MAY 14, 2021

Abstract

Despite having a philosophical grounding from empiricism that span some centuries, the algorithmization of causal discovery started only a few decades ago. 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. 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, which can encode CSI relations.

Sammanfattning

Trots en filosofisk grund från empirism som sträcker sig över några århundraden, började algoritmiseringen av kausal upptäckt bara för några decennier sedan. Denna formalisering av att studera orsakssamband beror på samband mellan grafer och sannolikhetsfördelningar. I den här inställningen är kausal upptäckt att återställa den graf som bäst beskriver kausalstrukturen baserat på de data vi får. En särskild klass av kausala upptäcktsalgoritmer, kallade begränsningsbaserade metoder, är beroende av Directed Acylic Graphs (DAG) som en kodning av förhållanden med villkorlig självständighet (CI) som bär någon nivå av kausal information. En CI-relation som X och Y är oberoende förutsatt att Z förutsätter att självständigheten gäller för alla möjliga värden Z kan ta, vilket kan vara orealistiskt i praktiken där orsakssamband ofta är sammanhangsspecifikt. I denna avhandling strävar vi efter att utveckla begränsningsbaserade algoritmer för att lära kausalstruktur från Contex-Specific Independence (CSI) -relationer inom den diskreta miljön, där självständighetsrelationerna har formen X och Y är oberoende av $Z = a$ för några a. Detta görs med hjälp av sammanhangsspecifika träd, eller kort CStrees, som kan koda CSI-relationer.

Acknowledgements

I would really like to thank Liam for providing me the opportunity to work on a very exciting project, and for being a superb supervisor in every way imaginable. His guidance is without a doubt the secret ingredient that went into this work. A special thanks to George for being on standby for small discussions, and of course to Nikos and Cindy for the virtual study sessions. Also a special thanks to my family, especially my mom, for valuing a good education back when I was younger, and their patience now that I am maybe studying a bit too much. Last but not least, I thank KTH for funding my curiosity.

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. 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] Huang, B., Zhang, K., Gong, M., and Glymour, C. (2019). Causal discovery and forecasting in nonstationary environments with state-space models

[2] Hu, P., Jiao, R., Jin, L., and Xiong, M. (2018). Application of causal inference to genomic analysis: Advances in methodology. *Frontiers in Genetics*, 9:238

[3] Runge, J., Bathiany, S., Bollt, E., Camps-Valls, G., Coumou, D., Deyle, E., Glymour, C., Kretschmer, M., Mahecha, M. D., Muñoz-Marí, J., et al. (2019). Inferring causation from time series in earth system sciences. *Nature communications*, 10(1):1–13

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

[5] Elias Bareinboim, Juan D. Correa, D. I. T. I. (2020). On pearl's hierarchy and the foundations of causal inference

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 (or treatment), 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] Fisher, R. (1935). *The design of experiments. 1935*. Oliver and Boyd, Edinburgh

whether smoking causes lung cancer which would involve forcing the subjects to smoke. This creates a need for more sophisticated methods, namely causal models, to learn causal relationships from both observational and interventional data. In fact one can even use such models to explain the success of RCTs in a formal manner.

1.3 Context specific causal discovery

Much of the causal models in wide use today make use of Directed Acyclic Graphs (DAGs), with the semantics that the directed edges represent causal relationships between the variables, which correspond to the nodes in the graph. As we operate under the assumption that any data we receive is generated by some underlying probability distribution, we relate these graphs to them via conditional independence relations which carry elementary causal information about the system of variables. Many algorithms have been proposed to learn such causal models from observed data in recent decades [7], and are in wide use across various domains. They are, however, restrictive in their modelling capacity, since a conditional independence relation by definition is assumed to hold over all possible outcomes of the conditioning set. We call these outcomes contexts, and it is realistic to assume that causal relations depend on context specific information. A number of approaches exist to model such context specific information [8], however their representations are not as intuitive as DAGs. Recently, a new representation for context specific causal relations have been proposed, called Context Specific trees (CStrees), which can be seen as a natural generalization of DAGs to the context specific scenario. Namely, CStrees allow for the representation of context specific information as a sequence of DAGs, thus striking a balance between modelling capacity and an intuitive representation. The next natural step is to learn CStrees from data, and this thesis provides the first novel causal discovery algorithms for CStrees to learn from observational data, alongside algorithms to compute the required minimal contexts, which are required for representing CStrees as a sequence of DAGs. This approach is a constraint based approach which generalizes the classic PC algorithm [9], alongside applications to both synthetic and real world data.

[7] Chickering, D. M. (2002b). Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554; Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72; Solus, L., Wang, Y., and Uhler, C. (2021). Consistency Guarantees for Greedy Permutation-Based Causal Inference Algorithms. *Biometrika*. asaa104; and Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78

[8] Collazo, R. A., Görgen, C., and Smith, J. Q. (2018). *Chain event graphs*. CRC Press; Silander, T. and Leong, T.-Y. (2013). A dynamic programming algorithm for learning chain event graphs. In *International Conference on Discovery Science*, pages 201–216. Springer; Thwaites, P., Smith, J. Q., and Riccomagno, E. (2010). Causal analysis with chain event graphs. *Artificial Intelligence*, 174(12):889–909; and Görgen, C. and Smith, J. Q. (2017). Equivalence classes of staged trees

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

1.4 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 is set in the assumption of independent and identically distributed (i.i.d) data, and ignores information from interventions, domain shifts and temporal structure [10]. 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 [11], such as Simpsons paradox [12], where one might for example have a positive correlation between 2 variables over the whole data, but dividing the samples into further groups would result in a negative correlation within each group. This is not just a theoretical issue, and has been reported in many real life data as well [13].

[10] Schölkopf, B. (2019). Causality for machine learning

[11] Pearl, J. and Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition

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

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

2

Causal Discovery with Directed Acyclic Graphs

2.1 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. 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 states that over a uniform distribution over search/learning problems (which includes causal discovery), all algorithms for such problems have equal performance.

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

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 remove connections based on statistical independence information 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

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

Bayesian Information Criterion (BIC). In this thesis we will mainly be concerned with constraint based methods, particularly in the discrete setting, where we assume the variables in the system can only take discrete values.

2.2 Direct Acyclic Graphs (DAGs)

We now cover some important definitions and concepts related to Directed Acyclic Graphs (DAGs). They are a convenient and informative graphical means of visualizing the direct cause-effect relationships between variables in a system, and the de-facto choice to model causal structures.

Definition 2.1 (DAGs) *A Directed Acyclic Graphic (DAG) is a directed graph* $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ *which has no cycles.*

Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be a graph. We then have the following definitions. A node $u \in \mathbb{V}$ is a **parent** of another node $u \in \mathbb{V}$ if $(u, v) \in \mathbb{E}$, in this case we also say v is a **child** of u. If there is an edge $(u, v) \in \mathbb{E}$ or $(v, u) \in \mathbb{E}$ we say the nodes u,v are **adjacent**. A set of nodes $(u_1, ..., u_k)$, $k \geq 2$ such that $(u_i, u_{i+1}) \in \mathbb{E}$ is called a path between u_1 and u_k. This in case, we say u_1 is an **ancestor** of u_k and u_k is a **descendant** of u_1. If we have a node v that is not a descendant of a node u we say v is a **non-descendant** of u.

For any node $u \in \mathbb{V}$ we denote $PA_{\mathbb{G}}(u)$, $CH_{\mathbb{G}}(u)$, $DS_{\mathbb{G}}(u)$, $ND_{\mathbb{G}}(u)$ to be set of parents, children, descendants and non-descendants of u respectively.

Throughout this work we let $[p] = \{1, ..., p\}$ be the set of nodes.

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 2.2 (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.

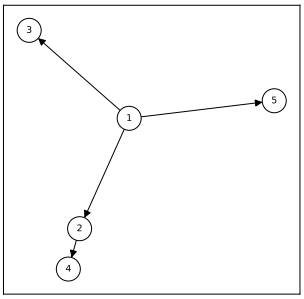


Figure 2.1: Example of a DAG $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with $\mathbb{V} = \{1,2,3,4,5\}$ and $\mathbb{E} = \{(1,2),(1,3),(1,5),(2,4)\}$. Here $PA_{\mathbb{G}}(2) = \{1\}$, $DS_{\mathbb{G}}(2) = CH_{\mathbb{G}}(2) = \{1\}$, $ND_{\mathbb{G}}(2) = \{3,5\}$

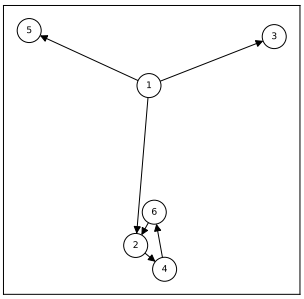


Figure 2.2: This graph is not a DAG since there is a cycle

Definition 2.3 (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 conditionally independent given S, (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 observed 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 distributions to DAGs based on the CI relations encoded by them. As the CI relations have a natural causal interpretation, the local Markov property provides a foundation to relate data generating distributions to DAG representations of a causal system.

Definition 2.4 (Local Markov property) *Let $\mathbb{G}$ 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. [3]

Definition 2.5 (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 $(X_i \perp\!\!\!\perp X_{\{1,...,i-1\} \setminus PA_{\mathbb{G}}(i)} \mid X_{PA_{\mathbb{G}}(i)})$*

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

A distribution $\mathbb{P}$ satisfying the local Markov property with respect to a DAG $\mathbb{G}$ is equivalent to that distribution also satisfying the ordered Markov property with respect to $\mathbb{G}$ and a linear ordering of $\mathbb{G}$.

An important notion in DAGs is that of d-separation and blocked paths. [4] .

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

Definition 2.6 (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 $x \to s \to y$, $x \leftarrow s \leftarrow y$, or $x \leftarrow s \to y$ with $s \in S$*
- *Along the path there is a triple of nodes (x, s, y) such that $x \to s \leftarrow y$ such that $s \notin S$ and no descendants of s are in S.*

Definition 2.7 (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.

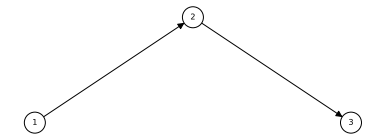


Figure 2.3: Chain

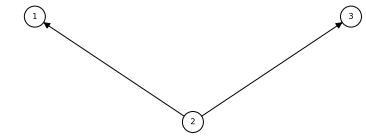


Figure 2.4: Fork/Common cause

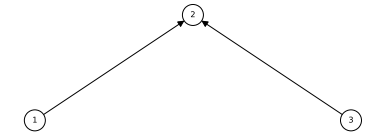


Figure 2.5: V-structure/ Collider/Immorality

Definition 2.8 (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 [5].

[5] Duarte, E. and Solus, L. (2020). Algebraic geometry of discrete interventional models

Theorem 2.9 (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 respect to $\mathbb{G}$ 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 $\mathbb{P}$ [6].

[6] Lauritzen, S. L. (1996). *Graphical models*, volume 17. Clarendon Press

Thus DAGs can be seen as structures that encode 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 [7].

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 [8].

Theorem 2.10 (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 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.

2.3 Causal Discovery Algorithms for DAGs

Theorem 2.9 suggests that we can make use of CI testing on a distribution $\mathbb{P}$ to learn a DAG $\mathbb{G}$. However, the distribution $\mathbb{P}$ may contain CI relations not encoded in the DAG, thus we make the following assumption.

Definition 2.11 (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, the global Markov property holds both ways. It should be noted that faithful distributions exist [9], and the set of distributions that are not faithful to a dag $\mathbb{G}$ have measure 0 [10], which suggets that in theory this is not a very restrictive assumption.

One of the first practical algorithms which make use of the theory above is the PC algorithm, [11] which is a constraint based causal discovery algorithm that relies of the characterization of DAGs in Theorem 2.10 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) [12], 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 a is Complete PDAG

[7] Meek, C. (2013b). Strong completeness and faithfulness in bayesian networks. *arXiv preprint arXiv:1302.4973*; and

[8] Verma, T. S. and Pearl, J. (2013). On the equivalence of causal models. *CoRR*, abs/1304.1108

[9] Meek, C. (2013b). Strong completeness and faithfulness in bayesian networks. *arXiv preprint arXiv:1302.4973*

[10] Uhler, C., Raskutti, G., Bühlmann, P., and Yu, B. (2013). Geometry of the Faithfulness Assumption in Causal inference. *The Annals of Statistics*, 41(2):436 – 463

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

[12] Meek, C. (2013a). Causal inference and causal explanation with background knowledge

(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. CPDAGs are also sometimes called essential graphs [13].

