that the higher the number of nodes $d$, the better the recovery of ML coefficients. At the same time, the edge recovery in the minimum ML DAG $\mathcal{D}^B$ is less accurate with increasing number of nodes.
For the recovery of ML coefficients it holds again that, if the number of nodes increases with fixed $p$, then the average number of incoming paths into a node $j$ increases, too. Thus, the value of the random variable $X_j$ is large and therefore the minimum of the ratio $X_j/X_i, i \in \mathrm{an}(j)$, is large and it is easy to identify the path from node $i$ to node $j$.
If the number of nodes increases, the amount of critical paths to some node $j$ increases

as well. Therefore, for some specific critical path $i \to k \to j$ it is more difficult to observe the equality $\widehat{b}_{ij} = \widehat{b}_{ik}\widehat{b}_{kj}$, if we keep the sample size $n$ fixed. Thus, we less often delete entries and we make more false positive errors.

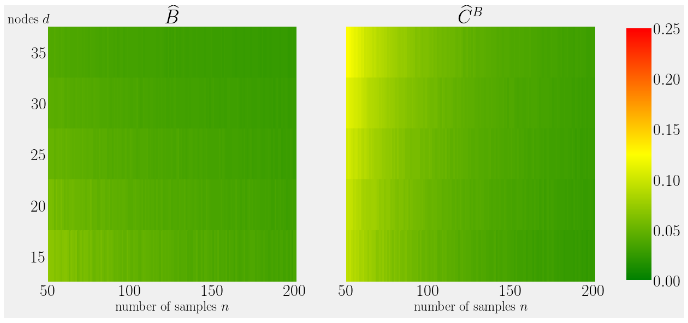


Figure 5.2: Probability for setting an edge is $p = 0.6$. Very good recovery of ML coefficients on the left and very good recovery of edges in the minimum ML DAG $\mathcal{D}^B$ on the right.

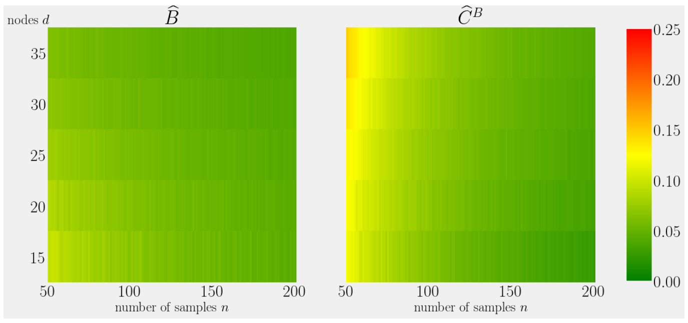


Figure 5.3: Probability for setting an edge is $p = 0.5$. Still very good recovery recovery of ML coefficients on the left and very good recovery of edges in the minimum ML DAG $\mathcal{D}^B$ on the right.

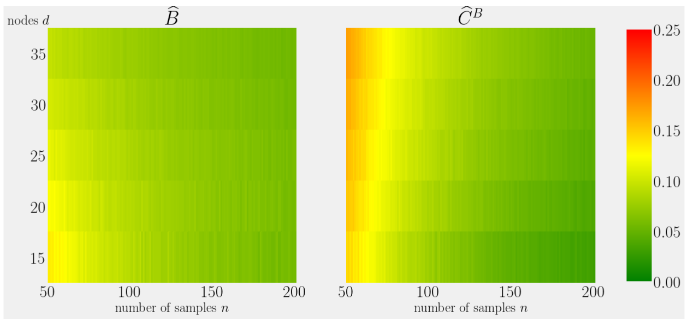


Figure 5.4: Probability for setting an edge is $p = 0.4$. Good recovery of ML coefficients on the left and medium to good recovery of edges in the minimum ML DAG $\mathcal{D}^B$ on the right.

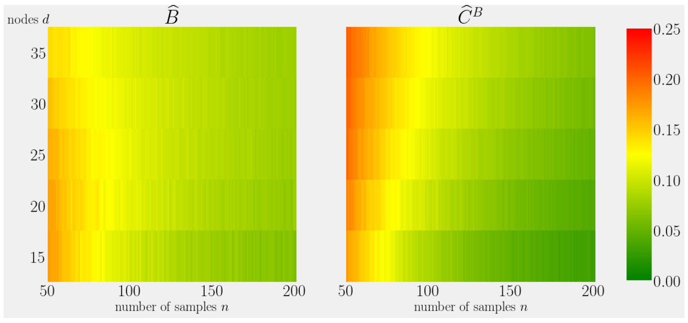


Figure 5.5: Probability for setting an edge is $p = 0.3$. Good to medium recovery of ML coefficients on the left and medium recovery of edges in the minimum ML DAG $\mathcal{D}^B$ on the right. However, the overall amplitude is between 0.0% and 25.0% which is still good.

In Table 5.6 we compare the number of false negative errors with the number of false positive errors in the estimated ML coefficient matrix $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$ after application of Algorithm 3.8. We observe that the number of false positive errors (inferred by mistake) is larger than then the number of false negative errors (path not recognized). Due to the massive inequality between false negative and false positive errors we can conclude that we chose the threshold parameter $c(n)$ too small for a limited sample size. Whether we obtain more false positive errors than false negative errors or the other way around, highly depends on this threshold parameter $c(n)$. If we choose a larger threshold, we obtain more false negative errors, because we delete more entries.

| | True pos. | True neg. | False neg. | False pos. | Sum of false pos. and neg. |
|---|---|---|---|---|---|
| $p = 0.30$ | | | | | |
| 20 nodes | | | | | |
| $n = 50$ | 114 (60.0%) | 48 (25.3%) | 8 (4.2%) | 20 (10.5%) | 28 (14.7%) |
| 125 | 117 (61.6%) | 49 (25.8%) | 4 (2.1%) | 20 (10.5%) | 24 (12.6%) |
| 200 | 118 (62.1%) | 50 (26.3%) | 2 (1.1%) | 20 (10.5%) | 22 (11.6%) |
| 30 nodes | | | | | |
| $n = 50$ | 303 (69.7%) | 72 (16.6%) | 14 (3.2%) | 46 (10.6%) | 60 (13.8%) |
| 125 | 310 (71.3%) | 71 (16.3%) | 6 (1.4%) | 48 (11.0%) | 54 (12.4%) |
| 200 | 313 (72.0%) | 69 (15.9%) | 5 (1.1%) | 48 (11.0%) | 53 (12.1%) |

Table 5.6: Recovery of ML coefficients over $m = 300$ replications after application of Algorithm 3.8. In this example, the number of false positive errors (inferred by mistake) is larger than the number of false negative errors (path not recognized). The reason is that we chose a small threshold parameter $c(n)$ in Algorithm 3.8.

