APRIL 23, 2021

Abstract

Despite having a philosophical grounding from empiricism that span some centuries [1], the algorithmization of causal discovery started only a few decades ago [2]. This formalization of studying causal relationships relies on connections between graphs and probability distributions. In this setting, the task of causal discovery is to recover the graph that best describes the causal structure based on the data we receive. A particular class of causal discovery algorithms, called constraint-based methods rely on Directed Acylic Graphs (DAGs) as an encoding of Conditional Independence (CI) relations that carry some level of causal information. However, a CI relation such as X and Y being independent conditioned on Z assumes the independence holds for all possible values Z can take, which can tend to be unrealistic in practice where causal relations are often context-specific [3]. In this thesis we aim to develop constraint based algorithms to learn causal structure from Context-Specific Independence (CSI) relations within the discrete setting, where the independence relations are of the form X and Y being independent of $Z = a$ for some a. This is done by using Context-Specific trees, or CSTrees for short, [4] which can encode CSI relations.

[1] David Hume. *Enquiry Concerning Human Understanding*. Clarendon Press, 1904

[2] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018

[3] Rodrigo A Collazo, Christiane Görgen, and Jim Q Smith. *Chain event graphs*. CRC Press, 2018; Dan Geiger and David Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82(1-2):45–74, 1996; and Nir Friedman and Moises Goldszmidt. Learning bayesian networks with local structure. In *Learning in graphical models*, pages 421–459. Springer, 1998

[4] Eliana Duarte and Liam Solus. Representation and learning of context-specific causal models with observational and interventional data, 2021

Acknowledgements

Contents

1

Introduction

1.1 Motivation

At the heart of scientific discovery, and thus of modern society, is our ability to gain knowledge about causal relations based on observations and experimentation. Whilst being an active research area for its own sake, causal discovery has many applications to diverse fields ranging from economics [1] and genomics [2] to the climate sciences [3]. In fact, any field involving any form of measurements/observations which are hypothesized to involve causal interactions can benefit from the tools which this research area has to offer, by allowing the user to get some causal model of the system from which their data is generated from. Armed with such a causal model, the user can answer more complex queries than they can using just observed data. Such queries form a hierarchy, referred to as Pearl's Causal Hierarchy, which starts from those related to observations, then interventions, then counterfactuals [4]. In fact it has been recently shown that their strict distinction holds in the measure theoretic sense [5].

[1] Biwei Huang, Kun Zhang, Mingming Gong, and Clark Glymour. Causal discovery and forecasting in nonstationary environments with state-space models, 2019

[2] Pengfei Hu, Rong Jiao, Li Jin, and Momiao Xiong. Application of causal inference to genomic analysis: Advances in methodology. *Frontiers in Genetics*, 9:238, 2018

[3] Jakob Runge, Sebastian Bathiany, Erik Bollt, Gustau Camps-Valls, Dim Coumou, Ethan Deyle, Clark Glymour, Marlene Kretschmer, Miguel D Mahecha, Jordi Muñoz-Marí, et al. Inferring causation from time series in earth system sciences. *Nature communications*, 10(1):1–13, 2019

[4] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018

[5] Duligur Ibeling Thomas Icard Elias Bareinboim, Juan D. Correa. On pearl's hierarchy and the foundations of causal inference, 2020

1.2 Classical methods

Historically speaking the task of causal discovery was achieved by using clever experimental design [6]. In particular, Randomized Controlled Trials (RCTs) are often called the gold standard in this regard. The common RCT setting involves allocating subjects in the experiment randomly into 2 groups, where one group called the treatment group receives an intervention and the other group called the control group receives no intervention, often in the form of a placebo. One of the main limitations of RCTs is that not every system on which we want to infer causal relationships lends to this setting. For example, it is not a good idea to have a RCT to determine

[6] R.A. Fisher. *The design of experiments. 1935*. Oliver and Boyd, Edinburgh, 1935

whether smoking causes lung cancer which would involve forcing the subjects to smoke.

1.3 Relevance to machine learning

Causal discovery as a subfield of causal modelling contains many ideas which can help in overcoming hard barriers in machine learning. Machine learning can be summarized as the field where practitioners formulate mathematical models of a system of interest, followed by incorporating observed data into this model using various algorithms with the aim of making better predictions about the system. This field has been enjoying significant breakthroughs recently in part due to the availability of a lot of data and faster computers. However, a lot of the work in this field are set in the assumption of independent and identically distributed (i.i.d) data, and ignores information from interventions, domain shifts and temporal structure [7]. As such, there are various problems which still require a causal model, which without it in some cases even give rise to seemingly nuanced paradoxes [8], such as Simpsons paradox [9], where one might for example have an increasing relationship between 2 variables over the whole data, but dividing the samples into further groups would result in a decreasing relationship within each group. This is not just a theoretical issue, and has been reported in many real life data as well [10].

[7] Bernhard Schölkopf. Causality for machine learning, 2019

[8] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018

[9] Edward H Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(2):238–241, 1951

[10] Clifford H. Wagner. Simpson's paradox in real life. *The American Statistician*, 36(1):46–48, 1982

2

The Causal Discovery problem

We first provide a formalization of the causal discovery problem. Suppose we have a system of p variables $X_1, ..., X_p$ which we assume has some underlying probability distribution $\mathbb{P}$, and from which we have n samples $\{x_1^i, ..., x_p^i\}_{i=1}^n$. The goal of causal discovery is to recover a structure $\mathbb{G}$ that best represents the causal mechanisms of the system with. The structure $\mathbb{G}$ is often a graph with certain properties that enables it to encode information about the system - this means we must make an assumption that such a structure $\mathbb{G}$ exists and it is related to the distribution $\mathbb{P}$. This information about the system is extracted from the samples we have from the distribution $\mathbb{P}$ - this means we have to make further assumptions to relate information we get from samples in $\mathbb{P}$ to our structure $\mathbb{G}$.

The assumptions to be made are an inevitable artefact of the No Free Lunch theorem [1] which state that over a uniform distribution over search/learning problems (which includes causal discovery), all algorithms for such problems have equal performance.

[1] David H. Wolpert. What is important about the no free lunch theorems?, 2020

There are two common approaches to causal discovery [2]. The first is constraint-based methods, which treat the problem of finding the structure as a contraint satisfaction problem. One approach to this is to start from a structure where all variables are causally connected then removes connections based on statistical independence from the observed samples. Second is score based methods, which select a causal representation by assigning a score to all possible models, and then choosing a model that minimizes the score. One approach in this direction is to start from a structure where all variables are not causally connected and then proceed to add connections based on how the observed samples give some score, like the Bayesian Information Criterion (BIC). In this thesis we will mainly be concerned with constraint based methods, particularly in the discrete setting,

[2] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10(nil):nil, 2019

where we assume the variables in the system can only take discrete values.

3

Causal Discovery with Directed Acyclic Graphs

We start this section with important definitions and concepts related to Directed Acyclic Graphs (DAGs). They are a convenient and informative graphical means of visualizing the directional relationships between variables in a system, and the de-facto choice to model causal structures.

3.1 Direct Acyclic Graphs (DAGs)

Definition 3.1 (DAGs) *A Directed Acyclic Graphic (DAG) is a directed graph* $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ *which has no cycles.*

Since we a working with discrete probability distributions, we introduce the (open) probability simplex as the space of all possible probability distributions over a set of discrete variables $X_1, ..., X_p$ whose outcomes are elements of $\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i$.

Definition 3.2 (Probability simplex) *Given a finite set* $\mathcal{X}$*, The probability simplex on this set is*
$\Delta_{|\mathcal{X}|-1} = \{(f_x : x \in \mathcal{X}) \in \mathbb{R}^{|\mathcal{X}|} : \forall x \in \mathcal{X}\ f_x > 0, \sum_{x \in \mathcal{X}} f_x = 1\}$

Each point in the probability simplex corresponds to a joint distribution over $(X_1, ..., X_p)$, and our interest mainly lies to the subset of of this space which are connected to structures we can use to model causal relations.

An important concept when relating DAGs to distributions is that of conditional independence, which we define below.

Definition 3.3 (Conditional Independence) *Let* $\mathbb{P}$ *be a distribution with variables* $X_1, ..., X_p$*. Given non-empty subsets* $A, B \subset [p]$ *and a (possibly empty) subset* $S \subset [p]$ *such that* $\mathbb{P}(X_B, X_S) > 0$ *and* $A \cap B \cap S = \{\}$*, we say the variables* X_A *and* X_B *are conditional independent given* S*,*

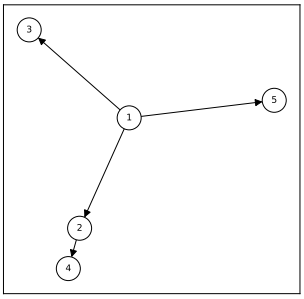

Figure 3.1: Example of a DAG.