[13] Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). A characterization of markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2)

2.4 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 2.12 (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 $(X_A \perp\!\!\!\perp_{\mathbb{P}} X_B \mid X_S, X_C = x_C)$) 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.

3

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 good way to approach the problem of encoding context information as well.

3.1 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] 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 3.1 (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 3.2 (Staged tree models) *The staged tree model* $\mathbb{M}_{(\mathbb{T},\theta)}$ *is the image of the map* $\varphi_{\mathbb{T}} : x \to f_v := \left(\prod_{e \in E_{\mathbb{T}(\lambda(v))}} x_{\theta(e)} \right)_{v \in \mathbf{i}_{\mathbb{T}}}$

Thus given variables $X_1, ..., X_p$, a causal ordering π, the staged tree for this with levels [2] $L_1, ..., L_p \sim X_{\pi_1}, ..., X_{\pi_p}$, each path from the root to the leaf defines a sequence of events $x_1, x_1x_2, ..., x_1 \cdots x_p$ where $x_i \in \mathcal{X}_{\pi_i}$. Since for the edge $e = ((x_1 \cdots x_k), (x_1 \cdots x_k x_{k+1}))$ we have $x_{\theta(e)} = \mathbb{P}(x_{k+1} \mid x_1 \cdots x_k)$, the product in Definition 3.2 does indeed result in $\mathbb{P}(v_1, ..., v_p)$ for each $v \in \mathbf{i}_{\mathbb{T}}$ by the chain rule in probability. [3]

The important characteristic of staged trees are the stages.

Definition 3.3 (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 are perhaps too general, in the sense that despite allowing for the representation of context specific information, they do not admit a intuitive representation of the causal structure. This creates the need for a structure that generalizes DAG models **and** admits an intuitive representation. The recently proposed subclass of staged trees, known as CStrees allow for this.

Definition 3.4 (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.* $\forall x_{\pi_k} \in \mathcal{X}_{\pi_k}$ *we have* $\theta(x_{\pi_1} ... x_{\pi_{k-1}} \to x_{\pi_{k-1}} x_{\pi_k}) = \theta(y_{\pi_1} ... y_{\pi_{k-1}} \to y_{\pi_{k-1}} x_{\pi_k})$ *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* $x_{C_i} \in \mathcal{X}_{C_i}$*, where the stages contain nodes generated from taking 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 $P(x_k | x_1 \cdots x_{k-1})$, which is also the value of the parameter associated to this edge. 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 $(X_{\pi_1} = a_1, ..., X_{\pi_k} =$

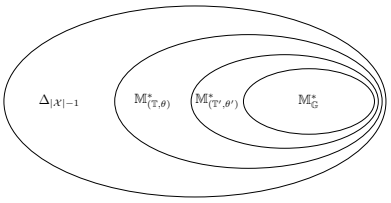


Figure 3.1: Hierarchy of models on $\mathcal{X}$ that we are concerned with, the most general space of models being probability simplex since it contains all distributions on $\mathcal{X}$, followed by the space of staged tree models $\mathbb{M}^*_{(\mathbb{T},\theta)}$, the space of Cstree models $\mathbb{M}^*_{(\mathbb{T}',\theta)}$ then DAG models $\mathbb{M}^*_{\mathbb{G}}$. More general models can explain more datasets whilst simpler models can often be easier to work with.

[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.

[3] The chain rule in probability states $\mathbb{P}(X_1, ..., X_p) = \mathbb{P}(X_p \mid X_{p-1}, ..., X_1)\mathbb{P}(X_{p-1} \mid X_{p-2}, ..., X_1) \cdots \mathbb{P}(X_2|X_1)\mathbb{P}(X_1)$

a_k). Thus, we can view this node n as representing 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.

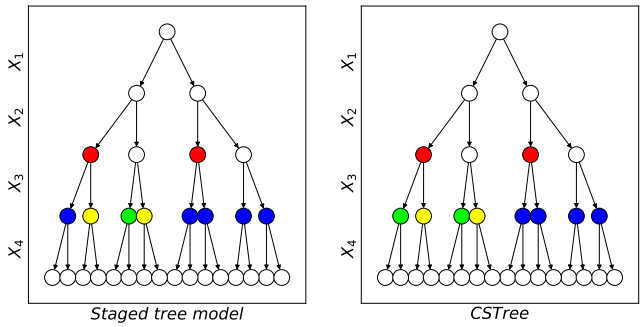


Figure 3.2: Example of a staged tree model that is not a CStree (Left) and a CStree (right) for binary variables X_1, X_2, X_3, X_4 in that causal ordering.

Both staged trees in Figure 3.2 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 - this is however not the case since there are nodes in level 3 which correspond to the yellow and green stages, thus not part of the blue stage.

A CStree encodes Context Specific Independence (CSI) relations according to the following lemma [4]

Lemma 3.5 (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 $X_{C_i} = x_{C_i}$ is the context fixed by the stage S_i.*

[4] Duarte, E. and Solus, L. (2021). Representation of context-specific causal models with observational and interventional data

The CSI relations from the CStrees look similar to the CI relations from the ordered Markov property. The difference is that the CStree encodes independence of X_k with all the variables preceding it in the causal ordering when conditioned on a context, compared to the ordered Markov property which contains the variables which represent the parents of k in the conditioning set. Thus, CStrees can be thought of as a relaxation of DAG models via a relaxation of the ordered Markov property, where we condition on the more general scenario of contexts, rather than variables.

One question that arises is how many CStrees are there for a given system of variables $X_1, ..., X_p$ taking values in $\mathcal{X}_1, ..., \mathcal{X}_p$. The authors of the original CStree paper provide a closed form expression for the number of CStrees on p binary variables, alongside a table comparing the number of DAGs (sequence A003024 in oeis.org), CStrees and comptabily labeled staged trees, which we include below in Table 3.1. A closed form expression for general CStrees remains to be discovered.

p	DAGs	CStrees	Compatibly labeled staged trees
1	1	1	1
2	3	4	4
3	25	96	180
4	543	59136	2980800
5	29281	26466908160	15619603855888000

Table 3.1: Number of DAGs, CStrees and compatibly labeled staged trees on p binary variables.

3.2 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 [5].

[5] Duarte, E. and Solus, L. (2020). Algebraic geometry of discrete interventional models

Proposition 3.6 (CStrees corresponding to DAGs) *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 $\mathbb{G}$. [6] .

[6] 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 ← Empty staged tree with ordering O;
S ← Empty List;
for k in |O| − 1 do
    v ← O[k + 1] ;
    T.add_level(v);
    pars ← Parents(G, v);
    contexts ← CartesianProduct(pars);
    stages_k ← Empty Dictionary;
    for c in contexts do
        stages_k[c] ← [nodes in level k where c is a subcontext];
    end
    S.append(stages_k);
end
return T, S
```

Algorithm 1 above does not necessarily need a causal ordering. This is because given a DAG we can perform a topological sort on it to get one, for which efficient algorithms exist [7].

[7] Tarjan, R. E. (1976). Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–185

Theorem 3.7 *Given variables $X_1, ..., X_p$ taking values in $\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i$, Algorithm 1 is correct i.e. it takes a DAG and returns the corresponding CStree 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 3.6 the tree is still a CStree. Since the staging process at each level only creates non-singleton stages of nodes within that level, and we go over each level except the last level which always contain singleton stages (one for each outcome of $\mathcal{X}$), 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 $12 \cdots p$, 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 contexts 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 at most* $3d^i$, *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 a statistical test to determine context specific independence relations. 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 the contexts of the 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 3.4 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 if we have 2 nodes representing the outcomes $X_{\{1,2,3,4\}} = 0110, X_{\{1,2,3,4\}} = 0011$ [8] 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.

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

We now describe the algorithm for learning a CStree. [9]

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, n_1, n_2, 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

[9] 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 - if one or both of them already belong to a stage, we take this to be the common context between these contexts; Test is a function that determines whether the distributions corresponding to both of the nodes belong to the same stage or not - this typically involves a statistical test; 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.

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 3.8 *Given variables $X_1, ..., X_p$ taking values in $\mathcal{X} = \prod_{i=1}^{p} \mathcal{X}_i$, Algorithm 2 is correct i.e. it merges all possible stages whilst maintaining the CStree property 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 3.7, level i has at most 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 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
        T, S ← DagToCStree(G,O);
        T, S ← 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 [10], 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.

[10] Thwaites, P., Smith, J. Q., and Riccomagno, E. (2010). Causal analysis with chain event graphs. *Artificial Intelligence*, 174(12):889–909; and Silander, T. and Leong, T.-Y. (2013). A dynamic programming algorithm for learning chain event graphs. In *International Conference on Discovery Science*, pages 201–216. Springer

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.

3.3 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 2.10 to define a characterization of Markov Equivalence for CStrees [11]. We start by first describing the procedure [12] to generate the CSI relations from a CStree and its stages, which uses Lemma 3.5

[11] Duarte, E. and Solus, L. (2021). Representation of context-specific causal models with observational and interventional data

[12] 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 ← StagesInLevel(S, l);
    for s in S_l do
        c ← ContextOfStage(S);
        v_c ← VariablesOfContext(c);
        v_o ← O[1 : l − 1]\v_c;
        J.append((X_{O[l+1]} ⫫ X_{v_o} | c))
    end
end
```

Theorem 3.9 *Given a CStree $\mathbb{T}$, Algorithm 4 is correct and returns the CSI relations encoded in $\mathbb{T}$ in $\mathcal{O}(pd^{2p})$ time.*

Proof: Correctness follows directly from Lemma 3.5 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 3.7 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 as a subroutine in the previous algorithms right before moving onto the next level.

From Lemma 3.5 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, which are captured by the CI relations $\mathbb{J}_2$ from the global Markov property (i.e. from the d-separations), and $\mathbb{J}_1 \subset \mathbb{J}_2$.

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 $(X_A \perp\!\!\!\perp X_B \mid X_S, X_C = x_c) \in \mathbb{J}$ and $\exists T \subset C$ such that $\forall x_T \in \mathcal{X}_T$ we have $(X_A \perp\!\!\!\perp X_B \mid X_S, X_{C\setminus T} = x_{C\setminus T}, X_T = x_T) \in \mathbb{J}$ then $(X_A \perp\!\!\!\perp X_B \mid X_{S\cup T}, X_{C\setminus T} = x_{C\setminus T}) \in \mathbb{J}$

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 $\overline{\mathbb{J}}$.

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 [13].

[13] Duarte, E. and Solus, L. (2021). Representation of context-specific causal models with observational and interventional data

Definition 3.10 (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.

Minimal contexts and the application of the graphoid axioms are key to represent the CStree as a sequence of DAGs. For now, we denote $\mathbb{C}(\mathbb{T})$ as the set of all minimal contexts of a CStree $\mathbb{T}$, for each minimal context $(X_C = x_C)_i \in \mathbb{C}(\mathbb{T})$, there exists a DAG that encodes the CI relations that hold under this context. We 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 $\mathbb{T}$, and define the procedure to generate them later on.

We give some examples of minimal contexts below.