In Table 5.7 we compare the number of false negative errors with the number of false positive errors in the estimated edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$ after application of Algorithm 3.8 and Algorithm 3.9. Again, the number of false positive errors is larger than the number of false negative errors. This is connected to the too small threshold parameter $c(n)$ which we chose in Algorithm 3.8 to estimate $\widehat{B} = (\widehat{b}_{ij})_{d\times d}$, since we estimate the edges in $\mathcal{D}^B$ based on $\widehat{B}$.

| | True pos. | True neg. | False neg. | False pos. | Sum of false pos. and neg. |
|---|---|---|---|---|---|
| $p = 0.30$ | | | | | |
| 20 nodes | | | | | |
| $n = 50$ | 47 (24.7%) | 115 (60.5%) | 2(1.1%) | 26 (13.7%) | 28 (14.8%) |
| 200 | 48 (25.3%) | 119 (62.6%) | 1 (0.5%) | 22 (11.6%) | 23 (12.1%) |
| 500 | 49 (25.8%) | 120 (63.2%) | 0 (0.0%) | 21 (11.1%) | 21 (11.1%) |
| 30 nodes | | | | | |
| $n = 50$ | 99 (22.8%) | 262 (60.2%) | 4 (1.0%) | 70 (16.1%) | 74 (17.1%) |
| 200 | 103 (23.7%) | 277 (63.7%) | 1 (0.2%) | 54 (12.4%) | 55 (12.6%) |
| 500 | 102 (23.5%) | 283 (65.1%) | 1 (0.2%) | 49 (11.3%) | 50 (11.5%) |

Table 5.7: Recovery of edges in the minimum ML DAG $\mathcal{D}^B$ averaged over $m = 300$ replications after application of Algorithms 3.8 and 3.9.

**Third setting: infer the topological order of the nodes of an unknown DAG**

In this setting we want to assess the goodness of

(i) the Branch & Bound algorithm (composition of Algorithms 3.18, 3.20, 3.21, 3.25 and 3.28),

(ii) the Greedy algorithm (Algorithm 3.12),

(iii) the Dynamic Programming method (Algorithm 3.30), and

(iv) the Brute-Force algorithm (Algorithm 3.24),

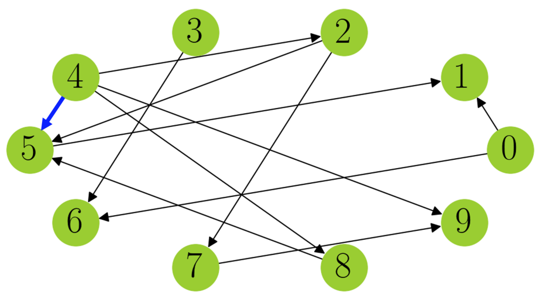
with respect to success probabilities introduced in Section 5.2 and with respect to computational time.

In a first experiment we give an overview of how we proceed, if we have only a (simulated) data set at hand and do not know anything about the structure of the DAG. We do this in order to explain the mechanisms of starting in setting 3 and inferring a topological order and then going backward to setting 2 to estimate the minimum ML DAG $\mathcal{D}^B$.
The first aim is to infer a topological order of the nodes with the Branch & Bound algorithm and subsequently we use this estimated topological order and apply Algorithm 3.8 to estimate the ML coefficient matrix $B$ and then Algorithm 3.9 to find the minimum ML DAG $\mathcal{D}^B$. Since the edges in $\mathcal{D}$ are not identifiable, we can only estimate the ML coefficients $b_{ij}$ and the edges $c_{ij}^B$ in $\mathcal{D}^B$. Since $\mathcal{D}^B$ leads to the same distribution $\mathcal{L}(\boldsymbol{X})$ as $\mathcal{D}$ does, this is enough.

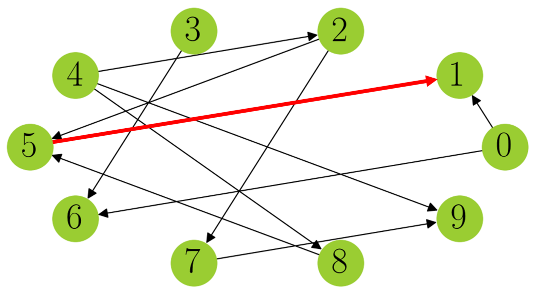
We set the following parameters, which seem to be reasonable in a real-world network:

$$d = 10, \qquad n = 100, \qquad p = 0.3.$$

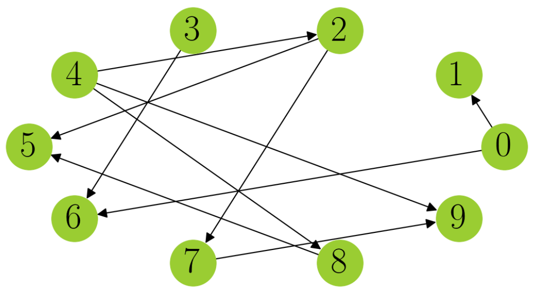
The true underlying DAG $\mathcal{D}$ was simulated with the procedure as described in Section 5.1 and the true minimum ML DAG is computed with Algorithm A.1. We obtain the following plots for the DAGs:



True DAG $\mathcal{D}$ with topological order $\pi = (4, 2, 7, 0, 8, 3, 6, 5, 9, 1)$. The edge marked in blue is not critical.



(a) True minimum ML DAG $\mathcal{D}^B$ with the same topological order $\pi$ as above. The edge marked in red was not inferred in $\widehat{\mathcal{D}}^{\widehat{B}}$ on the right.