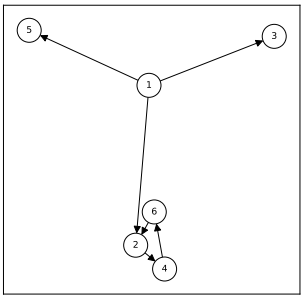

Figure 3.2: This graph is not a DAG since there is a cycle

(denoted $(X_A \perp\!\!\!\perp_{\mathbb{P}} X_B \mid X_S)$*) if* $\mathbb{P}(X_A, \mid X_B, X_S) = \mathbb{P}(X_A \mid X_S)$ *holds for all possible outcomes of* X_A, X_B, X_S*.*

The conditional independence statement $(X_A \perp\!\!\!\perp_{\mathbb{P}} X_B \mid X_S)$ can be viewed as a ternary relation on X_A, X_B, X_S, and is called a Conditional Independence (CI) relation. This relation formalizes the concept of X_B and X_A not providing any information when we have obsered X_S, which is to say, if we already know X_S, knowing X_B does not change the probabilities for X_A, and vice versa.

Using this we can now define the local Markov property which relates d-separations in DAGs to CI relations in distributions.

Theorem 3.4 (Local Markov property) *Let* $\mathcal{G}$ *be a DAG with nodes* $[p]$*. A probability distribution* $\mathbb{P}$ *satisfies the local Markov property with respect to* $\mathbb{G}$ *if for each node* $i \in [p]$*, the variable representing that node,* X_i *is independent of its non-descendants when conditioned on its parents, formally,* $(X_i \perp\!\!\!\perp X_{ND_{\mathbb{G}}(i)} \mid X_{PA_{\mathbb{G}}(i)})$

This formalizes the fact that in order to computationally generate data from a DAG $\mathbb{G}$, the value of each variable $X_i \in \mathcal{X}_i$ depends only on the values of the outcomes of its parents in $\mathbb{G}$. This means that for a (discrete) distribution $\mathbb{P}$ with p variables satisfying the Local Markov property, the distribution can be encoded with p probability tables which give the probabilities for each X_i taking a value when conditioned on all possible outcomes of its parents. From a storage perspective, this means we have to store $\sum_{i=1}^{p} |\mathcal{X}_i||\prod_{j\in PA_{\mathbb{G}}(i)} \mathcal{X}_j|$ which is significantly smaller than having to store all possible probability values which would require one table with $|\prod_{i=1}^{p} |\mathcal{X}_i|$ values. For binary variables assuming d parents for each variable, this is the difference between $p2^{d+1}$ and $p2^p$.

For the purposes of this thesis, it is worth introducing the Ordered Markov property which uses the concept of a linear ordering. [1]

Theorem 3.5 (Ordered Markov Property) *Let* $\mathbb{G}$ *be a DAG and* $\pi = \pi_1 \cdots \pi_p$ *a causal ordering of G. A probability distribution* $\mathbb{P}$ *satisfies the Ordered Markov property with respect to* $\mathbb{G}$ *if we have* $(X_i \perp\!\!\!\perp X_{\{1,\ldots,i-1\}\setminus PA_{\mathbb{G}}(i)} \mid X_{PA_{\mathbb{G}}(i)})$

An important notion in DAGs is that of d-separation and blocked paths. [2] .

Definition 3.6 (Blocked path) *Given a DAG* $\mathbb{G}$*, and a path between nodes* $i, j \in \mathbb{V}$*, we say the* ***path is blocked*** *by a (potentially empty) set of nodes S if either of the following hold:*

- *Along the path there is a triple of nodes* (x, s, y) *such that* $x \to s \to y$*,*

[1] For a DAG G with p nodes a linear ordering is an ordering of the nodes that respects the directions in G that is each node i always comes after each $j \in PA_{\mathbb{G}}(i)$. It is a also called a topological ordering

[2] A path between 2 nodes is any set of edges connecting them irrespective of the direction.

$x \leftarrow s \leftarrow y$, *or* $x \leftarrow s \rightarrow y$ *with* $s \in S$

- *Along the path there is a triple of nodes* (x, s, y) *such that* $x \rightarrow s \leftarrow y$ *such that* $s \notin S$ *and no descendants of* s *are in* S.

Definition 3.7 (d-separation) *Given a DAG* $\mathbb{G}$*, two (non-empty) sets of nodes* X, Y *are* ***d-separated*** *by a (potentially empty) set of nodes* S *in* $\mathbb{G}$*, denoted* $(X \perp\!\!\!\perp_{\mathbb{G}} Y \mid S)$ *if all paths between every node in* X *and every node in* Y *are blocked by* S.

The notion of d-separation relates DAGs to probability distributions from the following theorem.

Theorem 3.8 (Global Markov property) *Given a distribution* $\mathbb{P}$ *that satisfies the local Markov property with a DAG* $\mathbb{G}$*, we have that for any (non-empty) sets* A, B *and (possibly empty) set* S*,* $(X_A \perp\!\!\!\perp_{\mathbb{G}} X_B \mid X_S) \implies (X_A \perp\!\!\!\perp_{\mathbb{P}} X_B \mid X_S)$

An important result is the following that the above notions are indeed equivalent [3].

Theorem 3.9 (Markov theorems for DAGs) *Given a distribution* $\mathbb{P}$ *over* $X_1, ..., X_p$ *and a DAG* $\mathbb{G}$ *over* p *nodes, the following are equivalent*

- $\mathbb{P}$ *is Markov to* $\mathbb{G}$ *i.e.* $\mathbb{P}(X_1, ..., X_p) = \prod_{i=1}^{p} \mathbb{P}(X_i \mid X_{PA_{\mathbb{G}}(i)})$
- $\mathbb{P}, \mathbb{G}$ *satisfy the local Markov property*
- $\mathbb{P}, \mathbb{G}$ *satisfy the ordered Markov property*
- $\mathbb{P}, \mathbb{G}$ *satisfy the global Markov property*

If $\mathbb{P}$ satisfies the local Markov property with $\mathbb{G}$ and has a probability density with respect to a product measure, we say $\mathbb{P}$ is Markov with respect to $\mathbb{G}$, or equivalently, $\mathbb{G}$ is an Independence map (I-MAP) of $\mathbb{P}$ [4]. An important result is that

Thus DAGs can be used to store Conditional Independence (CI) relations. More importantly, d-separation encodes the complete set of CI relations satisfied by all distributions Markov to a DAG, i.e. distributions that are Markov to a DAG $\mathbb{G}$ **and** satisfy **exactly** the CI relations encoded by d-separation exist [5].

It is also possible to have 2 DAGs that encode the same CI relations, in which case we say that they are both in the same Markov Equivalence Class (MEC), and we say they are Markov Equivalent. MECs can be characterized by the following theorem [6].

Theorem 3.10 (Characterization of MECs) *Two DAGs* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *are Markov Equivalent if and only if they have the same skeleton (underlying*

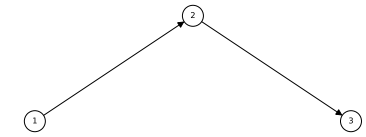

Figure 3.3: Chain

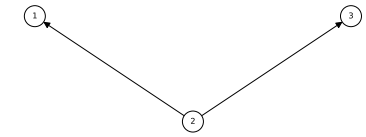

Figure 3.4: Fork/Common cause

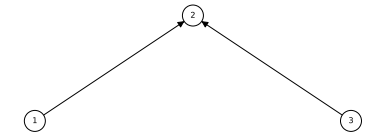

Figure 3.5: V-structure/ Collider/Immorality

[3] Eliana Duarte and Liam Solus. Algebraic geometry of discrete interventional models, 2020

[4] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996

[5] Christopher Meek. Strong completeness and faithfulness in bayesian networks. *arXiv preprint arXiv:1302.4973*, 2013; and

[6] Tom S. Verma and Judea Pearl. On the equivalence of causal models. *CoRR*, abs/1304.1108, 2013

undirected edges) and v-structures, where a v-structure is a triple of nodes (i, j, k) *with edges* $i \to j \leftarrow k$ *and* i, k *do not share an edge.*

For example, the Chain and Fork graphs from the previous page belong to the same Markov Equivalence class.

3.2 Causal Discovery Algorithms for DAGs

Theorem 3.8 suggests that we can make use of CI testing on a distribution $\mathbb{P}$ to learn a DAG G. However, the distribution $\mathbb{P}$ may contain CI relations not encoded in the DAG, thus we make the following assumption.

Definition 3.11 (Faithfulness) *A probability distribution* $\mathbb{P}$ *is faithful to a DAG* $\mathbb{G}$ *if it entails only the CI relations encoded by the d-separations in the DAG.*

Under the faithfulness assumption, Theorem 3.8 holds both ways. It should be noted that faithful distributions exist [7], and the set of distributions that are not faithful to a dag $\mathbb{G}$ have measure 0 [8], which suggets that in theory this is not a very restrictive assumption.

The PC algorithm [9] is a constraint based causal discovery algorithm which relies of the characterization of DAGs in Theorem 3.10 and the faithfulness assumption to find a DAG in the MEC of the true causal DAG. The algorithm starts from a complete graph and runs conditional independence tests to first find the DAG skeleton and then proceeds to direct the edges whenever possible. The output of the PC algorithm is a Completed Partially Directed Acyclic Graph (CPDAG) [10] [11] , which acts as a representation for the Markov Equivalence class. A Partially Directed Acylic Graph (PDAG) is a graph where some edges are directed and some are undirected and there is no cycle in the direction of the directed edges and any direction of the undirected edges. A PDAG is is Complete PDAG (CPDAG) if every directed edge exists also in every DAG in the Markov Equivalence class of the DAG and for every undirected edge between nodes i, j there exists a DAG with the edge $i \to j$ and a DAG with $j \to i$ in the equivalence class.