Example 3.11 *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 $(X_5 \perp\!\!\!\perp X_4 \mid X_{\{1,2,3\}} = 100)$ 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 3.12 *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 3.11 shows that we can get absorption from CSI relations we get from different levels, while 3.12 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, X_S 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. [14] This can be thought of as decomposing C into sets $C\setminus T$ and T such that Definition 3.10 holds. To find the largest such T, we can perform a binary search on the size of T, which can range from 0 to $|C|$. Algorithm 5 describes our approach, which makes use of Algorithm 6 as a subroutine to find the largest possible T as in the definition of the absorption axiom. [15]

[14] 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 $B, C \subset [p-1]$

[15] Each break statement leaves the first loop it meets. RelsWithTriple(A, B, S, J) returns the CSI relations with the variables A,B and conditioning set S in J.VariablesOfContext takes a context and returns the variables involved in it. Contained takes a possible outcome x_T, a list of contexts, and a context $X_L = x_L$ (which can be a possible minimal context) and returns true if there is a context $X_C = x_C$ in the list of contexts such that it can be decomposed into $X_{C\setminus T} = x_{C\setminus T}, X_T = x_T$ and $X_L = x_L$ is a subcontext of $X_C = x_C$. UpdateMinimalContexts isa a function that updates the dictionary which stores the minimal contexts alongside all the CSI relations in J that include it as a subcontext/

Theorem 3.13 *Given a set of Context Specific Independence relations $\mathbb{J}$, Algorithm 5 is correct, i.e. it returns the set of minimal contexts.*

Proof: We work in sets of CSI relations where the variables X_A,X_B and the conditioning set X_S remain the same, which we denote $\mathbb{J}_{A,B,S}$. For each CSI relation $r \in \mathbb{J}_{A,B,S}$ which has a context with context variables C_r, Algorithm 6 aims to find the largest subset T of C_r such that $\mathbb{J}_{A,B,S}$ contains relations with contexts containing all outcomes of T as a subcontext. The possible sizes of T have the property that if we have a set of size k_1 that contains all of its outcomes in $\mathbb{J}_{A.B,S}$ then all sets of size $0, ..., k_1 - 1$ also have all their outcomes in $\mathbb{J}_{A,B,S}$ as subcontexts. This means in order to maximize this size we have to search over the sizes $k_1, ..., |C_r|$. Similarly, if we find a set of size k_2 which does not contain a set with all possible sizes for T inside $\mathbb{J}_{A,B,S}$, we know this size will be in $0, ..., k_2 - 1$. Noting that the set with size 0 i.e. the empty set is always a subcontext, and since the algorithm searches over all possible sizes for T (which can only range from 0 to $|C_r|$ since it is a subset of C_r) we get the largest set T such that the absorption axiom is satisfied. Once we find the size of T, we check for T itself. For each possible T_i with the size we just found, this is done by decomposing the context variable of r, into T_i and $C_r\setminus T_i$ and ensuring all outcomes of T_i have a corresponding CI relation in $\mathbb{J}_{A,B,S}$ and the corresponding possible minimal context after this decomposition contains the context $X_{C_r\setminus T_i} = x_{C_r\setminus T_i}$ as a subcontext, which is consistent with the absorption axiom.

A crude time complexity approximation is as follows. Assume we have p variables. We start by computing how long it would take to find the optimal size for T given a CSI relation r and a set of CSI relations $\mathbb{J}_{A,B,S}$ which share the same variables X_A, X_B and conditioning set X_S. In the worst case, the possible variables for T can be all the variables, or none of them, in which case the algorithm to earch

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

```
set_triples ← TriplesOfSets(J);
MCs ← EmptyDictionary;
for A, B, S in set_triples do
    rels ← RelsWithTriple(A, B, S, J);
    cs ← ContextsOfRels(rels);
    for rel in rels do
        rel_context ← ContextOfRels([rel]);
        C ← VariablesOfContext(rel_context);
        T_sizes ← [0, ..., |C|];
        T_size ←RecursiveSearchT(T_sizes, C, cs);
        if T_size=1 then
            T = {};
        else if T_size = |C| then
            T = C;
        else
            possible_Ts ← [subsets of C of size T_size];
            for possible_T in possible_Ts do
                all_outcomes ←CartesianProduct(possible_T);
                possible_mc ← [(var, val) for (var, val) in
                 rel_context if var not in possible_T];
                outcome_count = 0;
                for outcome in all_outcomes do
                    contained ←
                     Contained(outcome, cs, possible_mc);
                    if contained = True then
                        outcome_count=outcome_count + 1;
                    end
                    if outcome_count = |all_outcomes| then
                        T = possible_T
                    end
                end
            end
        end
        MCs ←UpdateMinimalContexts(MCs, rel, T, J);
    end
end
```

Algorithm 6: RecursiveSearchT

Find the largest possible size of the set T for which the absorption axiom can be applied.

Input: Possible sizes for T denoted *T_sizes*, possible variables for T denoted T', set of contexts shared by the same triple A, B, S, denoted *cs*

Output: Size of the largest set T such that all outcomes under T are contained in *cs*

```
1  if |T_sizes| = 1 then
2  |  return T_sizes[0]
3  else
4  |  mid ← |T_sizes|/2;
5  |  candidate_size ← T_sizes[mid];
6  |  possible_Ts ← [subsets of C of size candidate_size];
7  |  not_contained = 0;
8  |  for T in possible_Ts do
9  |  |  all_outcomes ← CartesianProduct(T)
   |  |   outcome_count = 0;
10 |  |  for outcome in all_outcomes do
11 |  |  |  contained ← Contained(outcome, cs, {});
12 |  |  |  if contained = True then
13 |  |  |  |  outcome_count = outcome_count + 1;
14 |  |  |  end
15 |  |  |  if contained = False then
16 |  |  |  |  not_contained = not_contained + 1;
17 |  |  |  |  break
18 |  |  |  end
19 |  |  |  if outcome_count = |all_outcomes| then
20 |  |  |  |  RecursiveSearchT(T_sizes[mid:end], C, cs)
21 |  |  |  end
22 |  |  end
23 |  |  if not_contained = |possible_Ts| then
24 |  |  |  RecursiveSearchT(T_sizes[start:mid], C, cs)
25 |  |  end
26 |  end
```

for the size of T will run $\log p$ iterations. Within this algorithm, first midpoint is then $p/2$, which gives a total of $\binom{p}{\frac{p}{2}}$ possible subsets of p with this size. For each of set, we generate the Cartesian product, which has at most $d^{\frac{p}{2}}$ elements, and checking if the required element as per the absorption axiom is in $\mathbb{J}_{A,B,S}$ can take constant time if we use a hash table. Generalizing this, at iteration i, we will will look over $\binom{p}{\frac{p}{2^i}}$ possible subsets and corresponding $d^{\frac{p}{2^i}}$ outcomes. Doing this at most $\log p$ times gives the time complexity as

$$\sum_{i=1}^{\log p} \binom{p}{\frac{p}{2^i}} d^{\frac{p}{2^i}}$$

Once we find the optimal size for T, we then find the set T itself, which involves generating the Cartesian product over all possible subsets of the size itself. This is done for all CSI relations of the form $(X_A \perp\!\!\!\perp X_B | X_S, X_{C_i} = x_{C_i})$. Since we look at all pairs of sets A,B, and given that they are in general, subsets of $[p]$, we have $\binom{2^p}{2}$ possibilities for the pairs of sets A, B. In practice, the number of CSI relations we have to look at can be reduced, for example the Symmetry axiom allows us to half the number of pairs we need to check. We note that in practice, all of the above first assumes we have the ful set of CSI relations from the context specific closure of the CSI relations entailed by the CStree. This involves repeated application of the axioms above until no new new CSI relations are generated, and in practice is feasible for up to 4 or 5 variables depending on the independence structure of the CStree. This presents itelf to be the main bottleneck in this process for the time being.

This motivates us look at the pairwise case, where we focus on Context Specific Independence relations 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 $\binom{p}{2}$. We note that however that even though we fix pairs of variables X_i, X_j now, we could still have the conditioning set S to be any subset of size $p-2$ which does not include the nodes corresponding to X_i, X_j. The definition of pairwise minimal contexts is given below.

Definition 3.14 (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 $(X_i \perp\!\!\!\perp X_j \,|\, X_S, X_C = x_C) \in \mathbb{J}$ and there is no non-empty subset $T \subset C$ such that $(X_i \perp\!\!\!\perp X_j \,|\, X_{S \cup T}, X_C = x_C) \in \mathbb{J}$.*

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 [16]. An important question is whether the Context Specific closure of CSI relations from a CStree form a compositional context specific graphoid.

[16] Sadeghi, K. and Lauritzen, S. (2014). Markov Properties for Mixed graphs. *Bernoulli*, 20(2):676 – 696

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.

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, followed by 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 [17].

[17] Duarte, E. and Solus, L. (2021). Representation of context-specific causal models with observational and interventional data

Theorem 3.15 (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 theorem allows us to generate the Minimal Context DAGs once we have the context specific closure of the CSI relations entailed in the CStree. Namely, we take the CI relations that hold under each minimal context, and generate the minimal I-MAP [18], which we can

[18] Verma, T. and Pearl, J. (1990). *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

recover with the following procedure [19].

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

Algorithm 7: 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 3.16 *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 7 is correct i.e. it returns the minimal context DAGs for each minimal context 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 3.17 (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.

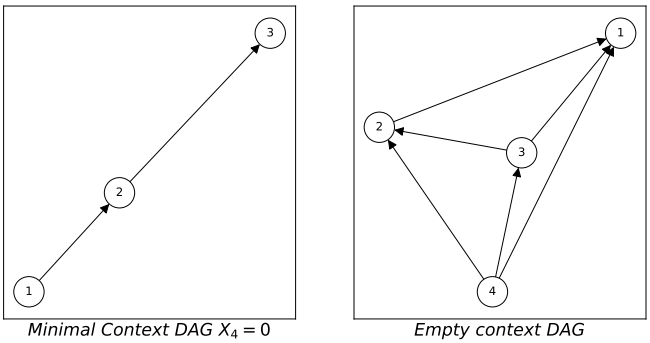


Figure 3.3: Example on why to consider all topological ordering of all Markov equivalent DAGs when learning a CStree

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.

We now present the main theorem of this paper.

Theorem 3.18 *Given variables $X_1, ..., X_p$, assuming that the data generating distribution $\mathbb{P}$ 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 $n \to \infty$.*

Proof: Since $\mathbb{P}$ is faithful to $\mathbb{T}$ it entails the CSI relations encoded by minimal context DAGs $\mathbb{G}_{\mathbb{T}}$. By Theorem 3.15, this is equivalent $\mathbb{P}$ factorizing according to $\mathbb{T}$. Faithfulness also implies that if two nodes are in different stages, they must have different labels, since otherwise we would have a CSI relation not entailed by $\mathbb{G}_{\mathbb{T}}$. Thus in the limit of large data, we can differentiate between different stages in $\mathbb{T}$, allowing us to recover it up to Markov equivalence.

3.4 Model selection for CStrees

In Algorithm 3 we select the CStree (or CStrees if there is more than one) with the causal ordering corresponding to the fewest stages, which is motivated by the principle of Occams razor [20] since few stages imply lower model complexity and all else being equal, the simplest model is the best model. The original authors also present the **Bayesian Information Criterion (BIC)** for CStrees alongside

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

a proof that it is locally consistent for CStrees, meaning it can be applicable to greedy search methods for CStrees. The BIC depends on two terms, the likelihood of the data, which assesses the quality of the model to explain the observed data, and a complexity term that depends on the amount of observed data and the free parameters of the model, with the idea being that models with more parameters should be penalized. Importantly, this helps against overfitting since one can simply add more parameters to the model to maximize the likelihood.

For a general model with random variables $X_1, ..., X_p$ the likelihood of observing a sample $(x_1, ..., x_p)$ can be described as below [21] .

[21] Here we compactify the notation further - $X_{\{1,...,p\}} = x_{\{1,...,p\}}$ is the same as $X_{\{1,...,p\}} = x_1 \cdots x_p$ which is $X_1 = x_1, ..., X_p = x_p$

$$\begin{aligned}\mathbb{P}(X_{\{1,...,p\}} = x_{\{1\cdots x_p\}}) = \mathbb{P}(X_i = x_i | X_{\{i-1,...,1\}} = x_{\{i-1,...,1\}}) \\ \cdots \mathbb{P}(X_2 = x_2 | X_1 = x_1)\mathbb{P}(X_1 = x_1)\end{aligned}$$

In the CStree model, independence is implied from the staging of the tree, and each distribution in the product above depends on the fixed context of the node in the CStree representing that distribution. Denoting C_i to be the context variables of the context of the node $x_{1\cdots i}$, this gives the following.

$$\mathbb{P}(X_{\{1,...,p\}} = x_{\{1,...,p\}}) = \prod_{i=1}^{p} \mathbb{P}(X_i = x_i | X_{C_i} = x_{C_i})$$

In order to get these values from the data, we denote $\mathbb{U}$ to be **contingency table** of the data, which a multi-dimensional array [22] with dimensions $d_1 \times \cdots \times d_p$, where d_i is the number of possible outcomes for variable X_i. Given a sample $(x_1, ..., x_p)$, the value in $\mathbb{U}[(x_1, ..., x_p)]$ is the number of times we have this sample in the dataset. The marginalized contingency table $\mathbb{U}_C$ represents the table after summing over the axes which are not in C. An important case is when C is the empty set, which results in the marginal table to be 1 dimensional scalar and is the total number of samples in the dataset - when having a stage with an empty context this reflects the conditional distribution for the corresponding variable to simply be the ratio of its outcomes in the dataset. This allows the following compact representation for the likelihood [23] for a tree with levels $(L_1, ..., L_p) \sim (X_1, ..., X_p)$.