(b) Estimated minimum ML DAG $\widehat{\mathcal{D}}^{\widehat{B}}$ with estimated topological order $\widehat{\pi} = (4, 0, 3, 8, 6, 2, 7, 9, 5, 1)$.

The edge $(4, 5)$ marked in blue in the true DAG $\mathcal{D}$ is not critical and therefore does not occur in the true minimum ML DAG $\mathcal{D}^B$. The edge $(5, 1)$ marked in red in the true minimum ML DAG $\mathcal{D}^B$ was not inferred in the estimated minimum ML DAG $\widehat{\mathcal{D}}^{\widehat{B}}$ in plot (b). It is the only (false negative) error.
Let $Q$ denote the equivalence class induced by the true underlying DAG $\mathcal{D}$. With Algorithm A.2 we deduce that $\widehat{\pi} \in Q$.
Furthermore, we obtain a normed Hamming distance for paths equal to

$$\overline{\mathrm{d_H}}(\widehat{P}, P) = 4/45 \approx 0.0888,$$

i.e. approximately 8.9% of the ML coefficients were inferred wrongly. These are the paths $5 \to 1$, $2 \to 1$, $4 \to 1$ and $8 \to 1$.

Now we compare the Branch & Bound algorithm (composition of Algorithms 3.18, 3.20, 3.21, 3.25 and 3.28) with the Greedy algorithm (Algorithm 3.12). In Figure 5.8 we observe the average time measurement in seconds for one replication with a sample size of $n = 500$ and a probability for setting an edge of $p = 0.4$ on a logarithmic timeline.
Although it is complex to compute the sum in Optimization Problem 3.22 (problem is NP-hard, cf. Lemma 3.23), we see that the running time is reasonably low even for a large number of nodes. In fact, it is no problem to use the Branch & Bound algorithm (and also the Greedy algorithm) even for $d = 200$ nodes (cf. Table 5.11). The growth of the time of one replication in both algorithms for increasing number of nodes is not linear on the logarithmic timeline, therefore it is not exponential on the standard timeline.

In Figure 5.9 we compare the overall success rate (OSR) of the Branch & Bound algorithm with that of the Greedy algorithm. We observe that the Branch & Bound algorithm has a better or equal OSR for all number of nodes, since we observe atoms and in particular Algorithm 3.18 performs very well. It confirms the theory, that maximizing the sum in Optimization Problem 3.22 is more robust than to solve Optimization Problem 3.11. As a trivial observation, the OSR for both algorithms decreases with increasing number of

nodes.
If we chose a larger probability $p$ for setting an edge in the true DAG than 0.4, the OSR as well as the ASR would be shifted downwards for all number of nodes $d$. The reason is that with increasing $p$ we have an increasing number of ancestral relations and it becomes more difficult to estimate the correct ancestral relations or the correct topological order, respectively. Furthermore, we observe that, as the number of nodes increases, the Branch & Bound algorithm becomes relatively better than the Greedy algorithm. That means, in particular for large dimensions, we should always prefer the Branch & Bound algorithm to the Greedy algorithm, although the Greedy algorithm is faster.

In Figure 5.10 we compare the ancestor success rate (ASR) of the Branch & Bound algorithm with that of the Greedy algorithm. We observe that the Branch & Bound algorithm has a better or equal ASR for all number of nodes. In the beginning, the ASR has a strong increase for DAGs with 4,5,6 and 7 nodes, but afterwards it decreases slowly with increasing number of nodes. The reason is that in the ASR as defined in (5.5), the denominator becomes larger than the nominator with increasing number of nodes and it even approaches the value $d(d-1)/2$, i.e. the number of paths in a complete DAG.

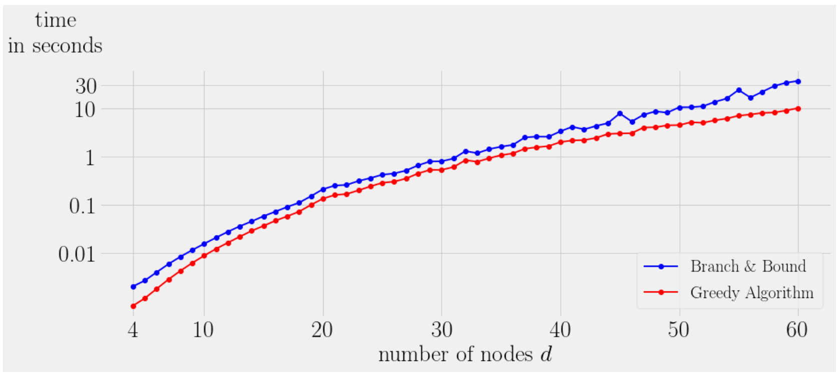


Figure 5.8: Time measurement in seconds on a logarithmic timeline for one replication of the Greedy algorithm compared to the Branch & Bound algorithm. We fix the parameters to $n = 500$ and $p = 0.4$.

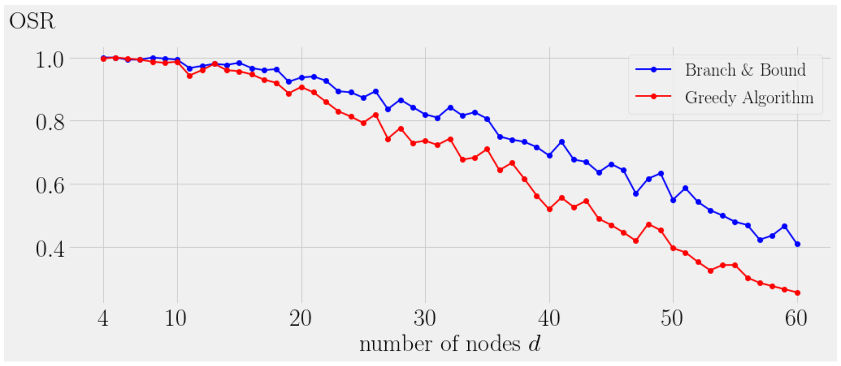


Figure 5.9: OSR as defined in (5.4) for the non-noisy model. We fix the parameters to $n = 500$ and $p = 0.4$. For increasing number of nodes the difference of the OSR becomes larger in favor of the Branch & Bound algorithm.