[7] Christopher Meek. Strong completeness and faithfulness in bayesian networks. *arXiv preprint arXiv:1302.4973*, 2013

[8] Caroline Uhler, Garvesh Raskutti, Peter Bühlmann, and Bin Yu. Geometry of the Faithfulness Assumption in Causal inference. *The Annals of Statistics*, 41(2):436 – 463, 2013

[9] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2nd edition, 2000; and Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(22):613–636, 2007

[10] Christopher Meek. Causal inference and causal explanation with background knowledge, 2013

[11] The CPDAG is also sometimes referred to as the Essential Graph or the DAG pattern

3.3 Limitations of using DAGs

DAGs are a simple and informative structure for causal discovery, however their ability to only encode CI relations is a limitation. This is because the CI relation $(X_A \perp\!\!\!\perp_{\mathbb{P}} X_B \mid X_S)$ implies that X_A and X_B are independent for all possible outcomes of X_S, which in some cases might be too strong of an assumption. A generalization of such

relations is Context Specific Independence (CSI) relations, defined below.

Definition 3.12 (Context Specific Independence) *Let $\mathbb{P}$ be a distribution with variables $X_1, ..., X_p$ with a state space $\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i$. Given non-empty subsets $A, B \subset [p]$ and (possibly empty) subsets $S, C \subset [p]$ and $x_C \in \prod_{i \in C} \mathcal{X}_i$ such that $\mathbb{P}(X_B, X_S, X_C = x_C) > 0$ and $A \cap B \cap S \cap C = \{\}$, we say the variables X_A and X_B are conditional independent given S, in the context $X_C = x_C$ (denoted $(X_A \perp\!\!\!\perp_{\mathbb{P}} X_B \mid X_S)$) if $\mathbb{P}(X_A \mid X_B, X_S, X_C = x_C) = \mathbb{P}(X_A \mid X_S, X_C = x_C)$ holds for all possible outcomes of X_A, X_B, X_S.*

In the next chapter we introduce Context Specific Trees (CSTrees) which can encode such relations, and thus provide a structure that can capture the context specific information glossed over in DAGs.

4

Causal Discovery with Context Specific Trees

One intuition is that to capture context specific relations one needs to make use of a structure that explicitly represents separate outcomes of a distribution. Typically in high school some might have encountered the use of trees to model small probabilistic systems, and they fully include all possible outcomes involved, and serve as an important tool to compute probabilities for relevant events. As we will see in this chapter, this is a correct way to approach the problem of encoding context information as well.

4.1 Context Specific Trees (CSTrees)

Before defining CSTrees we start by defining staged trees, which contain CSTrees as a subset. Both of these are rooted trees. [1]

[1] $CH_{\mathbb{G}}(v)$ refers to the set of children of node v in $\mathbb{G}$. A rooted tree $\mathbb{T} = (\mathbb{V}, \mathbb{E})$ is a directed graph whose skeleton is a tree and there exists a unique node r such that $PA_{\mathbb{T}}(r) = \{\}$ which is called the root.

Definition 4.1 (Staged trees) *Let $\mathbb{T} = (\mathbb{V}, \mathbb{E})$ be a rooted tree, $\mathbb{L}$ a finite set of labels for the edges, and $\theta : \mathbb{E} \to \mathbb{L}$ a labelling of the edges. Let $E_{\mathbb{T}}(v) = \{v \to w \in \mathbb{E} : w \in CH_{\mathbb{T}}(v)\}$, i.e. the set of edges coming out of v in $\mathbb{T}$. The pair $(\mathbb{T}, \theta)$ is a staged tree if*

- $\forall v \in \mathbb{V}$ *we have* $|\theta(E_{\mathbb{T}}(v))| = |E_{\mathbb{T}}(v)|$
- $\forall v, w \in \mathbb{V}$ *we have that both $\theta(E_{\mathbb{T}}(v))$ and $\theta(E_{\mathbb{T}}(w))$ are either equal or disjoint*

This can be thought of as a probability tree where each edge represents a probability value, and the probabilities coming out of all edges from any given node sum to 1. More formally, first define the space of canonical parameters of the staged tree $(\mathbb{T}, \theta)$ as

$$\Theta_{\mathbb{T}} = \{x \in \mathbb{R}^{|\mathbb{L}|} : \forall e \in \mathbb{E}, x_{\theta(e)} \in (0,1), \forall v \in \mathbb{V} \textstyle\sum_{e \in E_{\mathbb{T}}(v)} x_{\theta(e)} = 1\}.$$

Given the probability simplex $\Delta_{\mathcal{X}-1}$ and letting $\mathbf{i}_{\mathbb{T}}$ be the set of all leaves of the staged tree $\mathbb{T}$ the staged tree model is defined as below.

Definition 4.2 (Staged tree models) *The staged tree model* $\mathbb{M}_{(\mathbb{T},\theta)}$ *is the image of the map* $\varphi_{\mathbb{T}} : x \to f_v := (\prod_{e \in E_{\mathbb{T}(\lambda(v))}} x_{\theta(e)})_{v \in \mathbf{i}_{\mathbb{T}}}$

Thus given variables $X_1, ..., X_p$, a causal ordering π, the staged tree for this with levels [2] L_1

[2] The k^{th} level of a rooted tree, L_k, is the set of nodes such that the unique path from each node in L_k to the root consists of k edges.

The important characteristic of staged trees are the stages.

Definition 4.3 (Stages) *Given a staged tree* $(\mathbb{T}, \theta)$*, we say two nodes* v, w *are in the same stage if and only if* $\theta(E_{\mathbb{T}}(v)) = \theta(E_{\mathbb{T}}(w))$

Stages are represented by colours, and when a stage contains a single node, it is coloured white. Staged tree models generalize DAG models, i.e. distributions represented by DAGs, however they do not take into account the CI relations encoded in them, which arise from the ordered Markov property. Thus we need a structure that generalize DAG models **and** takes these CI relations into account. A subclass of staged trees, known as CSTrees allow for this, and more importantly, enable us to generalize to the context specific scenario.

Definition 4.4 (CSTrees) *Let* $\mathcal{X}_i$ *denote the state space of some variable* X_i *with* $\mathcal{X} = \Pi_{i=1}^{p} \mathcal{X}_i$*, and* $(\mathbb{T}, \theta)$ *be a staged tree with levels* $L_1, ..., L_p$ *corresponding to variables* $X_{\pi_1}, ..., X_{\pi_p}$ *where* $\pi = \pi_1 ... \pi_p$ *is the causal ordering of the variables. A CSTree is a staged tree* $(\mathbb{T}, \theta)$ *where each level of the tree corresponds to some variable and such that*

- *It is compatibly labelled, i.e.* $\forall x_{\pi_k} \in \mathcal{X}_i$ *we have* $\theta(x_{\pi_k} ... x_{\pi_{k-1}} \to x_{\pi_{k-1}} x_{\pi_k}) = \theta(y_{\pi_k} ... y_{\pi_{k-1}} \to y_{\pi_{k-1}} x_{\pi_k})$ *whenever* $x_{\pi_1} ... x_{\pi_{k-1}}$ *and* $y_{\pi_1} ... y_{\pi_{k-1}}$ *are in the same stage*
- *(**CSTree property**) Each stage* $S_i \subset L_k$ *of the tree has a fixed context, i.e.* $\exists C_i \subset [k]$ *and the fixed outcome context* $x_{C_i} \in \mathcal{X}_{C_i}$*, where the stages are the union over the variables beside those in* C_i*, i.e. if* $Y_i = [k] \backslash C_i$ *then* $S_i = \bigcup_{x_{Y_i} \in \mathcal{X}_{Y_i}} \{x_{C_i} x_{Y_i}\}$

Given a CSTree $\mathbb{T}$ and a causal ordering π, each node in level L_k corresponds to an outcome of the sequence of variables $X_{\pi_1}, ..., X_{\pi_k}$. Each edge coming into each node in L_k is of the form $(x_1 \cdots x_{k-1}, x_1 \cdots x_k)$ represents $P(x_k | x_1 \cdots x_{k-1})$. Suppose we fix a node $n = a_1 \cdots a_k \in L_k$. Each edge coming out of n gives the probabilities for the variable in the next level L_{k+1}, conditioned on the context $(X_{\pi_1} = a_1, ..., X_{\pi_k} = a_k)$. Thus, we can view this node n as containing the distribution $\mathbb{P}(X_{\pi_{k+1}} \mid X_{\pi_1} = a_1, ..., X_{\pi_k} = a_k)$ This is an important view which we will make use of when testing for context specific independence in the algorithms throughout this paper. We show an example of a CSTree and a staged tree that is not a CSTree below.

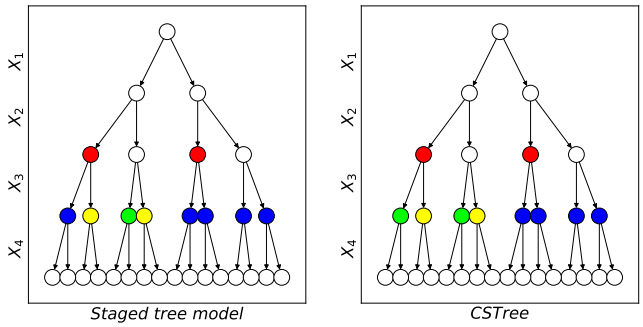

Figure 4.1: Example of a staged tree model that is not a CSTree (Left) and a CSTree (right) for binary variables X_1, X_2, X_3, X_4 in that causal ordering.