[22] This is the same as tensors in the context of computer science

[23] Duarte, E. and Solus, L. (2021). Representation of context-specific causal models with observational and interventional data

$$\mathbb{P}(X_{\{1,...,p\}} = x_{\{1,...,p\}}) = \prod_{k=1}^{p} \frac{\mathbb{U}_{C\cup k}[x_{C\cup k}]}{\mathbb{U}_C[x_C]}$$

Given n samples arranged into a $n \times p$ array $\mathbb{D}$, and under the assumption that they are independent, the total likelihood is simply the product over all samples in $\mathbb{D}$.

The free parameters is $d = \sum_{k=1}^{p-1}(|\mathcal{X}_k| - 1)S_k$ where S_k is the number of distinct stages in level k [24]. The BIC score for a CStree $\mathbb{T}$ with observed data $\mathbb{D}$ is then

[24] Duarte, E. and Solus, L. (2021). Representation of context-specific causal models with observational and interventional data

$$\mathbf{BIC}(\mathbb{T}, \mathbb{D}) = \log \mathbb{P}(\mathbb{D} \mid \mathbb{T}) - \frac{d}{2}\log(n)$$

4
Experiments

The aim of this section is to test the theory presented so far in both synthetic and real world data. Namely, we ask ourselves the following questions.

1. Given a DAG and a causal ordering, is there a difference between learning the context specific independence relations after encoding the CI relations from the DAG into CStree, or without encoding any of these CI relations?
2. Is the CStree property assumption reasonable in practice?
3. How sensitive are the results on the method used to determine whether nodes belong to the same stage?
4. How are the results in comparison to CStrees generated from DAGs learnt using other causal discovery algorithms?
5. How well do our learnt CStrees compared to staged tree models learnt using different algorithms?

Throughout this section we omit the final layer of nodes for all the CStrees since they always belong to the singleton stage.

4.1 Synthetic data

We start by generating random DAGs and generating the corresponding CStrees as per Algorithm 1. We run a sanity check by generating CStrees from DAGs in the 2 extreme cases, the fully connected DAG and the empty DAG. The results are included in the Appendix. For random DAGs, we first choose 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 $u < v$. The causal ordering is chosen as $12 \cdots p$. We show the generated CStree below, and

recover the original DAG as a minimal context DAG with the empty context with Algorithm 7.

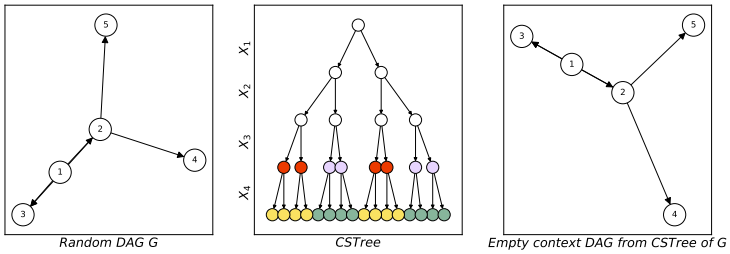


Figure 4.1: Generating CStrees from random DAGs and recovering the original random DAG from the CStree with ordering 1234 using Algorithm 7

The following table summarizes the empirical space time complexity of generating a CStree from certain random DAGs generated from the aforementioned procedure.

p_{edge}	DAG 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 4.1: 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 taken to encode the CI relations in the DAG to the CStree. We see the space complexity increase exponentially as mentioned. The time complexity however can depend on the DAG structure, however the trend suggests an exponential increase.

Next we generate a random DAG G using the procedure above, and then generate samples from this DAG, where each variable is binary valued. We then discard the DAG and apply Algorithm 3 on the generated samples. For nodes which are adjacent to another node, we sample the data by constructing a conditional probability distribution table by taking the parents of the nodes, and creating one row for each possible outcome of the parents. In this binary case, if a variable X_i has d parents $X_{PA_{X_1}}, ..., X_{PA_{X_d}}$, the corresponding table has 2^d rows. The 2 columns represent the outcome 0 and 1 respectively. The probability values are filled by taking a row corresponding to some outcome of the parents, and assigning the probability of $X_i = 0$ under this outcome to be distributed uniformly between $[0.01, 0.2]$ or $[0.8, 0.99]$ each with a probability 0.5. The probability for $X_i = 1$ under this outcome is then simply computed such that the probabilities sum to 1. For the variables corresponding to nodes with no edges, we give them a Bernoulli distribution with a parameter chosen uniformly between $[0, 1]$ for each such variable.

We simulate samples on 4 binary variables using the above procedure by fixing a DAG and a distribution, and sampling 2000 values from it. The random DAG we generate is shown in Figure 4.2 We then learn the CStree using Algorithm 3 and recover the minimal context DAGs via Algorithm 7 for a random subample of 500 and all 2000 of

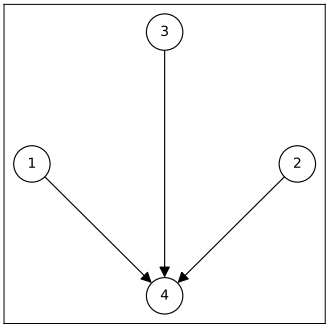


Figure 4.2: Random DAG generated to test Algorithm 3 and 7 on synthetic data. The exact probability distribution values are
$\mathbb{P}(X_4 = 0 \,|\, X_{\{1,2,3\}} = 000) = 0.06$,
$\mathbb{P}(X_4 = 0 \,|\, X_{X\{1,2,3\}} = 001) = 0.85$,
$\mathbb{P}(X_4 = 0 \,|\, X_{\{1,2,3\}} = 010) = 0.87$,
$\mathbb{P}(X_4 = 0 \,|\, X_{X\{1,2,3\}} = 011) = 0.1$,
$\mathbb{P}(X_4 = 0 \,|\, X_{\{1,2,3\}} = 100) = 0.1$,
$\mathbb{P}(X_4 = 0 \,|\, X_{X\{1,2,3\}} = 101) = 0.16$,
$\mathbb{P}(X_4 = 0 \,|\, X_{X\{1,2,3\}} = 110) = 0.94$,
$\mathbb{P}(X_4 = 0 \,|\, X_{X\{1,2,3\}} = 111) = 0.86$,
$\mathbb{P}(X_1 = 0) = 0.14$,$\mathbb{P}(X_2 = 0) = 0.39$,$\mathbb{P}(X_3 = 0) = 0.37$

these samples. The results of these are shown in Figure We choose 4 variables due to limitations of applying the context specific graphoid axioms. We also set the ordering to be 1234.

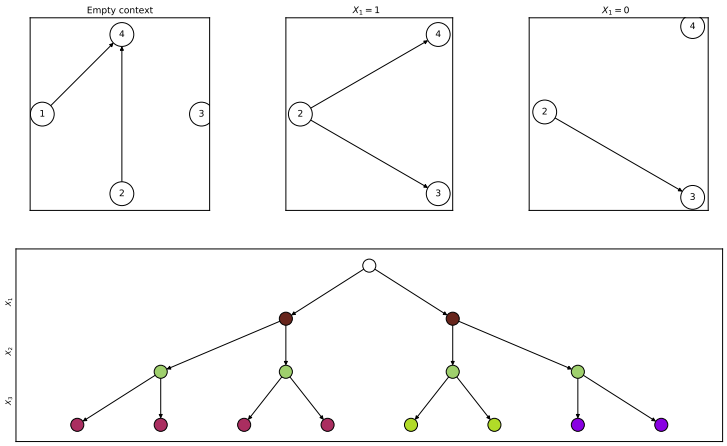


Figure 4.3: CSTree and corresponding minimal context DAGs from the 500 samples from the synthetic DAG experiment.

We expect that in the limit of large data, the CSI relations we learn would converge to the CI relations implied by the DAG. The above procedure could produce non empty minimal contexts for several reasons, for example due to small sample sizes, or errors in the context specific independence testing procedure. We see that the empty context DAG does not correspond to the true DAG in this case. Although this did not happen significantly often on our synthetic data experiments, in practice the CI testing used to learn the DAG model and the CSI relations can be highly unstable, and this will be a common theme in this chapter. For example it is possible for the CI testing to erroneously encode a CI relation, implying that that the independence relation holds for all outcomes whilst in reality it only

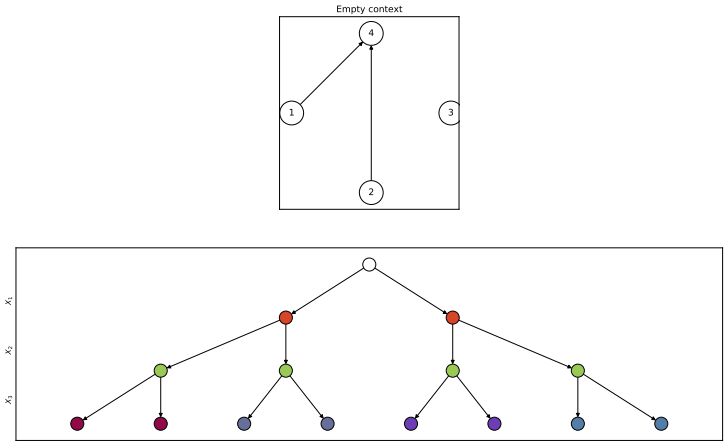


Figure 4.4: CSTree and corresponding minimal context DAG from the 2000 samples from the synthetic DAG experiment.

holds under specific contexts. Similarly, it possible for the CSI tests to erroneously merge stages, entailing a CSI relation that does not exist.

In order to assess the influence of the CI relations from the DAG in the final CStree, we tested the use of Algorithm 3 without the CI relations from the initial DAG on the same 2000 samples used to get the results in 4.4. The results are shown in Figure 4.5. In this case we do recover the true DAG. Since this CStree was learnt only using the CSI relations, it is reasonable to assume that the errors in CI testing contributed to the previous result.

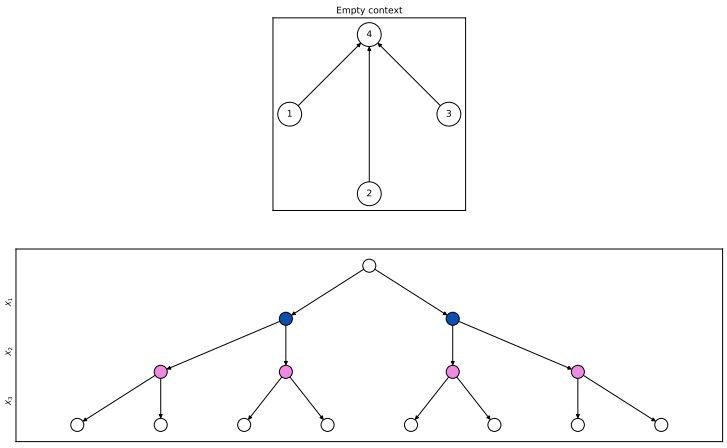


Figure 4.5: CSTree and corresponding minimal context DAG from the same samples as in Figure 4.4 but without encoding the CI relations from the initial DAG.

4.2 Coronary disease data

This is a dataset consisting of features which might increase the risk for coronary thrombosis, and consists of 1841 samples from men [1]. These features are detailed below.

[1] Reinis, Z., Pokorny, J., Basika, V., Tiserova, J., Gorican, K., Horakova, D., Stuchlikova, E., Havranek, T., and Hrabovsky, F. (1981). Prognostic significance of the risk profile in the prevention of coronary heart disease. *Bratisl Lek Listy*, 76:137–150

Variable name	Node	Outcomes
Smoking	1	$\{Yes, No\}$
Strenuous mental work	2	$\{Yes, No\}$
Strenuous physical work	3	$\{Yes, No\}$
Systolic blood pressure	4	$\{(-\infty, 140], (140, -\infty)\}$
Ratio of beta and alpha lipoproteins	5	$\{(-\infty, 3], (3, -\infty)\}$
Coronary heart disease family history	6	$\{Yes, No\}$

This data set does not require further pre-processing since the outcome space is already categorical, and there were no missing values.

We first perform the following experiment: Learn a DAG model from both the PC algorithm and the Hill climbing search algorithm. Now for each topological ordering of all the DAGs which are Markov equivalent, we store the following CStrees:

1. The CStree generated from the DAG itself
2. The CStree learnt only from context specific information without encoding the CI relations from the DAG
3. The CStree learnt from encoding the CI relations from the DAG and then learning further context specific information

For each of the cases above we create/merge stages using the Anderson-Darling test [2], the Epps-Singleton test [3], and the symmetric KL divergence with different threshold values. The use of the symmetric KL divergence is motivated by the unreliability of conditional independence testing in practice. Given 2 distributions $\mathbb{P}, \mathbb{Q}$ over the same discrete state space $\mathcal{X}$, the symmetric KL divergence is as follows.