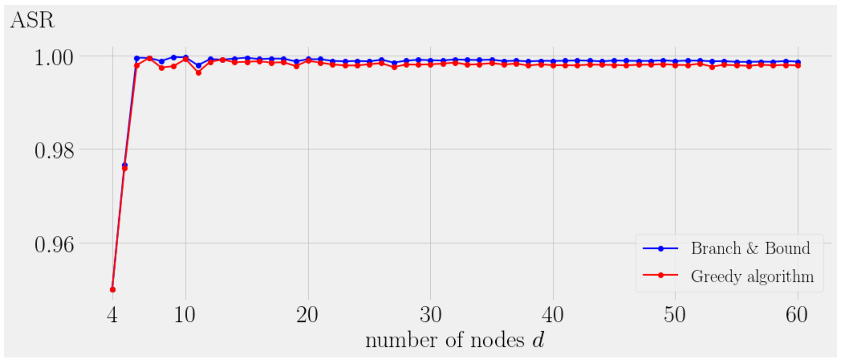


Figure 5.10: ASR as defined in (5.5) for the non-noisy model. We fix the parameters to $n = 500$ and $p = 0.4$.

| number of nodes $d$ | Branch & Bound (in minutes) | Greedy (in minutes) |
|---|---|---|
| 100 | 2.3 | 1.0 |
| 120 | 4.8 | 2.1 |
| 150 | 12.5 | 5.3 |
| 170 | 22.1 | 8.8 |
| 200 | 39.7 | 17.3 |

Table 5.11: Time measurement of one replication of the Branch & Bound algorithm compared to Greedy algorithm for large number of nodes. Probability for setting an edge is $p = 0.4$ and sample size is $n = 500$. Time is stated in minutes.

In the following plot we compare the average time measurement of one replication of the four methods which we introduced in Section 3.5 on a logarithmic timeline. It is only possible to compare them up to $d = 10$ nodes, since the Brute-Force method has a complexity of $\mathcal{O}(d!)$. Also the Dynamic Programming method with a complexity of $\mathcal{O}(d^2 2^d)$ is only applicable up to $d = 15$ nodes.

For the plot below we chose a sample size of $n = 50$ and a probability for setting an edge in the true underlying graph of $p = 0.4$.

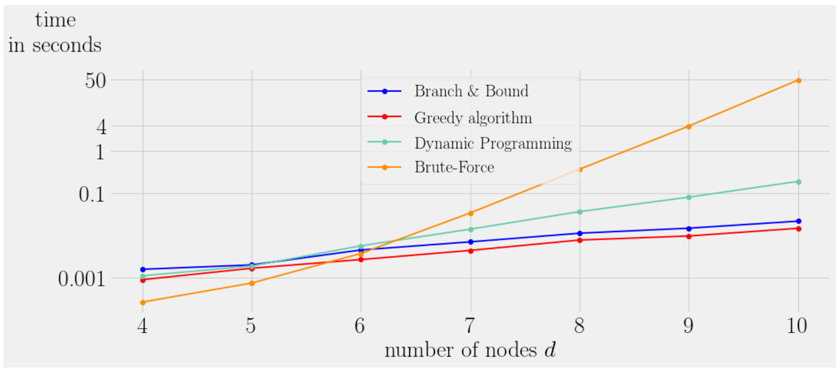


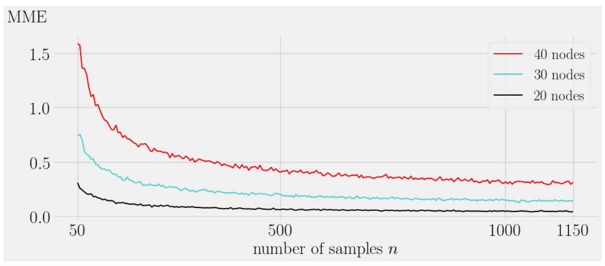
Figure 5.12: We observe that we improved the running time for inferring a topological order of the nodes of an unknown DAG significantly by the Branch & Bound algorithm and by the Greedy algorithm for DAGs with 7 nodes or more compared to the other two methods.

### 5.3.2 Recursive noise model

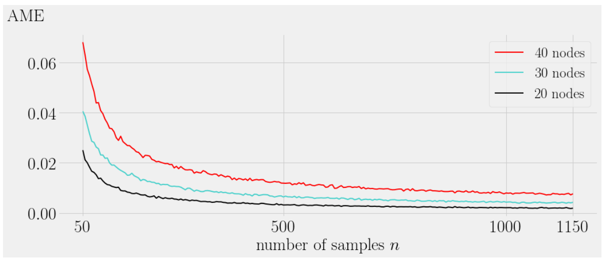
In this section we choose the noise in the following way: $\varepsilon_i = \exp\{E_i\}$ for all $i \in \{1, \ldots, d\}$, where the i.i.d. sequence $(E_i)_{i=1}^d$ is Erlang-distributed with shape parameter $z = 1$ and scale parameter $\lambda = 3$.

#### First setting: Estimation of ML coefficients with known DAG

In the plots in Figure 5.13 we set the probability for setting an edge to $p = 0.2$ as in the non-noisy model in this setting. For other probabilities $p$ the same observations as in the non-noisy model occur. In fact, the higher the probability $p$, the higher both measurement errors MME and AME. We also observe the exponential decay of the MME and the AME as in the non-noisy model.
Although we already increased the sample size in comparison to the non-noisy model to $n = 1150$, convergence is not reached in both plots due to the noise. However, for DAGs with 20 nodes or less, the MME, as well as AME, is close to zero at a sample size of $n = 1000$.



(a) MME as in (5.1)



(b) AME as in (5.2)

Figure 5.13: We fix the probability for setting an edge to $p = 0.2$ as in the non-noisy model. Convergence is more slowly than in the non-noisy model for all number of nodes due to the noise.

### Second setting: estimate ML coefficients and find the minimum ML DAG $\mathcal{D}^B$

In this setting we again use heatmaps with the normed Hamming distance (cf. (5.3)) as a measure of recovery of ML coefficients and of recovery of edges in the minimum ML DAG $\mathcal{D}^B$.