Both staged trees is Figure 4.1 represent 4 binary variables X_1, X_2, X_3, X_4 taking values in $\{0, 1\}$ in that causal order. Suppose each edge to the left corresponds to the outcome 0 and the other corresponds to one. In this case, the left edge coming out of the root represents $\mathbb{P}(X_1 = 0)$ and the right edge coming out the root represents $\mathbb{P}(X_1 = 1)$. The nodes represent distributions conditioned on the context unique to them. For example, the left most red node in both trees represent $\mathbb{P}(X_3 \mid X_2 = 0, X_1 = 0)$. The tree on the right is a CSTree because each of the nodes in the non-singleton stages, which are represented by a non-white colour, share exactly one fixed context. For example, the stage corresponding to the blue nodes in the tree on the right (the CSTree) corresponds to the contexts $(X_1 = 1, X_2 = 0, X_3 = 0), (X_1 = 1, X_2 = 1, X_3 = 1), (X_1 = 1, X_2 = 1, X_3 = 0), (X_1 = 1, X_2 = 1, X_3 = 1)$. The common context for this stage is $(X_1 = 1)$. Meanwhile, for the tree on the left, the stage corresponding to the blue nodes only share the empty context, meaning all nodes in level 3 must correspond to the stage with the empty context for it to be a CSTree.

A CSTree encodes Context Specific Independence (CSI) relations from the following lemma [3]

Lemma 4.5 (CSTrees and Context Specific Independence relations) *Let $\mathbb{T} = (\mathbb{V}, \mathbb{E})$ be a CSTree with levels $X_1, ..., X_p \sim L_1, ..., L_p$ and stages $S_1, ..., S_m$. Then for any $\mathbb{P} \in \mathbb{M}_{(\mathbb{T},\theta)}$ and $S_i \subset L_{k-1}$, $\mathbb{P}$ entails the CSI relation $(X_k \perp\!\!\!\perp_{\mathbb{P}} X_{[k-1]\setminus C_i} \mid X_{C_i} = x_{C_i})$ where C_i is the context fixed by the stage S_i.*

[3] Eliana Duarte and Liam Solus. Representation and learning of context-specific causal models with observational and interventional data, 2021

4.2 Learning CSTrees from observed data

Given a system of variables $X_1, ..., X_p$, we would first need a causal ordering $\pi_1 \cdots \pi_p$ in order to construct a CSTree for these variables. Since CSTrees encode CSI relations, they can also encode CI relations, which means we can generate a CSTree from a DAG. The following

proposition formalizes this notion [4].

[4] Eliana Duarte and Liam Solus. Algebraic geometry of discrete interventional models, 2020

Proposition 4.6 (Equivalent DAGs and CSTrees) *A compatibly labelled staged tree $\mathbb{T}$ with causal ordering $\pi_1 \cdots \pi_p$, levels $L_1, ..., L_p$ corresponding to variables $X_{\pi_1}, ..., X_{\pi_p}$ encodes the same CI relations as some DAG $\mathbb{G}$ if and only if for any topological ordering of $\mathbb{G}$, $\forall k \in [p-1]$, the level L_k has its nodes partitioned into stages where the context for each stage is an element of the Cartesian product of the parents of $X_{\pi_{k+1}}$ in $\mathbb{G}$.*

We describe the computational procedure to generate a CSTree $\mathbb{T}$ from a DAG $\mathbb{G}$ below, assuming that we are given a causal ordering of $\mathbb{G}$. [5] .

[5] PARENTS is a function that takes a graph and a node and returns the parents of that node in the graph; CARTESIANPRODUCT takes a set of variables and returns the cartesian product of these variables i.e. all possible values they can take

Algorithm 1: DAGTOCSTREE
Constructing a CSTree from a DAG

Input: A DAG G, causal ordering O
Output: CSTree T with ordering O and stages S defined by G

$T \leftarrow$ Empty staged tree with ordering O;
$S \leftarrow$ Empty dictionary;
for l *in* $|O| - 1$ **do**
  $v \leftarrow O[l+1]$;
  T.*add_level*(v);
  $a \leftarrow$ PARENTS(G, v);
  $b \leftarrow$ CARTESIANPRODUCT(a);
  `// Each element of` b `is a context which fixes a stage in level` l
  for c *in* b **do**
    $S[c] \leftarrow$ [nodes in level l such that c is a subcontext];
  end
end
return T, S

Algorithm 1 above does not necessarily need a causal ordering since given a DAG we can perform a topological sort on it to get one.

Theorem 4.7 *Given variables $X_1, ..., X_p$ taking values in $\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i$, Algorithm 1 is correct and runs in $\mathcal{O}(d^{2p})$ time and $\mathcal{O}(d^p)$ space where $d = \max_{i \in [p]} |\mathcal{X}_i|$.*

Proof: For correctness, at each level L_k, the non-singleton stages are created for the contexts fixed by the outcomes of the parents of $X_{\pi_{k+1}}$ thus by Proposition 4.6 the tree is still a CSTree. Since the staging process at each level only creates non-singleton of nodes within that level, and we go over each level except the last level which always contains singleton stages, the stages S lead to T being a CSTree. For time complexity, the worst case scenario is for the fully connected DAG, assuming the ordering $(1, 2, ..., p)$, node i has

$i-1$ parents. This however results in a CSTree with no non-singleton stages. Thus we look at the scenario where node i has $i-2$ parents. At the level for the variable representing node i, the variable b in Algorithm 1 which is all the elements of the the Cartesian product of values the parents take, has $|\prod_{j=1}^{i-2}\mathcal{X}_j|$ elements. For each element in this Cartesian product which fixes the context for the stage, we have to loop over all nodes in level i and to store the nodes for that stage, and level i has $|\prod_{j=1}^{i}\mathcal{X}_j|$ nodes. Thus the loop for level i takes $|\prod_{j=3}^{i-2}\mathcal{X}_j||\prod_{j=1}^{i}\mathcal{X}_j|$ where the indexing starts at 3 for the first term since the parent sets are non-empty starting from node 3. Since we have p levels, ignoring the first 2 since their variables have no parents, we have $\sum_{i=3}^{p}|\prod_{j=3}^{i-2}\mathcal{X}_j||\prod_{k=1}^{i}\mathcal{X}_k| < \sum_{i=1}^{p}|\prod_{j=1}^{i}|\mathcal{X}_j||\prod_{k=1}^{i}|\mathcal{X}_k| < \sum_{i=1}^{p}\prod_{j=1}^{i}d\prod_{k=1}^{i}d$ where $d = \max_{i\in[p]}|\mathcal{X}_i|$. This sum then becomes $\sum_{i=1}^{p}d^{2i} = \frac{d^2(d^{2p}-1)}{d^2-1} = \mathcal{O}(d^{2p})$. For space complexity, in the worst case DAG mentioned, level i which has $\prod_{j=1}^{i}|\mathcal{X}_j| < d^i$ nodes and the same amount of edges coming in. For storing the stages, the extra information we need to store is the fixed contexts for each stage, and there are $\prod_{j=3}^{i}|\mathcal{X}_j| < d^i$ stages in level i, thus the nodes, edges and stages for level i are atmost $3d^i$, summing for each level gives $\sum_{i=1}^{p}3d^i = \frac{3d(3d^p-1)}{3d-1} = \mathcal{O}(d^p)$

We mention the space complexity here to emphasize that it grows exponentially, which is one limitation of this approach. For p binary variables this means a CSTree takes $\mathcal{O}(2^p)$ space. This is in comparison to DAGs which in the worst case assuming full connectivity require $\mathcal{O}(p^2)$ space, independent of the state space of the variables.

In order to learn CSI relations, one can now take a CSTree from a DAG and perform statistical CSI testing. Recall that each node in level k represents a probability density of the variable in level $k+1$ under the context fixed by that node. Thus for each level, we can compare all possible pairs of nodes by taking the samples fixed by each pair, and testing whether they are from the same distribution. If so, we assign the same colour to both of them. Then by the CSTree property from Definition 4.4 we must have that all nodes in level k which share the same context as that of these 2 nodes must also have the same colour. For example with binary variables we have 2 nodes representing the outcomes $X_{\{1,2,3,4\}} = 0110, X_{\{1,2,3,4\}} = 0011$ [6] and we know they are in the same stage S_i, then the common context for that stage is $X_{\{1,4\}} = 01$, and by the CSTree property all nodes in that level with this subcontext belong to the same stage.

[6] $X_{\{1,2,3,4\}} = 0110$ is shorthand for $(X_1 = 0, X_2 = 1, X_3 = 1, X_4 = 0)$

We now describe the algorithm for learning a CSTree. [7]

Algorithm 2: LearnCSTree
Learning a CSTree with knowledge of causal ordering

Input: CSTree T, (possibly empty) stages S, causal ordering O, Data matrix D
Output: The CSTree T with ordering O and stages S

$l = 1$;
$p = |O|$;
while $l < p$ **do**
  $ns \leftarrow$ [nodes in level l of T];
  $ps \leftarrow$ [all pairs of nodes in level l];
  for (n_1, n_2) *in* ps **do**
    if Colour(n_1)=Colour(n_2) **then**
      skip
    else
      $c \leftarrow$ CommonContext(n_1, n_2);
      same_distr = Test(c, D, $O[l+1]$);
      if *same_distr* **then**
        new_nodes $\leftarrow$ NodesWithContext(ns, c);
        $S \leftarrow$ UpdateStages(S, c, *new_nodes*);
      end
    end
  end
  $l = l + 1$;
end
return T, S

Algorithm 2 can be sped up by already providing a non-empty CSTree containing stages which we may have inferred from a DAG. If one knows the DAG and the true causal ordering of the system they can learn a CSTree by using Algorithm 1 followed by Algorithm 2.