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

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

$$D_{SKL}(\mathbb{P}, \mathbb{Q}) = \sum_{x \in \mathcal{X}} \mathbb{P}(X = x) \log \frac{\mathbb{P}(X = x)}{\mathbb{Q}(X = x)} + \mathbb{Q}(X = x) \log \frac{\mathbb{Q}(X = x)}{\mathbb{P}(X = x)}$$

The output of the PC algorithm is a CPDAG, and whilst the outcome of the Hill climbing algorithm is a DAG, the process of converting a DAG to the CPDAG that represents its MEC has computationally efficient algorithms [4]. One we have the CPDAG, generating the

[4] Chickering, D. M. (2002a). Learning equivalence classes of bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498

	Goal: Min stages	Anderson		Epps		SKL 5×10^{-5}		SKL 5×10^{-6}		SKL 5×10^{-7}	
		Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC
PC	DAG	18	-6739.4	18	-6739.4	18	-6739.4	18	-6739.4	18	-6739.4
PC	CSTree w/o DAG	**6**	-6793.7	13	-6812.1	23	-6757.1	31	-7032.3	44	-6753.4
PC	CSTree with DAG	8	-6771.3	11	-6780.4	15	-6775.4	18	-6739.4	18	-6739.4
HC	DAG	28	-6714.78	28	-6714.8	28	-6714.8	28	-6714.8	28	-6714.8
HC	CSTree w/o DAG	7	-6791.0	30	-6841.9	24	-6802.9	36	-6828.1	47	-6755.9
HC	CSTree with DAG	8	-6788.7	15	-6809.7	21	-6794.9	20	-6786.9	27	**-6711.0**

Table 4.2: Experimental setup 1 on the coronary dataset. Each cell under the column "Stages" is the minimum number of stages amongst all causal orderings generated from the either the PC or Hill Climbing algorithm, for different staging criteria. Many orderings produce CStrees with the same number of stages, in which case the BIC score in the table above is the maximum over those CStrees.

DAGs consistent with it involves orienting the undirected edges such that we do not form any new v-structures and do not create cycles.

We apply Algorithm 3 to the setup above, and record the CStrees with the minimum number of stages over all possible causal orderings, alongside their BIC scores. If we have CStrees with the same number of minimum stages, we record the highest BIC score among them. We call this **experimental setup 1**, and the results of this experiment for the coronary dataset is summarized in the Table 4.2.

The CPDAG from the PC algorithm gave 12 orderings, whilst the Hill climbing algorithm provided 16, and optimized for the BIC score.

Here we see that the minimum number of stages over all the configurations is 6, and which results in the CStree shown in Figure 4.6. The causal ordering is 346125, and we get this CStree by learning the original DAG using the PC algorithm, followed by the Anderson-Darling test to learn context specific information without encoding the CI relations from the original DAG. However it can be seen that the BIC score is lower compared to the CStree generated from the DAG itself and the CStree learnt using the CI relations in the DAG before learning the context specific information. The latter CStree also has more stages than the CStree learnt without using the CI relations, which can be explained by the possibility that the CI relation(s) encoded

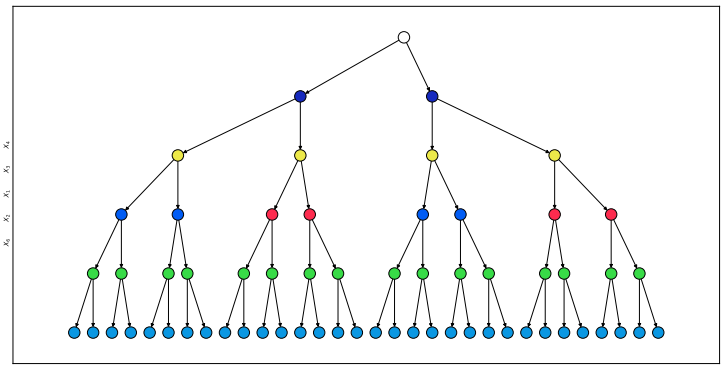


Figure 4.6: CStree for the coronary dataset with the lowest number of stages, which has ordering 346125.

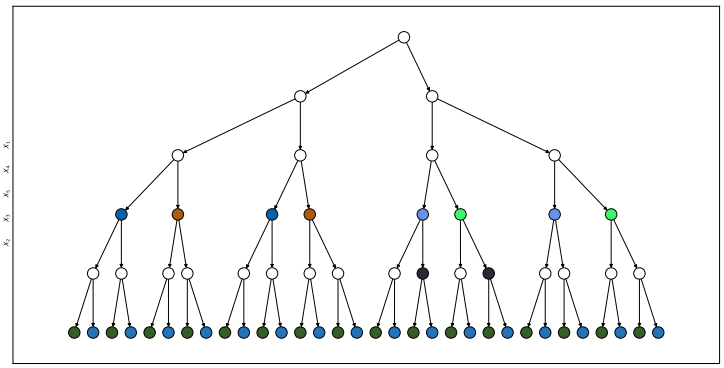


Figure 4.7: CStree with the highest BIC score among all CStrees with the minimum number of stages in experimental setup 1, which has ordering 541326.

in the DAG may have put nodes in the same stage, which otherwise may have been merged from the context specific testing procedure.

Among the trees with the lowest stages, the CStree with the highest BIC score we recorded is generated with the ordering 541326, using the symmetric KL divergence with a threshold of 5×10^{-7} for the staging procedure, from the DAG learnt from the Hill Climbing algorithm and without encoding the CI relations from it. It shown in Figure 4.7. We see that this CStree has 27 stages, which motivates us to define the **experimental setup 2**, where we find the CStree with the highest BIC score instead of the minimum number of stages as in experimental setup 1. We show the results of this in Table 4.3.

In order to compare our CStrees with similar staged trees, we run another experiment. First take the BIC optimal CStree that we have learnt which does not correspond to a DAG model. Take the ordering of this tree, and then fit corresponding staged trees using score based

	Goal: Max BIC	Anderson		Epps		SKL 5×10^{-5}		SKL 5×10^{-6}		SKL 5×10^{-7}	
		Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC
PC	DAG	18	-6739.4	18	-6739.4	18	-6739.4	18	-6739.4	18	-6739.4
PC	CSTree w/o DAG	**7**	-6783.7	13	-6812.1	23	-6757.1	43	-6749.6	44	-6753.4
PC	CSTree with DAG	8	-6771.3	11	-6780.4	15	-6775.4	18	-6739.4	18	-6739.4
HC	DAG	31	-6714.8	28	-6714.8	28	-6714.8	28	-6714.8	28	-6714.8
HC	CSTree w/o DAG	8	-6775.5	31	-6826.4	27	-6802.8	39	-6828.0	47	-6755.9
HC	CSTree with DAG	9	-6773.2	16	-6794.1	24	-6786.6	20	-6786.49	27	**-6711.0**

Table 4.3: Experimental setup 2 on the coronary dataset, the only difference to experimental setup 1 is that we now maximize the BIC score over all possible causal orderings.

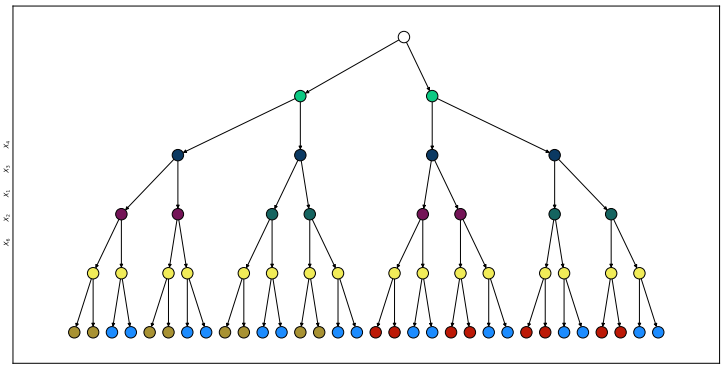


Figure 4.8: CStree with causal ordering 431265 learnt from the coronary dataset which has 8 stages and does not correspond to a DAG model.

learning algorithms, optimizing for the BIC score. The algorithms used to fit the staged trees are the following:

1. Hill climbing - Start from the full independence model, then in a greedy manner, for each node, either assign it an existing stage or create a new stage depending on whichever option best increases the score.

2. Backward hill climbing - Similar to hill climbing except we start from a full model, i.e. one where each node belongs to a singleton stage.

3. Backward joining - Start from the full model, and iteratively merge stages depending on whether a certain measure between them is below a threshold. We use the KL divergence with twice the threshold which gave us the BIC optimal CStree from our methods.

4. Hierarchial clustering - Start from a full model, and iteratively cluster nodes in level k to c_k clusters using hierarchial clustering. We use the total variation measure alongside 2 clusters for each level.

We call our approach to fit the CStree modified Algorithm 3 since the BIC optimal CStree is not necessarily learnt using the PC algorithm to get the initial DAG.

We call this **experimental setup 3**, and detail the results of it on the coronary dataset below. None of the staged trees are CStrees, and they are included in the Appendix.

Model	Algorithm	BIC	Stages
CStree	Modified Algorithm 3	-6729.5	30
Staged tree	Hill climbing	-6645.3	11
Staged tree	Backward hill climbing	-6641.1	13
Staged tree	Backward joining	-6790.4	57
Staged tree	Hierarchial clustering	-6654.7	10

We see that the staged trees achieve a higher BIC score, which is probably due to their greater flexibility.

In order to generate the sequence of DAGs that capture the context specific causal information from this dataset, we remove one variable for computational feasibility. We remove the variable that results in the highest BIC score after its removal, which in this case is the variable corresponding to systolic blood pressure (variable 4). The initial DAG is generated from the PC algorithm, and we learn the tree structure using both the CI relations from the DAG and the context

specific staging procedure using the symmetric KL divergence with a threshold of 5×10^{-6}. We choose the BIC optimal tree rather the one with the fewest stages, since the latter results in a context specific closure which could be computed in a reasonable amount of time. This results in the CStree with ordering 62351 which as 10 stages, shown in Figure 4.9.

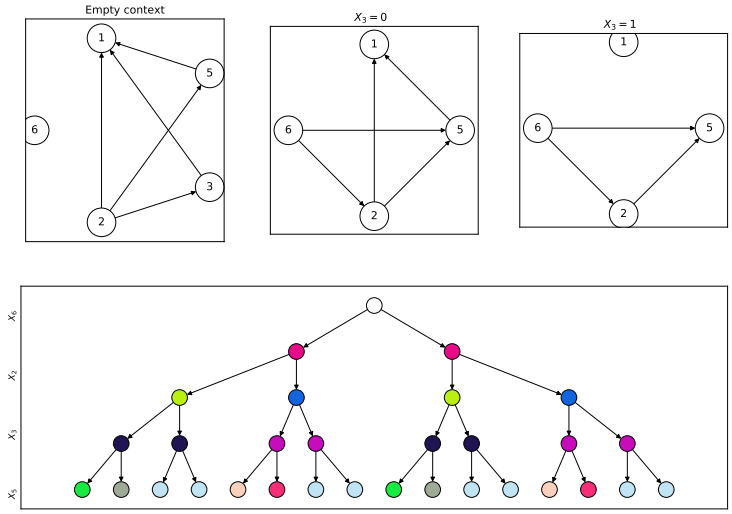


Figure 4.9: BIC optimal CStree alongside the minimal context DAGs on the coronary dataset, which has the ordering 62351. The initial DAG is learnt with the PC algorithm, and the stages were merged using the symmetric KL divergence with a threshold of 5×10^{-6}.

4.3 Mice protein expression data

This is a dataset with expression levels of 77 proteins, measured in the cerebral cortex of 8 classes of mice [5]. There are 38 control mice and 34 trisomic mice, and for each of them 15 separate measurements were taken which gives a total of 1080 samples. The aim is to identify features that can discriminate between the 8 classes that make up all possible combinations of the following 3 binary features - Control or Trisonomy, stimulated to learn or not, injected with saline or memantine. This dataset had missing values, which we handle by first inspecting the missing value counts for each feature and then removing features which had 180 or more missing values. This removes the expression data corresponding to the columns $BAD_N, BCL2_N, H3AcK18_N, EGR1_N, H3MeK4_N$. This still leaves 72 features, which we further reduce by recursive feature elimination [6] where we first train a linear support vector classifier on all the features and prune the least important features in a greedy manner until we arrive at the number of features we want. After this we compute the medians for each feature and assign each feature to take binary outcomes depending on whether or not the value is above or below the median.