The first observation is that the overall amplitude of errors is larger than in the non-noisy model, it is between 0.0% and 40%. The reason is that we do not observe any atoms (i.e. we do not observe equalities like $\widehat{b}^*_{ij} = \widehat{b}^*_{ik}\widehat{b}^*_{kj}$) and therefore it is harder to infer edges in the minimum ML DAG correctly. The growth of the upper bound of the overall amplitude stems from this fact.

The recovery of ML coefficients is indeed easier than the in the non-noisy model. We chose the noise such that it has support larger than 1. Hence, the value of a node $\widetilde{X}_j$ with incoming paths is larger than in the non-noisy model and therefore the minimum of the ratio $\widetilde{X}_j/\widetilde{X}_i$, $i \in \mathrm{an}(j)$, is larger than in the non-noisy model and the path from node $i$ to node $j$ is easier to identify. On the other hand, the minimum of the ratio $\widetilde{X}_j/\widetilde{X}_i$ is smaller, if $j \in V \setminus \mathrm{An}(i)$, and it is easier to identify, if there is no path from node $i$ to node $j$ and the other way around.

A second observation is, that as the probability of setting an edge $p$ increases, the recovery of edges in the minimum ML DAG $\mathcal{D}^B$ becomes harder, since we have more possibilities not to recognize a critical path. This is different in the non-noisy model, where the recovery of edges in the minimum ML DAG $\mathcal{D}^B$ is easier with increasing $p$.

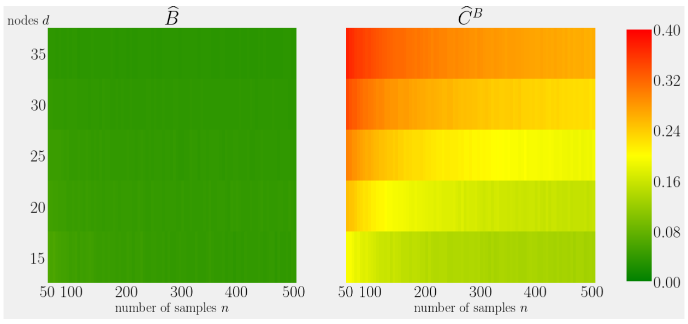


Figure 5.14: Probability for setting an edge is $p = 0.6$. The overall amplitude is larger than in the non-noisy model due to the noise.

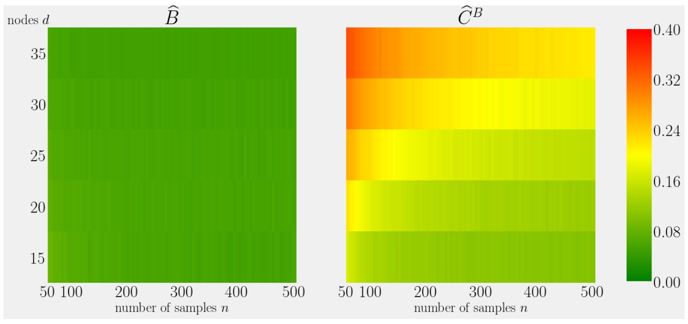


Figure 5.15: Probability for setting an edge is $p = 0.5$.

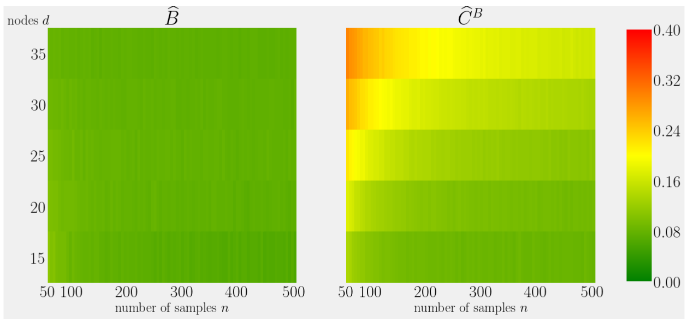


Figure 5.16: Probability for setting an edge is $p = 0.4$.

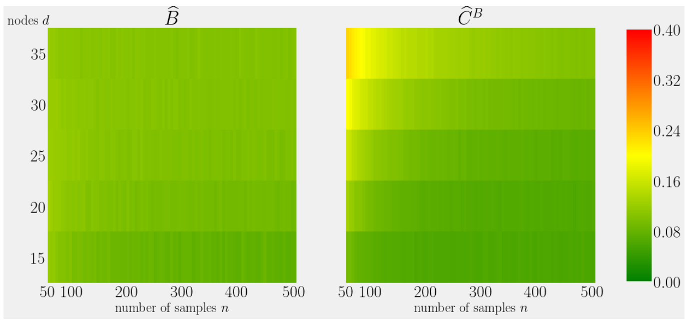


Figure 5.17: Probability for setting an edge is $p = 0.3$.

**Third setting: infer the topological order of the nodes of an unknown DAG**

In Figure 5.18 we see the average time measurement in seconds of one replication of the Branch & Bound algorithm compared to the Greedy algorithm on a logarithmic timeline for the recursive noise model. If we compare it to the time measurement in the non-noisy model (cf. Figure 5.8), we observe that the Branch & Bound algorithm is faster in the recursive-noise model (for instance compare the time for node 60). This is what we expected, since we chose the distribution of the recursive noise with support larger than 1. If we now consider the ratio $\widetilde{X}_j/\widetilde{X}_i$ for $i \in \text{an}(j)$, it is larger than the ratio $X_j/X_i$ in the non-noisy model. For $j \in V \setminus \text{An}(i)$, the ratio $\widetilde{X}_j/\widetilde{X}_i$ becomes smaller than the ratio $X_j/X_i$ in the non-noisy model. Therefore, in the recursive noise model we delete more branches when we apply the Depth-First Search (cf. Algorithm 3.28) and thus Branch & Bound in the recursive noise model is faster than in the non-noisy model.

In Figure 5.19 we compare the OSR in the recursive noise model. Overall, the Branch & Bound algorithm is still better than the Greedy algorithm, however less, than in the non-noisy model. Since we do not hit any atoms in the recursive noise model, the effect of Algorithm 3.18 is weakened and is responsible for this difference.
Although we have noisy data, we observe that the overall performance of both algorithms is still good. That shows, that the algorithms behave very well, even if we do not observe atoms.