Theorem 4.8 *Given variables $X_1, ..., X_p$ taking values in $\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i$, Algorithm 2 is correct and runs in $\mathcal{O}(d^{2p})$ time assuming constant time for statistical independence testing, where $d = \max_{i \in [p]} \mathcal{X}_i$.*

Proof: For correctness, at level loop iteration we compare all pairs of nodes and only update the stages if they do not belong to the same non-singleton stage. In this case if they do belong to the same stage according to the statistical testing, we add exactly the nodes that belong to the stage according to the CSTree property. Thus the CSTree property is intact throughout the algorithm. For time complexity, using notation from Theorem 4.7, level i has d^i nodes and in the worst case we run statistical testing on all pairs of nodes, of which there are $\binom{d^i}{2} = \frac{d^i!}{2!(d^i-2)!} < \frac{d^{2i}}{2}$, summing for each level

[7] Colour is a function that takes a node and returns the colour of it if it belongs to a non-singleton stage - note here we represent the stage using a colour; CommonContext is a function that takes 2 nodes and returns their common context; NodesWithContext takes a set of nodes and a context c and returns the nodes which have the c as a subcontext; UpdateStages is a function that updates the stages of the tree with the new nodes

gives $\sum_{i=1}^{p} \frac{d^{2i}}{2} = \mathcal{O}(d^{2p})$.

In the general case it is possible that the true causal ordering is unknown. In fact, we need to consider the set of all causal orderings for each DAG in the MEC of the true DAG. Thus we first learn the CPDAG of the true underlying DAG using the PC algorithm and then apply Algorithms 1 and 2.

Algorithm 3: CSTreePcAlgorithm
Learning a CSTree from observational data

Input: Data matrix D, (optional causal ordering O)
Output: List of CSTrees T with minimum number of stages

```
CPDAG ← PcAlgorithm(D);
if O given then
    G ← g ∈ CPDAG with ordering O;
    dags ← [G];
    orderings ← [O];
else
    dags ← [g in CPDAG];
min_stage_trees ← [];
min_stage ← ∞;
for G in dags do
    if O not given then
        orderings ← AllTopologicalSort(G);
    end
    for O in orderings do
        S, T ← DagToCSTree(G,O);
        S, T ← LearnCSTree(T, S, O, D);
        if |S| < min_stages then
            min_stages ← |S|;
            min_stage_trees ← [(T, S)];
        end
        if |S| = min_stages then
            min_stage_trees.append((T, S));
        end
    end
end
return min_stage_trees
```

We consider all topological orderings of all Markov equiavalent DAGs learnt by the PC algorithm because we might not be able to encode some context specific information otherwise. We show an example of this in the next section after introducing minimal context DAGs.

Algorithm 3 considers many possible candidate CSTree models, thus we have to pick the best model with respect to some criterion. There are however many instances where we could know the causal ordering apriori [8], for example a temporal relation between nodes known through physical laws. In this case we can either start statistical testing from an empty tree, or apply the PC algorithm to the data and find a DAG in the Markov Equivalence class with the known ordering and then run the extra testing.

[8] Peter Thwaites, Jim Q. Smith, and Eva Riccomagno. Causal analysis with chain event graphs. *Artificial Intelligence*, 174(12):889–909, 2010; and Tomi Silander and Tze-Yun Leong. A dynamic programming algorithm for learning chain event graphs. In *International Conference on Discovery Science*, pages 201–216. Springer, 2013

Unlike DAGs, as the number of variables increase it gets progessively harder to visually understand the learnt causal structure by just looking at the learnt CSTree.

4.3 Understanding high-dimensional CSTrees

From a pragmatic perspective the aim of this section is to introduce the notion of Minimal Context (MC) DAGs which can help visualize CSTrees with more variables and the context specific information they encode. On a theoretical note, this work has led to the generalization of Theorem 3.10 to define a characterization of Markov Equivalence for CSTrees [9]. We start by first describing the procedure [10] to generate the CSI relations from a CSTree and its stages, which uses Lemma 4.5

[9] Eliana Duarte and Liam Solus. Representation and learning of context-specific causal models with observational and interventional data, 2021

[10] StagesInLevel takes a set of stages and a level and returns the stages in that level; ContextOfStage takes a stage and returns the common context of that stage; VariablesOfContext takes a context and returns the variables in it

Algorithm 4: GenerateCsiRelations
Generate the CSI Relations from the CSTree

Input: CSTree T, its stages S and its causal ordering O
Output: Set of CSI Relations J encoded in the CSTree

$l = 1$;
$p = |O|$;
$J = []$;
while $l < p$ **do**
  $S_l \leftarrow$ StagesInLevel(S, l);
  for s *in* S_l **do**
    $c \leftarrow$ ContextOfStage(S);
    $v_c \leftarrow$ VariablesOfContext(c);
    $v_o \leftarrow O[1 : l-1] \backslash v_c$;
    $J.append((X_{O[l+1]} \perp\!\!\!\perp X_{v_o} \mid c))$
  end
end

Theorem 4.9 *Given a CSTree $\mathbb{T}$, Algorithm 4 is correct and returns the CSI relations encoded in $\mathbb{T}$ in $\mathcal{O}(pd^{2p})$ time.*

Correctness follows directly from Lemma 4.5 since at each loop we add

exactly the CSI relations mentioned inthe lemma, and we do this for all levels thus include all stages of the CSTree. For time complexity, for each level we first get the stages associated with it, which can be done in constant time if we store this information. We know from the Proof of Theorem 4.7 that the number of stages in level i is bounded above by d^i, and for each stage in that level we get the context of the stage, which can be done as a constant lookup operation, and we get relevant variables in Lines 8,9 which is bounded above by $2p$. Thus adding this for all the levels give $\sum_{i=1}^{p} 2pd^i = \mathcal{O}(pd^{2p})$

In practice a slightly modified version of Algorithm 4 can be placed a subroutine in the previous algorithms right before moving onto the next level.

From Lemma 4.5 we know any distribution in the CSTree model $\mathbb{M}_{(\mathbb{T},\theta)}$ encodes the CSI relations of the given form, there could be more CSI relations satisfied by every distribution in $\mathbb{M}_{(\mathbb{T},\theta)}$, which is similar to how a DAG model encodes the CI relations $\mathbb{J}_1$ implied by the local Markov property, on top of the CI relations $\mathbb{J}_2$ from the global Markov property (i.e. rom the d-separations), and $\mathbb{J}_1 \subset \mathbb{J}_2$.

We call a collection of CSI relations a Context Specific Conditional Independence model $\mathbb{J}$. The complete of set of all CSI relations satisfied by each distribution $\mathbb{P} \in \mathbb{M}_{(\mathbb{T},\theta)}$ includes the CSI relations recovered from Algorithm 4, and also include those implied by the succesive application of the Context Specific Conditional Independence axioms to generate further CSI relations. The axioms are as follows

1. Symmetry, If $(X_A \perp\!\!\!\perp X_B \,|\, X_C = x_c) \in \mathbb{J}$ then $(X_B \perp\!\!\!\perp X_A \,|\, X_c = x_c) \in \mathbb{J}$
2. Decomposition, If $(X_A \perp\!\!\!\perp X_{B\cup D} \,|\, X_S, X_C = x_c) \in \mathbb{J}$ then $(X_A \perp\!\!\!\perp X_B \,|\, X_S, X_C = x_c) \in \mathbb{J}$
3. Weak union, If $(X_A \perp\!\!\!\perp X_{B\cup D} \,|\, X_S, X_C = x_c) \in \mathbb{J}$ then $(X_A \perp\!\!\!\perp X_B \,|\, X_{S\cup D}, X_C = x_c) \in \mathbb{J}$
4. Contraction, If $(X_A \perp\!\!\!\perp X_B \,|\, X_{S\cup D}, X_C = x_c) \in \mathbb{J}$ and $(X_A \perp\!\!\!\perp X_D \,|\, X_S, X_C = x_c) \in \mathbb{J}$ then $(X_A \perp\!\!\!\perp X_{B\cup D} \,|\, X_S, X_C = x_c) \in \mathbb{J}$
5. Intersection, If $(X_A \perp\!\!\!\perp X_B \,|\, X_{S\cup D}, X_C = x_c) \in \mathbb{J}$ and $(X_A \perp\!\!\!\perp X_B \,|\, X_{B\cup D}, X_C = x_c) \in \mathbb{J}$ then $(X_A \perp\!\!\!\perp X_{B\cup S} \,|\, X_D, X_C = x_c) \in \mathbb{J}$
6. Specialization, If $(X_A \perp\!\!\!\perp X_B \,|\, X_S, X_C = x_c) \in \mathbb{J}$ and $T \subset S, x_T \in \mathcal{X}_T$ then $(X_A \perp\!\!\!\perp X_B \,|\, X_{S\setminus T}, X_{T\cup C} = x_{T\cup C}) \in \mathbb{J}$
7. Absorption, If $(X_A \perp\!\!\!\perp X_B \,|\, X_S, X_C = x_c) \in \mathbb{J}$ and $\exists T \subset C$ such that $\forall x_T \in \mathcal{X}_T$ we have $(X_A \perp\!\!\!\perp X_B \,|\, X_S, X_{C\setminus T} = x_{C\setminus T}, X_T = x_T) \in \mathbb{J}$ then $(X_A \perp\!\!\!\perp X_B \,|\, X_{S\cup T}, X_{C\setminus T} = x_{C\setminus T}) \in \mathbb{J}$

Given a Context Specific Conditional Independence model $\mathbb{J}$, the successive application of the axioms above results in the Context Specific closure of the model, denoted $\bar{\mathbb{J}}$

The Absorption axiom helps us to get a representation of the CSTree as a sequence of DAGs. For this we need the definition of minimal contexts [11].