[5] Higuera, C., Gardiner, K. J., and Cios, K. J. (2015). Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):e0129126

[6] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). *Machine Learning*, 46(1/3):389–422

We choose 7 features and the predictor variable, giving a system of 8 variables. This number is chosen since this gives a maximum of 196 causal orderings which is computationally feasible, meanwhile increasing the variables to 9 and 10 increase this to 1008 and 13608 respectively if we use the PC algorithm for the initial DAG. The chosen variables are detailed below.

Variable name	Node	Outcomes
AKT_N	1	$\{(-\infty, Q_2^1], (Q_2^1, \infty)\}$
APP_N	2	$\{(-\infty, Q_2^2], (Q_2^2, \infty)\}$
$SOD1_N$	3	$\{(-\infty, Q_2^3], (Q_2^3, \infty)\}$
$NR2B_N$	4	$\{(-\infty, Q_2^4], (Q_2^4, \infty)\}$
$pNUMB_N$	5	$\{(-\infty, Q_2^5], (Q_2^5, \infty)\}$
$IL1B_N$	6	$\{(-\infty, Q_2^6], (Q_2^6, \infty)\}$
SYP_N	7	$\{(-\infty, Q_2^7], (Q_2^7, \infty)\}$
Class	8	$\{C_i\}_{i=1}^8$

Here, Q_2^i refers to the median of the variable corresponding to node i.

We show results of applying experimental setup 1 on this dataset in Table 4.4 below. The PC algorithm resulted in 196 possible causal orderings, whilst the Hill climbing algorithm resulted in 12. We use the K2 score [7] for the Hill climbing since otherwise the number of possible orderings increases to 6408.

[7] Carvalho, A. M. (2009). Scoring functions for learning bayesian networks

	Goal: Min Stages	Anderson		Epps		SKL 5×10^{-4}		SKL 5×10^{-5}		SKL 5×10^{-6}	
		Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC
PC	DAG	154	**-5615.9**	154	**-5615.9**	154	**-5615.9**	154	**-5615.9**	154	**-5615.9**
	CStree w/o DAG	**7**	-6753.0	8	-6572.7	61	-5917.8	64	-5823.9	78	-5807.6
	CStree with DAG	11	-6591.4	11	-6591.4	67	-5702.0	68	-5705.4	71	-5692.2
HC	DAG	502	-6472.1	502	-6472.1	502	-6472.1	502	-6472.1	502	-6472.1
	CStree w/o DAG	8	-6191.5	17	-6068.9	67	-6386.9	90	-6348.8	90	-6348.8
	CStree with DAG	9	-6173.3	9	-6173.3	29	-6015.7	37	-6429.9	45	-6323.8

Table 4.4: Experimental setup 1 on the mice protein expression dataset

The results from Table 4.4 show once again that the Anderson-Darling test tends to merge stages more often thus resulting in a CStree with the minimum number of stages at the expense of a worse fit to the data, as indicated by the BIC score. The minimum number of stages we get is 7, which is not helpful considering that this is the full independence model. We also note the high number of stages in the DAG model from the Hill climbing algorithm, which is due to the high degree of connectivity in the graph, alongside the variable corresponding to the predictor class taking 8 possible outcomes. The CStrees corresponding to 8 and 9 stages are also DAG models, however we do learn CStrees with 10 stages which are not DAG models. We show one of these trees below in Figure 4.10.

We show the result of experimental setup 2 on this dataset in Table 4.5.

The CStree with the highest BIC score has 133 stages, with a score of -5433.0. This tree has an ordering of 32765148, and we show it in Figure 4.11

We now take the recorded ordering which gave the highest BIC score on this dataset (which is 3276514) and perform experimental setup 3. The results are summarized in Table 4.6.

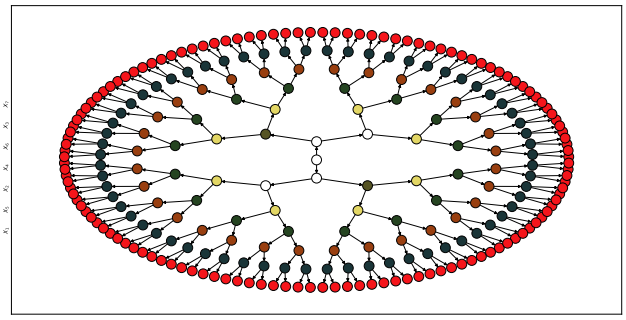


Figure 4.10: CStree with ordering 73642518 learnt on the mice protein expression dataset, and has 10 stages. This CStree does not correspond to a DAG model. The initial DAG was learnt using the PC algorithm, and we use the Epps-Singleton test to merge stages.

	Goal: Max BIC	Anderson		Epps		SKL 5×10^{-4}		SKL 5×10^{-5}		SKL 5×10^{-6}	
		Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC
PC	DAG	154	-5615.9	154	-5615.9	154	-5615.9	154	-5615.9	154	-5615.9
PC	CSTree w DAG	15	-6542.4	45	-5804.4	113	-5474.4	115	-5513.2	133	**-5433.0**
PC	CSTree with DAG	11	-6591.4	36	-5906.2	75	-5598.9	76	-5602.4	79	-5589.1
HC	DAG	502	-6472.1	502	-6472.1	502	-6472.1	502	-6472.1	502	-6472.1
HC	CSTree w DAG	11	-6067.4	20	-5944.9	68	-5906.6	97	-5679.8	96	-5684.9
HC	CSTree with DAG	12	-6049.3	25	-5944.8	45	-5993.4	47	-5843.9	55	-5786.0

Table 4.5: Experimental setup 2 on the mice protein expression dataset.

Model	Algorithm	BIC	Stages
CStree	Modified Algorithm 3	-5433.0	133
Staged tree	Hill climbing	-5320.1	28
Staged tree	Backward hill climbing	-5227.6	36
Staged tree	Backward joining	-6427.1	148
Staged tree	Hierarchial clustering	-5990.4	17

Table 4.6: Experimental setup 3 on the mice protein expression dataset

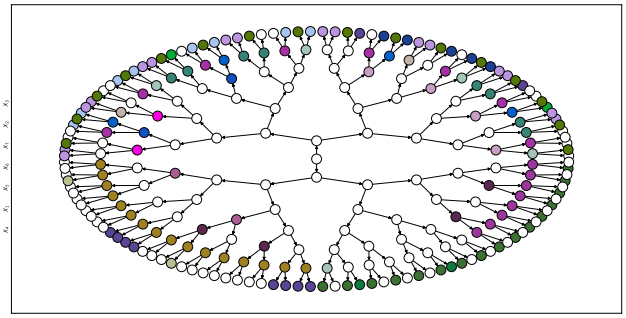


Figure 4.11: CStree with ordering 32765148 learnt on the mice protein expression dataset, and has 133 stages. This CStree does not correspond to a DAG model, and is the BIC optimal tree we learnt on this dataset. The initial DAG was learnt using the PC algorithm, and we use the symmetric KL divergence with a threshold of 5×10^{-6} to merge stages.

We see that for this dataset our BIC optimal CStree has a higher number of stages, whilst more sparse staged trees have a higher BIC score. This could be due to the higher flexibility in how staging could be done in staged trees.

In order to visualize the minimal context DAGs for the mice protein expression dataset, we first again perform recursive feature elimination to get 4 features plus the predictor variable. This results in the variables 2,3,4,5,8 from the original variables. We then apply Algorithm 3 to learn the CStree using the PC algorithm alongside the symmetric KL divergence with a threshold of 5×10^{-6}. The results are shown below in Figure 4.12 below.

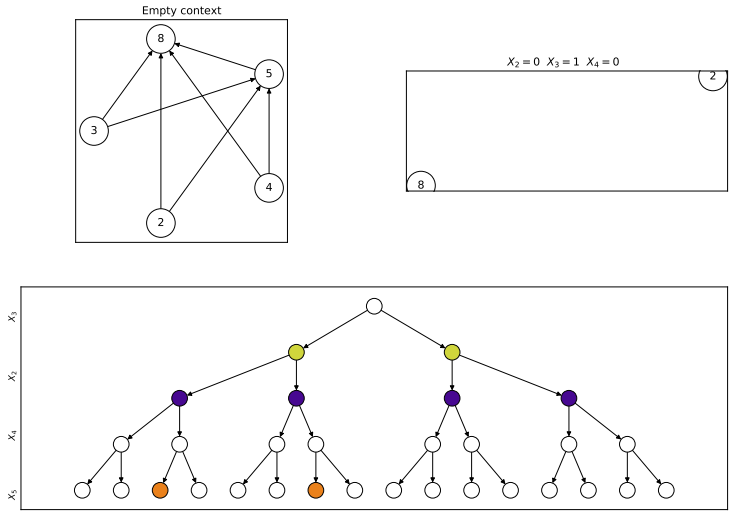


Figure 4.12: Minimum stage CStree learnt from the mice protein expression dataset on the subset of variables 2,3,4,5,8. This CStree has the ordering 32458. The original DAG was learnt using the PC algorithm, and we use the symmetric KL divergence with a threshold of 5×10^{-6} to merge stages.

We note that all of the CStrees we believe might be good candidate

models and thus shown above have a causal ordering ending with 8, which is the predictor variable for this dataset. This aligns with the aim of this dataset is to discriminate between the 8 classes for this variable, since identifying features that discriminate the predictor class has the same meaning as finding the features that cause changes in the predictor class, in which case we would expect the feature values to happen before the predictor variable values.

4.4 Supersymmetry data

This is a simulated dataset involving 8 kinematic features measured by detectors in a particle accelerator alongside 10 more high-level features which are functions of the original 8 features [8]. The aim is is to identify the class label for each sample which denotes whether the measurement corresponds to a signal or background event. The dataset contains in total 5000000 samples. There are no missing values in this dataset, and we apply it on the 8 low-level features, and after taking a random subsample of size 2500000 samples. Our aim with using this dataset is to assess the empirical feasibility of our methods on large sample sizes. The number of valid causal orderings varies largely depending on the subsampled data, which ranged from 200 to 8000 in our experiments. As a result, we simply learn the CStree with the first ordering in the algorithm, using the KL divergence with a threshold of 5×10^{-7}. The process of learning the DAG model, converting it to a CStree, and learning CStrees using with and without the CI relations from the DAG for one ordering took approximately 6-7 hours on an Intel i7-7700 machine. Similar to before, we categories each variable in binary classes depending on whether or not they have higher or lower than the median value of the corresponding samples.

[8] Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1)

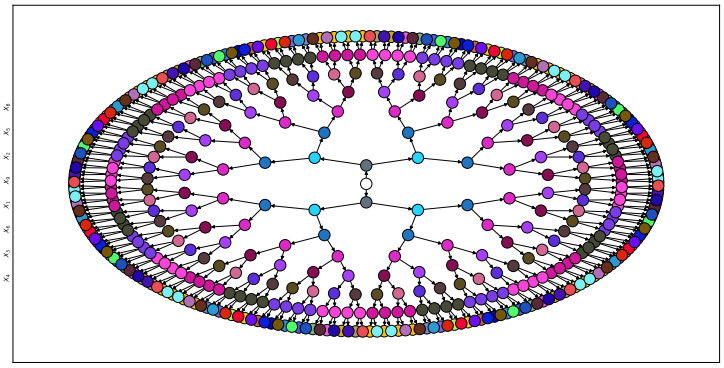


Figure 4.13: First CStree learnt from the supersymmetry dataset, with causal ordering 852916347.

We see that the resulting CStree perhaps surprisingly does not include any non-singelton stages.

4.5 Vitamin-D data

This dataset comes from a real study on Vitamin D and mortality [9], and contains 4 feature variables corresponding to age, a binary indicator denoting mutations in the filaggrin gene, vitamin D levels measured as serum 25-OH-D (nmol/L), follow up time, alongside the predictor variable which is a binary value indicating knowledge of whether the subject passed away during the follow up. There are 2571 samples, with no missing data. Looking at the data it is clear that the ages are grouped into 4 distinct groups. We group the vitamin D measurement into 4 groups based on the quartiles of the data, and the follow up time is grouped into a binary outcome with the median being the cutoff point. The resulting dataset is summarized in the table below.