The ASR behaves analogoulsy to the ASR in the non-noisy model and the explanation from there also applies here. A little difference is that the decrease from node 13 on is slightly faster than in the non-noisy model. The reason is that the OSR in the recursive noise model decreases faster as well.

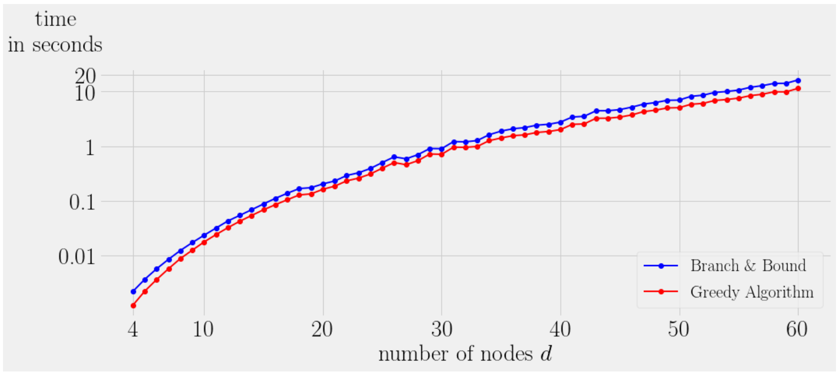


Figure 5.18: Time measurement in seconds on a logarithmic timeline for one replication of the Greedy algorithm compared to the Branch & Bound algorithm for the recursive noise model. We fix the parameters to $n = 500$ and $p = 0.4$.

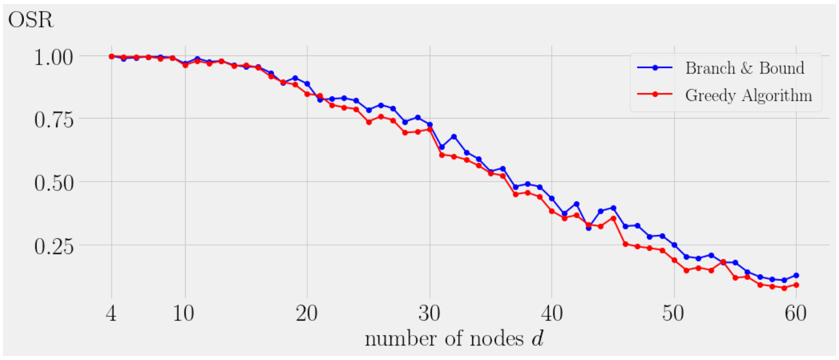


Figure 5.19: OSR as defined in (5.4) for the recursive noise model. We fix the parameters to $n = 500$ and $p = 0.4$.

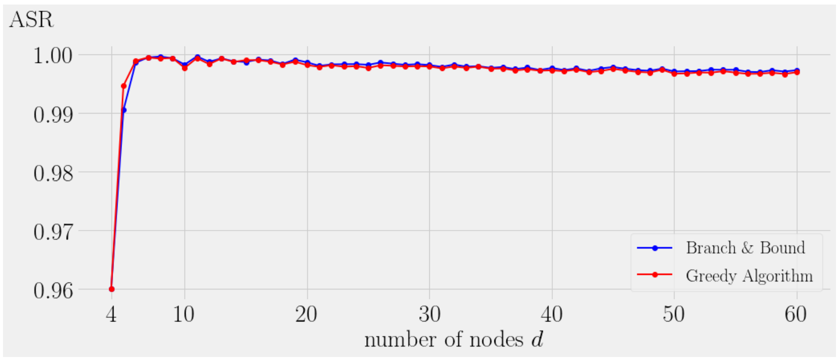


Figure 5.20: ASR as defined in (5.5) for the recursive noise model. We fix the parameters to $n = 500$ and $p = 0.4$. The decrease from node 13 on is slighty faster than in the non-noisy model.

### 5.3.3 Hadamard noise model

In this section we choose a loguniform distribution for the noise variables, i.e. $\varepsilon_i = \exp\{U_i\}$, where $U_i \sim U([\ln(a), \ln(1/a)])$, $a = 0.8$. If we choose the truncated lognormal distribution mentioned in Section 4.2, all results are similar.

#### First setting: Estimation of ML coefficients with known DAG

Since the plots look very similar to the plots in the recursive noise model and no additional knowledge can be gained, we skip the simulations for this setting in the Hadamard noise model.

#### Second setting: infer the ML coefficients and find the minimum ML DAG $\mathcal{D}^B$

Also in the second setting of the Hadamard noise model we use heatmaps with the normed Hamming distance as a measure of recovery of ML coefficients and of recovery of edges in the minimum ML DAG $\mathcal{D}^B$. The overall amplitude is the same as in the non-noisy model, it is between 0.0% and 25%. However, we observe that recovery of edges in the minimum ML DAG $\mathcal{D}^B$ is harder than in the non-noisy model. The reason is that we do not hit any atoms.
A second observation is that the convergence for increasing sample size is weaker than in the recursive noise model. The color in the heatmaps for the minimum ML DAG are more uniform than in the recursive noise model. This is because of the different support of the noise. If we have noise variables with support larger than 1, it is easier to identify critical paths, since the values of the nodes with incoming paths amplify. If we have noise variables with support larger than 1 and smaller than 1, it is harder, since the values of the nodes with incoming paths do not amplify that much.

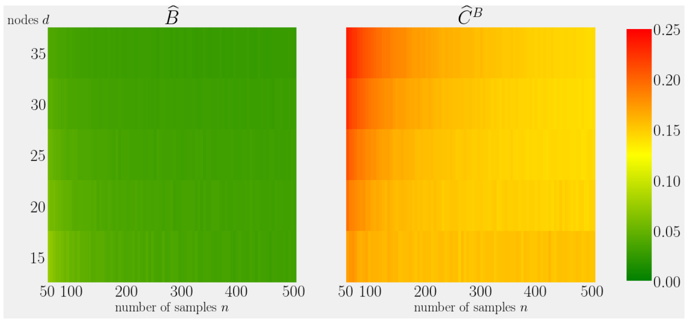


Figure 5.21: Probability for setting an edge is $p = 0.6$. The overall amplitude of errors is the same as in the non-noisy model. We observe that the convergence for recovery of edges with increasing sample size is weaker than in the recursive noise model, due to the different support of the noise.