[11] Eliana Duarte and Liam Solus. Representation and learning of context-specific causal models with observational and interventional data, 2021

Definition 4.10 (Minimal contexts) *Given a set Context Specific Independence model $\mathbb{J}$, we say that $X_C = x_C$ is a minimal context if we have $(X_A \perp\!\!\!\perp X_B \mid X_S, X_C = x_C) \in \mathbb{J}$ and there is no non-empty subset $T \subset C$ such that $(X_A \perp\!\!\!\perp X_B \mid X_{S\cup T}, X_{C\setminus T} = x_{C\setminus T}) \in \mathbb{J}$.*

Intuitively, the minimal contexts are the smallest contexts that get left behind after repeated application of the Absorption axiom. By the Specialization axiom, given a minimal context $(X_C = x_C)$ we can recover all the CI relations implied by the CSTree. We give some examples of minimal contexts below.

Example 4.11 *Let X_1, X_2, X_3, X_4, X_5 be binary variables taking values in $\{0, 1\}$. If we have just the CSI relations $(X_5 \perp\!\!\!\perp X_4 \mid X_{\{1,2,3\}} = 000)$ and $(X_5 \perp\!\!\!\perp X_4 \mid X_{\{1,2,3\}} = 1)$ they get absorbed into $(X_5 \perp\!\!\!\perp X_4 | X_{\{1\}}, X_{\{2,3\}} = 00)$ leaving the minimal context $X_{\{2,3\}} = 00$, or simply $(X_2 = 0, X_3 = 0)$*

Example 4.12 *Let X_1, X_2, X_3, X_4 be binary variables taking values in $\{0, 1\}$. If we have the CSI relations $(X_4 \perp\!\!\!\perp X_2 \mid X_{\{1,3\}} = 00)$, $(X_4 \perp\!\!\!\perp X_2 \mid X_{\{1,3\}} = 01), (X_4 \perp\!\!\!\perp X_2 \mid X_{\{1,3\}} = 10), (X_3 \perp\!\!\!\perp X_1 \mid X_2 = 1)$ then they absorb to give the equivalent CSI relations $(X_4 \perp\!\!\!\perp X_2 \mid X_1, X_3 = 0), (X_4 \perp\!\!\!\perp X_2 \mid X_3, X_1 = 0), (X_3 \perp\!\!\!\perp X_1 \mid X_2 = 0)$ thus the minimal contexts are $(X_1 = 0), (X_2 = 0), (X_3 = 0)$*

Example 4.11 shows that we can get absorption from CSI relations we get from different levels, while 4.12 shows that we have to check at least all possible pairs of CSI relations to get the minimal contexts. It is also possible to get the empty context as the minimal context, which happens when the all CSI relations $(X_A \perp\!\!\!\perp X_B \mid X_S, X_C = x_C)$ hold for all outcomes $x_C \in \mathcal{X}_C$ in which case the absorption axiom gives the CI relation $(X_A \perp\!\!\!\perp X_A \mid X_S, X_C)$.

From a computational perspective, given all CSI relations involving sets of variables X_A, X_B i.e. those of the form $(X_A \perp\!\!\!\perp X_B \mid X_S, X_C = x_C)$, we want to find the largest set $T \subset C$ such $(X_A \perp\!\!\!\perp X_B \mid X_S, X_T = x_T, x_{C\setminus T} = x_{C\setminus T})$ is also in the CSI relations. [12] This can be thought of as decomposing C into sets $C\setminus T$ and T such that Definition 4.10 holds. To find the largest such T, we can perform a binary search on the size of T, which can range from 0 to $|C|$. We describe this method below. [13]

[12] If we are to use the CSI relations extracted from Algorithm 4 before getting their Context Specific closure they would be of the form $(X_i \perp\!\!\!\perp X_B | X_C = x_C)$ where $i \in [p]$ and $B, C \subset [p-1]$

[13] Each break statement leaves the first loop it meets.

Algorithm 5: GenerateMinimalContexts
Generate the Minimal Contexts from a set of CSI relations

Input: Set of CSI Relations J
Output: Set of minimal contexts MCs from J

```
1  set_pairs ← PairsOfSets(J),MCs ← EmptyDictionary;
2  for A, B in set_pairs do
3      rels ← RelsOfPair(A, B),cs ← ContextsOfRels(rels);
4      for rel in rels do
5          C ← VariablesOfContext(ContextOfRels(rel));
6          T_sizes ← [0,...,|C|],T_found = False;
7          while not T_found do
8              mid ← |T_sizes|/2 , T_size ← T_sizes[mid];
9              T_candidates ← [subsets of C of size T_size] ;
10             for T in T_candidates do
11                 T_contained ← False;
12                 x_Ts ← CartesianProduct(T), x_Ts_count = 0;
13                 for x_T in x_Ts do
14                     x_T_contained ← True if x_T in cs else False;
15                     if x_T_contained is True then
16                         x_Ts_counts+ = 1;
17                         if x_Ts_counts = |x_Ts| then
18                             T_contained ← True;
19                             search_upperhalf ← True;
20                         end
21                     end
22                     if x_T_contained is False then
23                         search_lowerhalf ← True break
24                     end
25                 end
26                 if (T_contained and |T_candidates|=1) or
                    T_sizes=[0] or T_sizes=[|C|] then
27                     T_found ← True;
28                     MCs ← UpdateMinContexts(MCs, rels);
29                 end
30                 if search_upperhalf then
31                     T_sizes = T_sizes[mid:end] break
32                 if search_lowerhalf then
33                     T_sizes = T[start:mid] break
34                 end
35             end
36             if T_found then
37                 break
38             end
39         end
40         if T_found then
41             break
42         end
43     end
44 end
```

Theorem 4.13 *Given a set of Context Specific Independence relations* $\mathbb{J}$*, Algorithm 5 returns the set of minimal contexts in [!!! a long time theoretically]*

[!!! Describe subprocedures in minimal context generator algorithm]

In theory, the number of possible pairs of sets A, B is $\binom{2^p}{2}$, though in practice it can be reduced, for example by the Symmetry axiom allows us to half the number of pairs we need to check. This motivates us to the pairwise case, where we focus on Context Specific Independence relations of the form between 2 variables X_i, X_j rather than two sets of variables X_A, X_B. This drops the number of possible pairs to $\binom{p}{2}$. This leads us to the definition of pairwise minimal contexts.

Definition 4.14 (Pairwise minimal contexts) *Given a set Context Specific Independence model $\mathbb{J}$, we say that $X_C = x_C$ is a pairwise minimal context if we have $(X_i \perp\!\!\!\perp X_j \mid X_S, X_C = x_C) \in \mathbb{J}$ and there is no non-empty subset $T \subset C$ such that $(X_i \perp\!\!\!\perp X_j \mid X_{S \cup T}, X_C = x_C) \in \mathbb{J}$.*

We can make use of the Context Specific Conditional independence axioms to generate pairwise CSI relations to get the pairwise minimal contexts.

The introduction of pairwise relations motivates the inclusion of the following axiom.

8. Composition, If $(X_A \perp\!\!\!\perp X_B \mid X_S, X_C = x_C)$ and $(X_A \perp\!\!\!\perp X_D \mid X_S, X_C = x_C)$ then $(X_A \perp\!\!\!\perp X_{B \cup D} \mid X_S, X_C = x_C)$

The composition axiom allows us to go from pairwise relations between variables to the more general pairwise relations between sets of variables. We call a Context Specific Conditional Independence model satisfying the additional composition axiom a Context Specific compostional graphoid, which generalizes the notion of compositional graphoids [14]. An important question is whether the Context Specific closure of CSI relations from a CSTree form a compositional context specific graphoid.

[14] Kayvan Sadeghi and Steffen Lauritzen. Markov Properties for Mixed graphs. *Bernoulli*, 20(2):676 – 696, 2014

Pairwise minimal contexts are always minimal contexts however but we could have minimal contexts that are not pairwise minimal contexts. If all CSTrees have an associated compositional context specific independence model, the pairwise relations may indeed be all that we need to get the set of complete set of minimal contexts. We leave this as an open question and use pairwise minimal contexts to offer a visualization of CSI relations in the CSTree which may potentially be incomplete.