[9] Martinussen, T., Sørensen, D. N., and Vansteelandt, S. (2017). Instrumental variables estimation under a structural cox model. *Biostatistics*, 20(1):65–79

Variable name	Node	Outcomes
Age	1	$\{[40,47),[47,57),[57,67),[67,80)\}$
Filaggrin	2	$\{Yes, No\}$
Vitamin D	3	$\{[-\infty, Q_1^3), [Q_1^3, Q_2^3), [Q_2^3, Q_3^3), [Q_1^3, -\infty)\}$
Follow up time	4	$\{(-\infty, Q_2^4], (Q_2^4, \infty)\}$
Passed away	5	$\{Yes, No\}$

Here Q_1^i, Q_2^i, Q_3^i respectively denote the lower quartile, median and upper quartile of the variable corresponding to node i.

The results of experimental setup 1 and 2 are shown in Tables 4.7 and 4.8 respectively. We note that the PC algorithm gave us 40 possible causal orderings, whilst the Hill climbing algorithm gave 10.

We again get the full independence model when using the Anderson test. The symmetric KL divergence thresold of 5×10^{-1} also results in the full independence model which suggests that it is too low. Among the CStrees with the minimum stages, the highest BIC score is -8405.2 which is achieved by using the CI relations learnt from the PC algorithm alongside context specific merging using the symmetric KL divergence with a threshold of 5×10^{-2}. We show the resulting CStree and minimal context DAGs in Figure 4.14 below.

We show the results of experimental setup 2 on the Vitamin D dataset below in Figure 4.8. The results are exactly the same, namely, the CPDAG learnt from both the PC algorithm and the Hill climbing algorithm coincide, alongside the entries corresponding to the minimum number of stages and maximum BIC over all orderings per

	Goal: Min stages	Anderson		Epps		SKL 5×10^{-1}		SKL 5×10^{-2}		SKL 5×10^{-4}	
		Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC
PC	DAG	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1
PC	CSTree w DAG	**4**	-9041.5	13	-8800.0	**4**	-9041.5	5	-8771.7	9	-8805.6
PC	CSTree with DAG	6	-8480.1	12	-8410.1	**4**	-9041.5	9	**-8405.2**	12	-8410.1
HC	DAG	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1
HC	CSTree w DAG	**4**	-9041.5	13	-8800.0	**4**	-9041.5	5	-8771.7	9	-8805.6
HC	CSTree with DAG	6	-8480.1	12	-8410.1	**4**	-9041.5	9	**-8405.2**	12	-8410.1

Table 4.7: Experimental setup 1 on the Vitamin D dataset

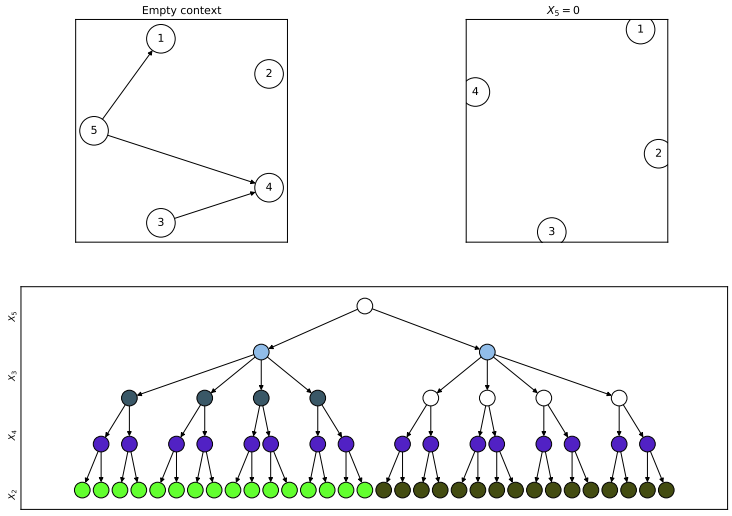


Figure 4.14: CStree and corresponding minimal context DAGs with the lowest BIC in both experimental setup 1 and 2 on the Vitamin D dataset. The causal ordering is 53421.

configuration.

Table 4.9 shows the results of experimental setup 3 on the Vitamin D dataset.

	Goal: Max BIC	Anderson		Epps		SKL 5×10^{-1}		SKL 5×10^{-2}		SKL 5×10^{-4}	
		Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC	Stages	BIC
PC	DAG	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1
	CSTree w DAG	6	-8487.9	14	-8500.0	5	-8749.9	19	-8428.2	37	-8489.0
	CSTree with DAG	6	-8480.1	12	-8410.1	5	-8749.9	9	**-8405.2**	12	-8410.1
PC	DAG	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1	12	-8410.1
	CSTree w DAG	6	-8487.9	14	-8500.0	5	-8749.9	19	-8428.2	37	-8489.0
	CSTree with DAG	6	-8480.1	12	-8410.1	5	-8749.9	9	**-8405.2**	12	-8410.1

Table 4.8: Experimental setup 2 on the Vitamin D dataset

Model	Algorithm	BIC	Stages
CStree	Modified Algorithm 3	-8405.2	9
Staged tree	Hill climbing	-8423.1	11
Staged tree	Backward hill climbing	-8423.4	12
Staged tree	Backward joining	-8560.8	26
Staged tree	Hierarchial clustering	-8910.0	10

Table 4.9: Experimental setup 3 on Vitamin D dataset with ordering 53421

The results of Table 4.9 show that the CStree does better than all staged tree models learnt in terms of the BIC score, and also provides the sparsest model learnt directly from the data, which is a fair comparison if we exclude the number of stages learnt from the hierarchial clustering algorithm where we pre define the number of maximum stages per level. However, it is very important to note that the causal ordering is starting from the variable corresponding to whether the subject passed away or not, which is not consistent with the true causal ordering. This is because the outcome corresponding to this variable is the last to be measured in the data generating process. This suggests errors in the initial DAG, which may have been caused due to instability in the conditional independence tests or being stuck at a local minima. Similarly, it would make sense if the variable corresponding to the follow up time happens second to last, since the only measurement taken during the follow up is whether the subject passed away or not. We thus learn the CStrees for all possible orderings ending with 45, and rerun experimental setup 3 with a more realistic causal ordering.

The result of this is that none of the DAGs in the MEC of the DAG learnt using both the PC algorithm and the Hill climbing search were consistent with any of the orderings ending with 45. Thus, we only rely on context specific information learnt from the staging procedure, without any CI relations. Technically, the highest BIC score was achieved using the Epps test, which gives the CStree with ordering 12345 a BIC score of -8734.5 - however this CStree has 48 stages. This is followed by the CStree obtained using the symmetric KL divergence with a threshold of 5×10^{-2}, which gives the ordering 21345 with a BIC score of -8771.7. However, this is a DAG model with one edge $4 \rightarrow 5$ in the empty context graph. A balance is achieved when taking the causal ordering 32145, which has a slightly worse BIC score of -8776.5 and 9 stages. As a result, we use this as the basis for the comparison with staged trees. The results are summarized in Table 4.10 below.

Model	Algorithm	BIC	Stages
CStree	Modified Algorithm 3	-8776.5	9
Staged tree	Hill climbing	-8406.5	13
Staged tree	Backward hill climbing	-8403.5	16
Staged tree	Backward joining	-8547.1	19
Staged tree	Hierarchial clustering	-8628.6	10

Table 4.10: Experimental setup 3 on Vitamind D dataset with orrdering 32145

We see that in terms of the BIC score the CStree is not optimal, however the model has fewer stages when compared to the staged trees. Despite this, the CStree maintains the benefit of the more intuitive representation in terms of minimal context DAGs. We show these

alongside the corresponding CStree in Figure 4.15 below.

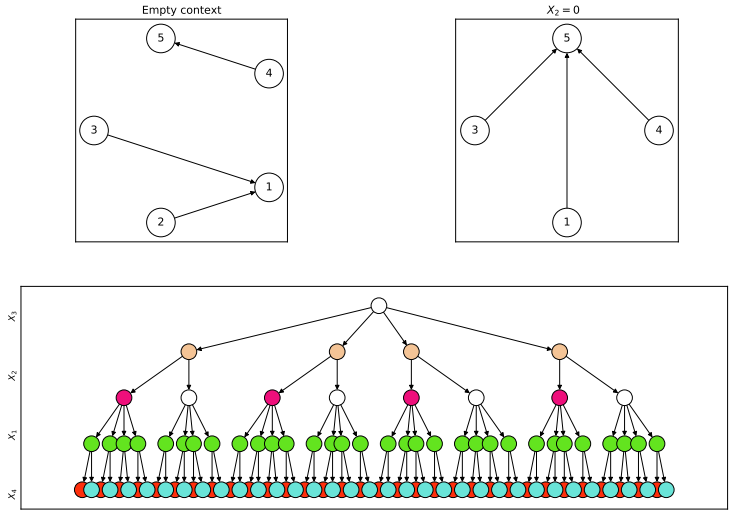


Figure 4.15: CStree and minimal context DAGs of the Vitamin D dataset for the causal ordering 32145.

We see that the minimal context DAGs contain v-structures, meaning the directed edges are fixed in the Markov Equivalence class. Thus under our assumptions, they represent direct causal relationships learnt from the data.

4.6 Discussion of empirical performance

We see in general that using the symmetric KL divergence to identify stages in the CStree result a higher BIC score when compared to choosing a CStree with the minimum number of stages There are many reasons which could lead to this happening in practice, with the first being unreliability of the statistical independence tests, which is in general a very hard task [10]. We also see that the Anderson-Darling test tends to merge stages more frequently when compared to other staging procedures, which sometimes result in the full independence model.

[10] Shah, R. D. and Peters, J. (2020). The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48(3)

If the data generating distribution happens to be faithful to a CStree with a large number of stages, particularly non-singleton stages, then it is possible that there is not enough samples to learn the context specific information in the finite data setting. We note that a lot of this information was learnt by comparing samples with extremely different sizes, so in practice it might be a good idea to merge the corresponding nodes into the same stage if a certain criteria is met, for example, only if both sample sizes are at least half of their mean.

The instability of DAG learning algorithms used to get the causal orderings also play a crucial role, and we observed this in practice. For example, we observed that if we happen to know the causal ordering and then feed it to the algorithm, we could learn a DAG in the initial step whose MEC contains no DAGs which are consistent with this causal ordering. This may also present itself to be a problem if we have partial knowledge of the ordering, for example in the mice cortex data experiment, it is plausible to think of the predictor class to be a function of the gene expression levels.

One important aspect of this process is to choose the best CStree from all possible causal ordering, which is a model selection problem. In Algorithm 3 we opt to choose the one with the fewest stages since it relates to choosing the model with the fewest parameters, i.e. the simplest model. This goes in hand with the principle of Occam's razor [11] - in the face of many possible models, choose the simplest model. Here a simple model refers to one which has few parameters. In practice however we see that problems arising due to misleading conditional independence tests and small data samples might lead to the simplest model being too simple, compromising the fit to the data.

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

All of the above analysis also takes for granted the validity of the approaches taken to group data into categories in the first place. The approach we take here is on the more simpler side, since we partition variables whos values are ordered. Even if we are given categorical data, it might be in the best interest to group since it is possible to have 2 or more features which provide the same level of information. This would also help with potential issues small data sizes, since more outcomes can thin out samples which have fixed contexts.

Depending we truly want to optimize for a score in practice, it is inevitable that there will be some level of hyperparameter tuning involved. Some examples include the threshold for the symmetric KL divergence and the value corresponding to the minimum number of increase required to not stop the Hill climbing algorithm. One method is to use a grid search, however this is not scalable, in which case one can use probabilistic search methods to search for a good hyperparameter setting [12].

[12] Baptista, R. and Poloczek, M. (2018). Bayesian optimization of combinatorial structures. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 462–471. PMLR

5
Conclusions

5.1 Summary

We started with DAGs as a means to encode CI relations, and how one can use the characterization of Markov Equivalence in DAGs to learn causal structure through CI testing. We then covered the limitations of CI relations in comparison to CSI relations, and covered CStrees as a means to encode such CSI relations. We then showed how one can learn these CSI relations from observational data to learn a CStree, and how to compute minimal contexts which are important when it comes to visualizing higher dimensional CStrees. We then apply these techniques to synthetic and real data.