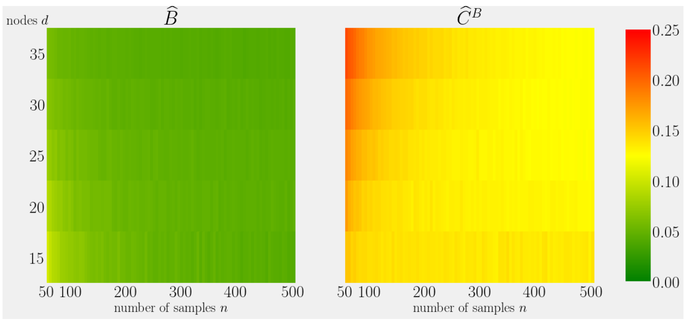


Figure 5.22: Probability for setting an edge is $p = 0.5$.

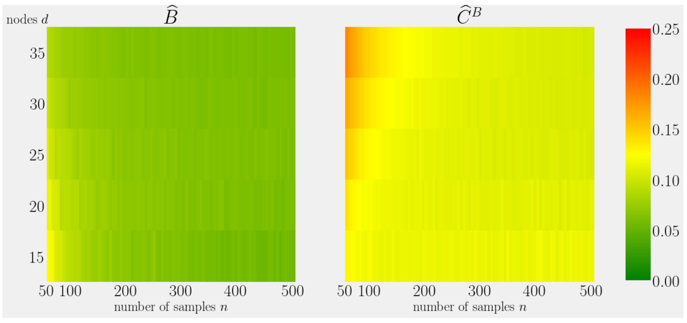


Figure 5.23: Probability for setting an edge is $p = 0.4$.

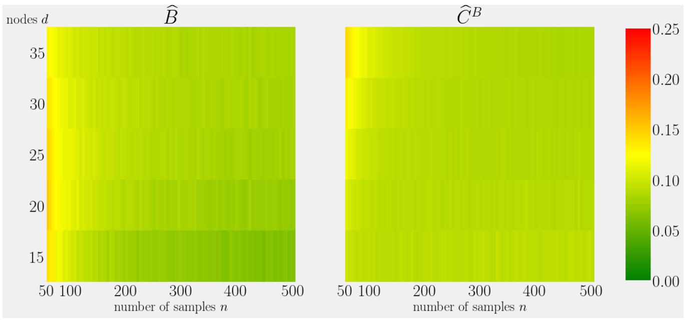


Figure 5.24: Probability for setting an edge is $p = 0.3$.

#### Third setting: infer the topological order of the nodes of an unknown DAG

In Figure 5.26 we compare the OSR between the Branch & Bound algorithm and the Greedy algorithm in the Hadamard noise model. We observe that the improvement of Branch & Bound algorithm compared to the Greedy algorithm is limited. The reason is that Algorithm 3.18 gives little to no additional benefit in the calculation. The ratio $\widetilde{X}_j/\widetilde{X}_i = \varepsilon_j X_j/\varepsilon_i X_i$ needs to come close to the bound $\frac{a}{b}b_{ij}$ for a noticeable effect. However, the probability that the noise variable $\varepsilon_j$ is close to the left interval border $a$ and in the same realization the noise variable $\varepsilon_i$ is close to the right interval border $b$ is low and highly depends on the probability distribution and the tail around the interval borders.

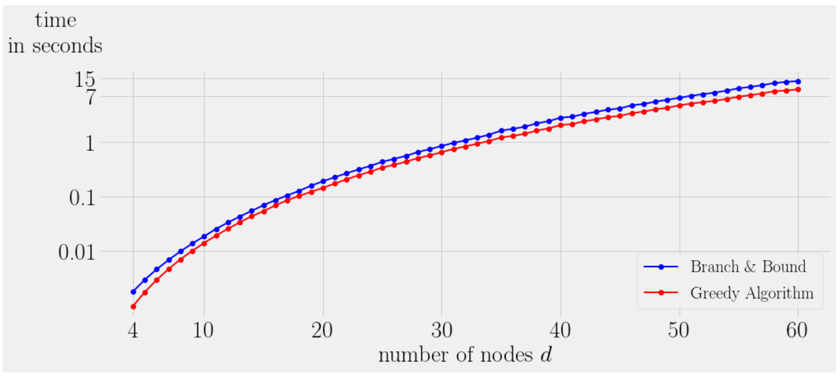


Figure 5.25: Time measurement in seconds on a logarithmic timeline for one replication of the Greedy algorithm compared to the Branch & Bound algorithm for the Hadamard noise model. We fix the parameters to $n = 500$ and $p = 0.4$.

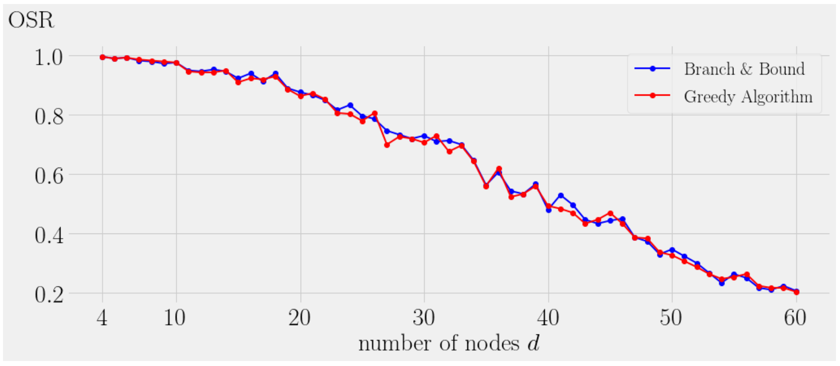


Figure 5.26: OSR as defined in (5.4) for the Hadamard noise model. We fix the parameters to $n = 500$ and $p = 0.4$.