Denote $\mathbb{C}(\mathbb{T})$ as the set of all minimal contexts of a CSTree $\mathbb{T}$, and denote $\mathbb{G}(\mathbb{T}) := \{\mathbb{G}_{X_C = x_C}\}_{X_C = x_C \in \mathbb{C}(\mathbb{T})}$ as the set of all minimal context DAGs of $\mathbb{T}$.

We now have the machinery to visualize higher dimensional CSTrees.We start by generating the CSI relations from the trees. Then get the Context Specific closure, and then get the minimal contexts. Once we have the minimal contexts, the CI relations for each minimal context define a graphoid. This is called a Minimal Context DAG. Thus, the CSTree can be represented as a sequence of DAGs for each minimal context. This representation is a consequence of the following theorem [15].

[15] Eliana Duarte and Liam Solus. Representation and learning of context-specific causal models with observational and interventional data, 2021

Theorem 4.15 (Markov theorem for CSTrees) *Given a CSTree $\mathbb{T}$, with levels $L_1, ..., L_p \sim X_1, ..., X_p$, minimal contexts $\mathbb{C}(\mathbb{T})$ and $\mathbb{P} \in \Delta_{\mathcal{X}-1}$. The following are equivalent.*

- $\mathbb{P}$ *factorizes according to* $\mathbb{T}$.
- $\mathbb{P}$ *is Markov to* $\mathbb{G}(\mathbb{T})$.
- $\forall\, X_C = x_C \in \mathbb{C}(\mathbb{T})$ *we have* $\mathbb{P}(X_{[p]\setminus C} \mid X_C = x_C) = \prod_{k\in[p]\setminus C} \mathbb{P}(X_k \mid X_{PA_{\mathbb{G}_{X_C=x_C}}(k)}, X_C = x_C)$.

This DAG is also called the minimal I-MAP [16], which we can recover with the following procedure [17].

[16] Thomas Verma and Judea Pearl. *Causal Networks: Semantics and Expressiveness. "Graphoids: A Computer Representation for Dependencies and Relevance in Automated Reasoning (Computer Information Science).",* pages 69–76. Uncertainty in Artificial Intelligence. Elsevier, 1990

[17] L Solus, Y Wang, and C Uhler. Consistency Guarantees for Greedy Permutation-Based Causal Inference Algorithms. *Biometrika*, 01 2021. asaa104

Algorithm 6: GENERATEMINCONTEXTDAGS
Generating minimal context DAGs

Input: Causal ordering O, Minimal Contexts *MCs* as a dictionary with minimal contexts as keys and the CI relations under the minimal context as values
Output: List of minimal contexts with their minimal context DAGs *MCDAGS*

```
MCDAGS ← [];
for MC, Ci_Rels in MCs do
    G ← Empty Graph;
    nodes ← O\ VariablesOfContext(MC);
    for i in [1,...,|nodes|] do
        for j in [i+1,...,|nodes|] do
            π_i ← nodes[i];
            π_j ← nodes[j];
            G.add_edge(π_i, π_j);
            S = O[1:j−1]\VariablesOfContext(MC) \{π_i} ;
            if (X_{π_i} ⫫ X_{π_j} | X_S) ∈ Ci_Rels then
                G.remove_edge(π_i, π_j)
            end
        end
    end
    MCDAGS.add((MC, G));
end
return MCDAGS
```

Theorem 4.16 *Given variables $X_1, ..., X_p$, a set of minimal contexts C and for each minimal context C the conditional independence relations $\mathbb{J}_C$ that hold under this minimal context, Algorithm 6 is correct and runs in $\mathcal{O}(p^2|C||\mathbb{J}|)$ time.*

We end this section by stating the definition of faithfulness for

CSTrees which allows us to explain why we consider all Markov equivalent DAGs when learning a CSTree, followed by the main theorem for Algorithm 3

Definition 4.17 (Faithfulness for CSTrees) *A distribution $\mathbb{P}$ is faithful to a CSTree $\mathbb{T}$ if it entails exactly the CSI relations encoded by the set of minimal context DAGs $\mathbb{G}(\mathbb{T})$.*

To see why we need to consider all Markov equivalent DAGs in Algorithm 3, suppose we have the following case with 4 binary variables.

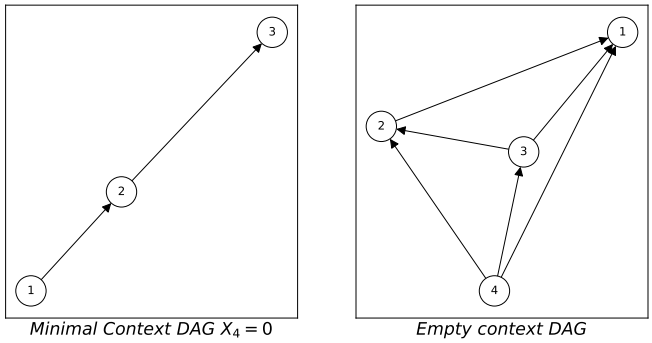

Figure 4.2: Example on why to consider all topological ordering of all Markov equivalent DAGs when learning a CSTree

Let $\mathbb{P}$ be the data generating distribution and suppose it is faithful to the CSTree with the context graph shown above for the context $X_4 = 0$. The empty context DAG could possibly be the fully connected shown above. If we do happen to learn this DAG from the PC algorithm step, it only has one topological ordering which is 4321, and there is no way to join the stages for the empty context graph which this order to encode $(X_1 \perp\!\!\!\perp X_3 \,|\, X_4 = 0)$, in which case we do not learn the true model.

Theorem 4.18 *Given variables $X_1, ..., X_p$, assuming that the data generating distribution is faithful to some unknown CSTree $\mathbb{T}$ with a known causal ordering $\pi = \pi_1 \cdots \pi_p$, Algorithm 3 is consistent, i.e. it recovers $\mathbb{T}$ as the number of samples $n \to \infty$.*

5

Experiments

5.1 Encoding DAGs as CSTrees

We first run a sanity check on generating the CSTrees from DAGs in the 2 extreme cases, when we have no connectivity and full connectivity.

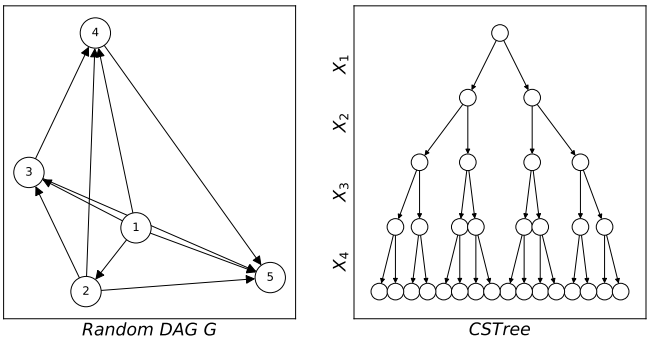

Figure 5.1: Generating the CSTrees for a fully connected DAG using the ordering 1234 with Algorithm 1

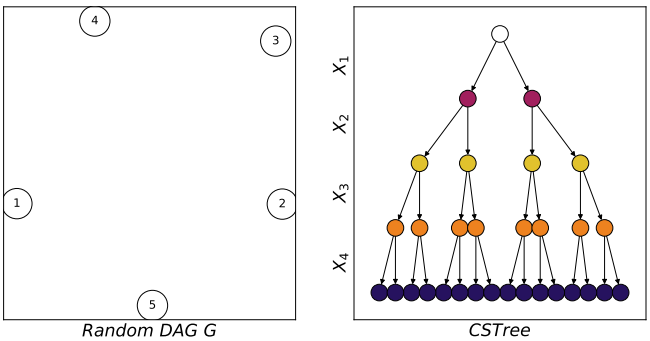

Figure 5.2: Generating the CSTrees for DAG with no edges using the ordering 1234 with Algorithm 1

The results are as expected since there are no conditional independence stations encoded in the fully connected, and no context specific independence statements since they are unable to encode them, meaning no 2 nodes share a fixed context and all stages have a single element. For the DAG with no edges, each variable is independent of each other, and each node has no parents thus by Proposition **??**

5.2 Recovering the empty context DAG

We start by generating random DAGs and generating the CSTrees as per Algorithm 1. This is done by first choosing the number of variables (nodes in the DAG) p, and then choosing an edge with probability $p_{edge} \in (0,1)$, and then keeping edge (u,v) if and only if $u < v$. We show the generated CSTree below, and recover the original DAG as a minimal context DAG with the empty context with Algorithm 6.

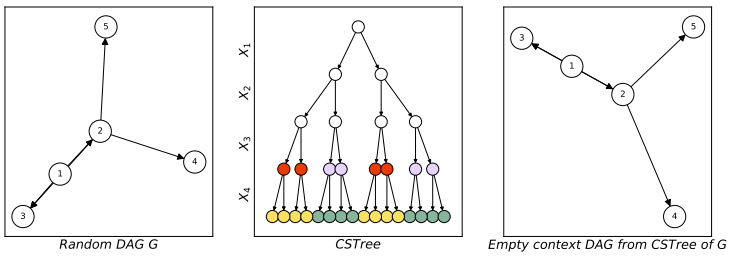

Figure 5.3: Generating CSTrees from random DAGs and recovering the original random DAG from the CSTree with ordering 1234 using Algorithm 6

We show a CSTree for a random DAG model with 8 binary variables below in a more compact visualization, alongside the recovered empty context DAG in Figures 5.45.5.

p_{edge}	Nodes	Space (GB)	Time (s)
0.2	20	1.203	14
0.2	21	2.422	43
0.2	22	4.875	74
0.2	23	9.812	240

Table 5.1: Statistics from generating CSTrees for random DAGs, Space refers to the amount of RAM occuppied by the CSTree structure, and time refers to the amount of time taking to encode the CI relations in the DAG to the CSTree.