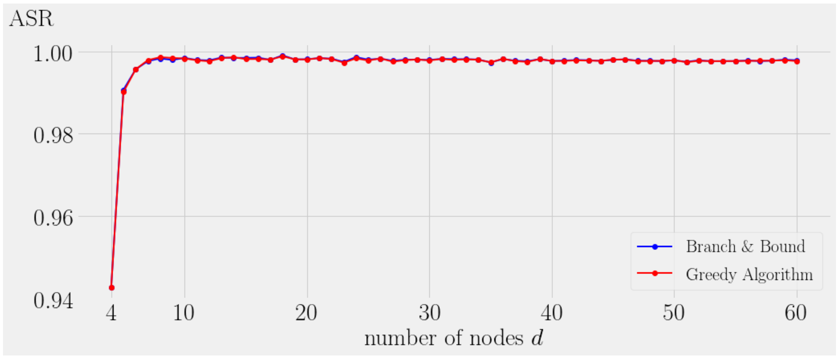


Figure 5.27: ASR as defined in (5.5) for the Hadamard noise model. We fix the parameters to $n = 500$ and $p = 0.4$.

# Appendix A

With the following algorithm we compute the true underlying edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$ in the simulation study. The algorithm corresponds to Definition 2.10.

**Algorithm A.1** (Computation of the edge-weight matrix $C^B$ of the minimum ML DAG $\mathcal{D}^B$)**.**

```
Input:  The edge-weight matrix $C = (c_{ij})_{d\times d}$ and the ML coefficient matrix
        $B = (b_{ij})_{d\times d}$.
Output: The edge-weight matrix $C^B = (c^B_{ij})_{d\times d}$ of the minimum ML DAG $\mathcal{D}^B$.

  1. Initialize $C = C^B$.

  2. For all $i, j \in V$ with $i \neq j$,
         if $b_{ij} > c_{ij}$, then set $c^B_{ij} = 0$;
     end for-loop.
```

With the following algorithm we check whether a given topological order $\pi$ belongs to the equivalence $Q$ induced by the true underlying DAG $\mathcal{D}$. It used in the simulation study.

**Algorithm A.2.**

```
Input:  The ML coefficient matrix $B = (b_{ij})_{d\times d}$ and the given topological
        order $\pi$.
Output: Boolean value.

  1. Set $R = \mathrm{sgn}(B)$. The matrix $R = (r_{ij})_{d\times d}$ is a reachability matrix.

  2. For all $i, j \in V$ with $r_{ij} = 1$ and $i \neq j$,
         if $\pi(i) < \pi(j)$, then continue with the next iteration;
         else, $\pi$ does not belong to $Q$.
```

This algorithm updates a matrix to a reachability matrix.

**Algorithm A.3** (Update $R$ to a reachability matrix)**.**

```
Input:  A reachability matrix $R = (r_{ij})_{d\times d}$ and a pair of vertices $(i, j)$
        with $r_{ij} = 0$.
Output: An updated reachability matrix $R = (r_{ij})_{d\times d}$.

  1. Set $K_1 = \varnothing$ and $K_2 = \varnothing$ and $r_{ij} = 1$.

  2. For all $k_1 \in \{1, \ldots, d\}$ such that $r_{k_1 i} = 1$, set $r_{k_1 j} = 1$ and
     $K_1 = K_1 \cup \{k_1\}$;
     end for-loop.

  3. For all $k_2 \in \{1, \ldots, d\}$ such that $r_{j k_2} = 1$, set $r_{i k_2} = 1$ and
     $K_2 = K_2 \cup \{k_2\}$;
     end for-loop.

  4. If $K_1 \neq \varnothing$ and $K_2 \neq \varnothing$,
         for all $k_1 \in K_1$,
             for all $k_2 \in K_2$,
                 set $r_{k_1 k_2} = 1$;
             end for-loop;
         end for-loop.
```

# Bibliography

P. Asadi, A. C. Davison, and S. Engelke. Extremes on river networks. *Annals of Applied Statistics*, 9(4):2023–2050, 2015.

N. H. Bingham, C. M. Goldie, and J. L. Teugels. *Regular Variation.* Cambridge University Press, 1989.

Y. Bu and J. Lederer. Integrating additional knowledge into estimation of graphical models. `https://arxiv.org/abs/1704.02739`, 2017.

J. Buck and C. Klüppelberg. Recursive max-linear models with propagating noise. In preparation, 2019.

P. Bühlmann and J. Peters. Identifiability of gaussian structural equation models with equal error variances. *Biometrika*, 101:219–228, 2014.

M. Champion, V. Picheny, and Vignes M. Inferring large graphs using $\ell_1-$penalized likelihood. *Statistics and Computing*, 28(4):905–921, 2017.

T. H. Cormen, C. H. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, Cambridge, 3rd edition, 2009.

J. Einmahl, A. Kiriliouk, and J. Segers. A continuous updating weighted least squares estimator of tail dependence in high dimensions. *Extremes*, 21(2):205–233, 2018.

P. Erdös and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

N. Gissibl and C. Klüppelberg. Max-linear models on directed acyclic graphs. *Bernoulli*, 24(4A):2693–2720, 2018.

N. Gissibl, C. Klüppelberg, and J. Mager. Big data: progress in automating extreme risk analysis. In W. Pietsch, J. Wernecke, and M. Ott, editors, *Berechenbarkeit der Welt?* Springer, 2017.

N. Gissibl, C. Klüppelberg, and S. Lauritzen. Identifiability and estimation of recursive max-linear models. `https://arxiv.org/abs/1901.03556`, 2018.

C. Glymour and P. Spirtes. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

J. Hartl. Estimating the coefficients of max-linear structural equation models. Master's thesis, Technische Universität München, 2015.

G. F. Italiano. Amortized efficiency of a path retrieval data structure. *Theoretical Computer Science*, 48:273–281, 1986.

C. Klüppelberg and S. Lauritzen. Bayesian networks for max-linear models. `https://arxiv.org/abs/1901.03948`, 2019.

D. Knuth. *The Art of Computer Programming*, volume 3: Sorting and Searching. Addison-Wesley, 1973.

B. Korte and J. Vygen. *Kombinatorische Optimierung - Theorie und Algorithmen*. Springer-Verlag, 3rd edition, 2018.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

S. Mishra and K. Sikdar. On approximability of linear ordering and related NP-optimization problems on graphs. *Discrete Applied Mathematics*, 136(2-3):249–269, 2004.

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