5.3 Synthetic data

Next we generate a random DAG $\mathcal{G}$ using the procedure above, and then generate samples from this DAG. We then discard the DAG and apply Algorithm 3. We let all variables be binary valued, and for each variable X_i, generate conditional probability distribution tables when conditioned on the all possible outcomes of the parents of X_i. We give variables without parents a Bernoulli distribution with a randomly chosen parameter for each such variable. To introduce a

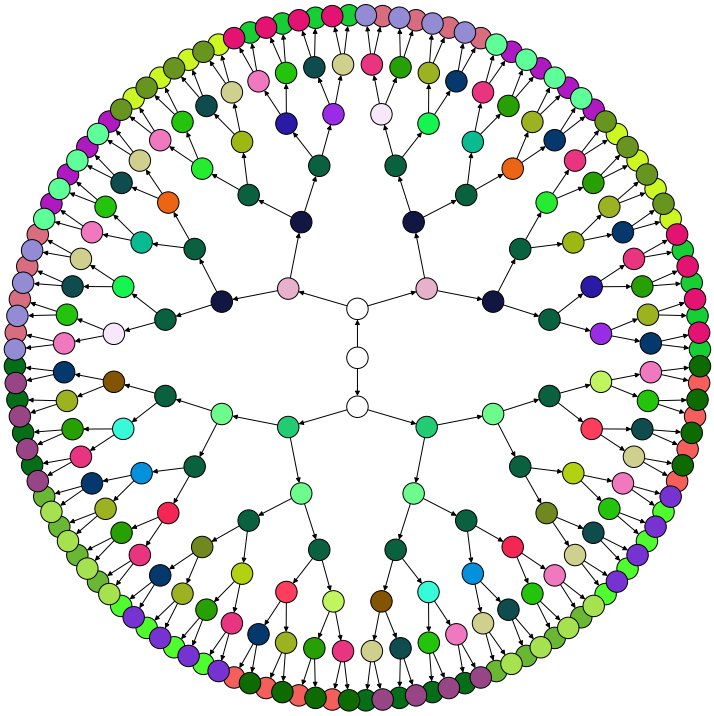

Figure 5.4: CSTree corresponding to the 8 node DAG in Figure 5.5

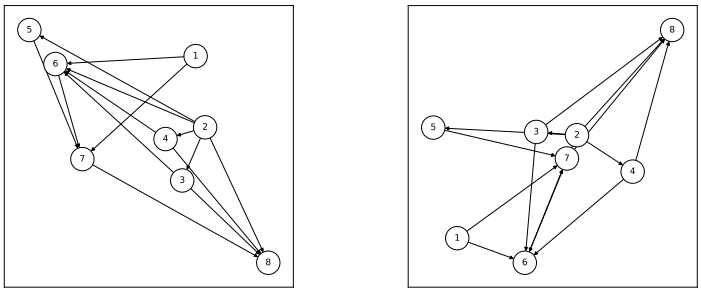

Figure 5.5: Left: A random DAG with 10 binary variables which represents the CSTree in Figure 5.4. Right: The corresponding empty context DAG from that CSTree

causal element, we divide the rows of each such table into 2 halves, and for the all the outcomes of the first half we assign a higher probability for X_i to take 0, and for the second half we assign a higher probability for X_i to be 1. We then normalize probabilities such that $\mathbb{P}(X_i = 1|X_{Pa_{\mathcal{G}(i)}} = x) + \mathbb{P}(X_i = 0|X_{Pa_{\mathcal{G}(i)}} = x) = 1$ for each possible outcome x of the parent set.

5.4 Empirical study of algorithms

Here we look at some more general empirical results which we observed throughout our experiments.

- A high ratio of tests are skipped in practice
- As we moving up the levels, the number of samples to run the independence tests decrease rapidly, in which case we assume they are dependent
- The staging can be highly sensitive to the independence testing being done. We use the Epps and Singleton test [1] and the Anderson-Darling test [2].

[1] T.W. Epps and Kenneth J. Singleton. An omnibus test for the two-sample problem using the empirical characteristic function. *Journal of Statistical Computation and Simulation*, 26(3-4):177–203, 1986; and Sebastian J. Goerg and Johannes Kaiser. Nonparametric testing of distributions-the epps-singleton two-sample test using the empirical characteristic function. *The Stata Journal*, 9(3):454–465, 2009

[2] F. W. Scholz and M. A. Stephens. K-sample anderson-darling tests. *Journal of the American Statistical Association*, 82(399):918–924, 1987

6

Bibliography

[1] David Hume. *Enquiry Concerning Human Understanding*. Clarendon Press, 1904.

[2] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018.

[3] Rodrigo A Collazo, Christiane Görgen, and Jim Q Smith. *Chain event graphs*. CRC Press, 2018.

[4] Dan Geiger and David Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82(1-2):45–74, 1996.

[5] Nir Friedman and Moises Goldszmidt. Learning bayesian networks with local structure. In *Learning in graphical models*, pages 421–459. Springer, 1998.

[6] Eliana Duarte and Liam Solus. Representation and learning of context-specific causal models with observational and interventional data, 2021.

[7] Biwei Huang, Kun Zhang, Mingming Gong, and Clark Glymour. Causal discovery and forecasting in nonstationary environments with state-space models, 2019.

[8] Pengfei Hu, Rong Jiao, Li Jin, and Momiao Xiong. Application of causal inference to genomic analysis: Advances in methodology. *Frontiers in Genetics*, 9:238, 2018.

[9] Jakob Runge, Sebastian Bathiany, Erik Bollt, Gustau Camps-Valls, Dim Coumou, Ethan Deyle, Clark Glymour, Marlene Kretschmer, Miguel D Mahecha, Jordi Muñoz-Marí, et al. Inferring causation from time series in earth system sciences. *Nature communications*, 10(1):1–13, 2019.

[10] Duligur Ibeling Thomas Icard Elias Bareinboim, Juan D. Correa. On pearl's hierarchy and the foundations of causal inference, 2020.

[11] R.A. Fisher. *The design of experiments. 1935*. Oliver and Boyd, Edinburgh, 1935.

[12] Bernhard Schölkopf. Causality for machine learning, 2019.

[13] Edward H Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(2):238–241, 1951.

[14] Clifford H. Wagner. Simpson's paradox in real life. *The American Statistician*, 36(1):46–48, 1982.

[15] David H. Wolpert. What is important about the no free lunch theorems?, 2020.

[16] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10(nil):nil, 2019.

[17] Eliana Duarte and Liam Solus. Algebraic geometry of discrete interventional models, 2020.

[18] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[19] Christopher Meek. Strong completeness and faithfulness in bayesian networks. *arXiv preprint arXiv:1302.4973*, 2013.

[20] Tom S. Verma and Judea Pearl. On the equivalence of causal models. *CoRR*, abs/1304.1108, 2013.

[21] Caroline Uhler, Garvesh Raskutti, Peter Bühlmann, and Bin Yu. Geometry of the Faithfulness Assumption in Causal inference. *The Annals of Statistics*, 41(2):436 – 463, 2013.

[22] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2nd edition, 2000.

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

[24] Christopher Meek. Causal inference and causal explanation with background knowledge, 2013.

[25] Peter Thwaites, Jim Q. Smith, and Eva Riccomagno. Causal analysis with chain event graphs. *Artificial Intelligence*, 174(12):889–909, 2010.

[26] Tomi Silander and Tze-Yun Leong. A dynamic programming algorithm for learning chain event graphs. In *International Conference on Discovery Science*, pages 201–216. Springer, 2013.

[27] Kayvan Sadeghi and Steffen Lauritzen. Markov Properties for Mixed graphs. *Bernoulli*, 20(2):676 – 696, 2014.

[28] Thomas Verma and Judea Pearl. *Causal Networks: Semantics and Expressiveness. "Graphoids: A Computer Representation for Dependencies and Relevance in Automated Reasoning (Computer Information Science)."*, pages 69–76. Uncertainty in Artificial Intelligence. Elsevier, 1990.

[29] L Solus, Y Wang, and C Uhler. Consistency Guarantees for Greedy Permutation-Based Causal Inference Algorithms. *Biometrika*, 01 2021. asaa104.

[30] T.W. Epps and Kenneth J. Singleton. An omnibus test for the two-sample problem using the empirical characteristic function. *Journal of Statistical Computation and Simulation*, 26(3-4):177–203, 1986.

[31] Sebastian J. Goerg and Johannes Kaiser. Nonparametric testing of distributions-the epps-singleton two-sample test using the empirical characteristic function. *The Stata Journal*, 9(3):454–465, 2009.

[32] F. W. Scholz and M. A. Stephens. K-sample anderson-darling tests. *Journal of the American Statistical Association*, 82(399):918–924, 1987.

7
Appendix A

7.1 Remarks on implementation

Implementation of this work was done in Python. Draft code repository is available at github.com/mnazaal/masters-thesis. This document was generated with org-mode.