5.2 Future work

One of the more natural extensions of this work is to learn CStrees from interventional data. This can already be done with DAGs [1]. On the topic of model selection, it might be interesting to see the applicability of Bayesian model selection for CStrees, whereby the model evidence is used as the basis for comparing models which automatically penalizes over-complex models whilst also penalizing models which do not agree with the observed data [2]. Similar approaches have been applied to Gaussian DAG models [3]. Generalization to missing data problems would also be an interesting avenue, considering that the PC algorithm has been recently extended for missing data instances in DAGs [4]. Based on the empirical results presented in this work, it might also be interesting to jointly optimize for the minimization of the stages and the maximization of some score like the BIC score. But perhaps the most impactful extension would be formulating the problem of finding the best CStree into a continuous optimization problem, which can lead to scalable score based

[1] Yang, K., Katcoff, A., and Uhler, C. (2018). Characterizing and learning equivalence classes of causal DAGs under interventions. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5541–5550. PMLR

[2] MacKay, D. J. C. (1992). *Bayesian Interpolation*, pages 39–66. Springer Netherlands, Dordrecht

[3] Castelletti, F. (2020). Bayesian model selection of gaussian directed acyclic graph structures. *International Statistical Review*, 88(3):752–775

[4] Tu, R., Zhang, C., Ackermann, P., Mohan, K., Kjellström, H., and Zhang, K. (2019). Causal discovery in the presence of missing data. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1762–1770. PMLR

methods. Recent work has formulate the problem of searching for the best DAG according to some metric by using a characterization of acyclicity that is smooth and exact, allowing the conversion of the combinatorial problem into a purely continuous problem [5].

[5] Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc

6

Bibliography

Andersson, S. A., Madigan, D., and Perlman, M. D. (1997). A characterization of markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2).

Ankan, A. and Panda, A. (2015). pgmpy: Probabilistic graphical models using python. In *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer.

Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1).

Baptista, R. and Poloczek, M. (2018). Bayesian optimization of combinatorial structures. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 462–471. PMLR.

Carli, F., Leonelli, M., Riccomagno, E., and Varando, G. (2020). The r package stagedtrees for structural learning of stratified staged trees.

Carvalho, A. M. (2009). Scoring functions for learning bayesian networks.

Castelletti, F. (2020). Bayesian model selection of gaussian directed acyclic graph structures. *International Statistical Review*, 88(3):752–775.

Chickering, D. M. (2002a). Learning equivalence classes of bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498.

Chickering, D. M. (2002b). Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554.

Collazo, R. A., Görgen, C., and Smith, J. Q. (2018). *Chain event graphs*. CRC Press.

Duarte, E. and Solus, L. (2020). Algebraic geometry of discrete interventional models.

Duarte, E. and Solus, L. (2021). Representation of context-specific causal models with observational and interventional data.

Elias Bareinboim, Juan D. Correa, D. I. T. I. (2020). On pearl's hierarchy and the foundations of causal inference.

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

Fisher, R. (1935). *The design of experiments. 1935*. Oliver and Boyd, Edinburgh.

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

Görgen, C. and Smith, J. Q. (2017). Equivalence classes of staged trees.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). *Machine Learning*, 46(1/3):389–422.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Higuera, C., Gardiner, K. J., and Cios, K. J. (2015). Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):e0129126.

Hu, P., Jiao, R., Jin, L., and Xiong, M. (2018). Application of causal inference to genomic analysis: Advances in methodology. *Frontiers in Genetics*, 9:238.

Huang, B., Zhang, K., Gong, M., and Glymour, C. (2019). Causal discovery and forecasting in nonstationary environments with state-space models.

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

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., and Bruhin, F. (2004). pytest x.y.

Lauritzen, S. L. (1996). *Graphical models*, volume 17. Clarendon Press.

MacKay, D. J. C. (1992). *Bayesian Interpolation*, pages 39–66. Springer Netherlands, Dordrecht.

Martinussen, T., Sørensen, D. N., and Vansteelandt, S. (2017). Instrumental variables estimation under a structural cox model. *Biostatistics*, 20(1):65–79.

Meek, C. (2013a). Causal inference and causal explanation with background knowledge.

Meek, C. (2013b). Strong completeness and faithfulness in bayesian networks. *arXiv preprint arXiv:1302.4973*.

Pearl, J. (2009). *Causality*. Cambridge University Press.

Pearl, J. and Mackenzie, D. (2018). *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition.

Reinis, Z., Pokorny, J., Basika, V., Tiserova, J., Gorican, K., Horakova, D., Stuchlikova, E., Havranek, T., and Hrabovsky, F. (1981). Prognostic significance of the risk profile in the prevention of coronary heart disease. *Bratisl Lek Listy*, 76:137–150.

Runge, J., Bathiany, S., Bollt, E., Camps-Valls, G., Coumou, D., Deyle, E., Glymour, C., Kretschmer, M., Mahecha, M. D., Muñoz-Marí, J., et al. (2019). Inferring causation from time series in earth system sciences. *Nature communications*, 10(1):1–13.

Sadeghi, K. and Lauritzen, S. (2014). Markov Properties for Mixed graphs. *Bernoulli*, 20(2):676 – 696.

Schölkopf, B. (2019). Causality for machine learning.

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

Shah, R. D. and Peters, J. (2020). The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48(3).

Silander, T. and Leong, T.-Y. (2013). A dynamic programming algorithm for learning chain event graphs. In *International Conference on Discovery Science*, pages 201–216. Springer.

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

Solus, L., Wang, Y., and Uhler, C. (2021). Consistency Guarantees for Greedy Permutation-Based Causal Inference Algorithms. *Biometrika*. asaa104.

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

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. MIT press, 2nd edition.

Tarjan, R. E. (1976). Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–185.

Thwaites, P., Smith, J. Q., and Riccomagno, E. (2010). Causal analysis with chain event graphs. *Artificial Intelligence*, 174(12):889–909.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78.

Tu, R., Zhang, C., Ackermann, P., Mohan, K., Kjellström, H., and Zhang, K. (2019). Causal discovery in the presence of missing data. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1762–1770. PMLR.

Uhler, C., Raskutti, G., Bühlmann, P., and Yu, B. (2013). Geometry of the Faithfulness Assumption in Causal inference. *The Annals of Statistics*, 41(2):436 – 463.

Verma, T. and Pearl, J. (1990). *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.

Verma, T. S. and Pearl, J. (2013). On the equivalence of causal models. *CoRR*, abs/1304.1108.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J.,

van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

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

Wolpert, D. H. (2020). What is important about the no free lunch theorems?

Yang, K., Katcoff, A., and Uhler, C. (2018). Characterizing and learning equivalence classes of causal DAGs under interventions. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5541–5550. PMLR.

Zheng, X., Aragam, B., Ravikumar, P. K., and Xing, E. P. (2018). Dags with no tears: Continuous optimization for structure learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

7
Appendix A : Implementation remarks

All of the code used in this work is available at github.com/mnazaal/masters-thesis. We make extensive use of numpy [1], scipy [2] (for statistical testing), networkx [3] (for graphs), pgmpy [4] (for initial DAG learning) in Python for the work on CStrees and the stagedtrees package [5] in R to learn the staged trees.

When using the Epps-Singleton test implemented in scipy, one must add an epsilon to the difference between the upper and lower quartile within the implementation since the test performs a computation involving divison by this value. For the KL divergence we also add an epsilon to the estimated probability density in order to avoid dividing by 0 which can happen if there are no samples taking a specific value.

Our usage indicated that the Hill climbing algorithm is highly susceptible to local minima when optimizing for the BIC score.

We make use of pytest [6] to run unit tests. This thesis is written in org-mode with the intention for users to be able to execute the code during the compilation of the document itself to ensure full reproducibility.

[1] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362

[2] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272

[3] Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA

[4] Ankan, A. and Panda, A. (2015). pgmpy: Probabilistic graphical models using python. In *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer

[5] Carli, F., Leonelli, M., Riccomagno, E., and Varando, G. (2020). The r package stagedtrees for structural learning of stratified staged trees

[6] Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., and Bruhin, F. (2004). pytest x.y

8

Appendix B : Extra figures

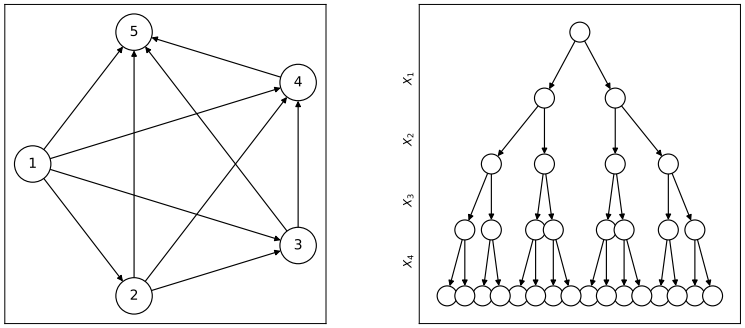


Figure 8.1: Generating the CStree for a fully connected DAG using the ordering 1234 with Algorithm 1

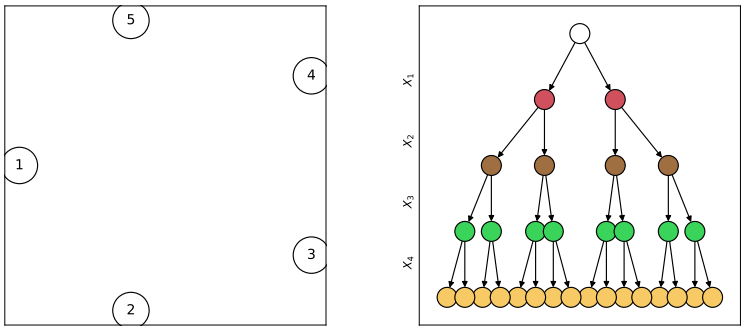
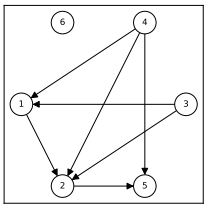


Figure 8.2: Generating the CStree for the empty DAG using the ordering 1234 with Algorithm 1

PC



Hill climbing

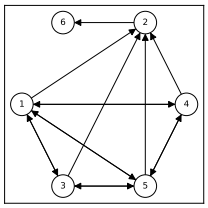
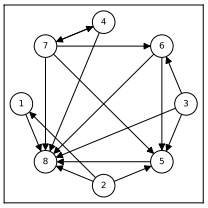


Figure 8.3: CPDAG representing the MEC of the DAG learnt from the coronary dataset. The PC algorithm suggested 12 causal orderings, whilst the Hill climbing algorithm suggested 16.

PC



Hill climbing

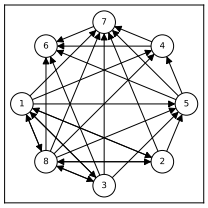
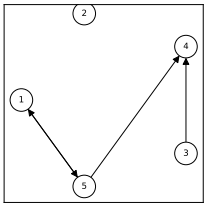


Figure 8.4: CPDAG representing the MEC of the DAG learnt from the mice protein expression dataset. The PC algorithm suggested 196 causal orderings, whilst the Hill climbing algorithm suggested 12.

PC



Hill climbing

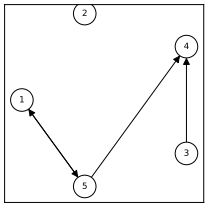
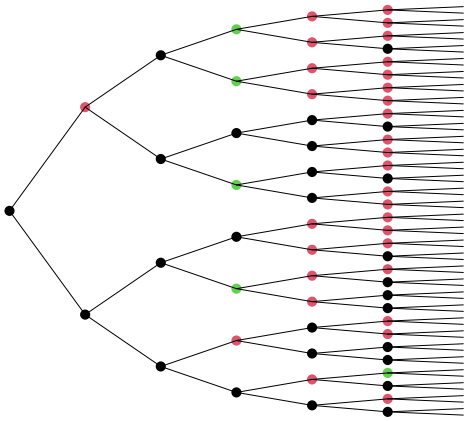
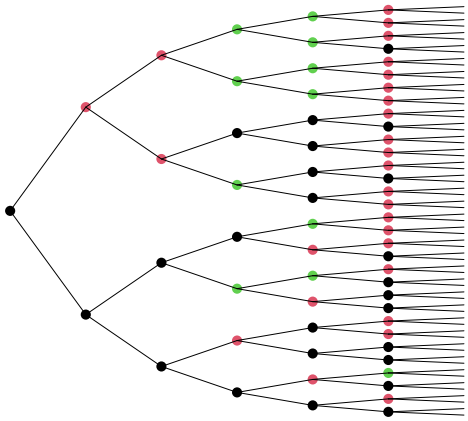


Figure 8.5: CPDAG representing the MEC of the DAG learnt from the Vitamin D dataset. Both algorithms gave the same CPDAG and suggested 40 possible causal orderings.

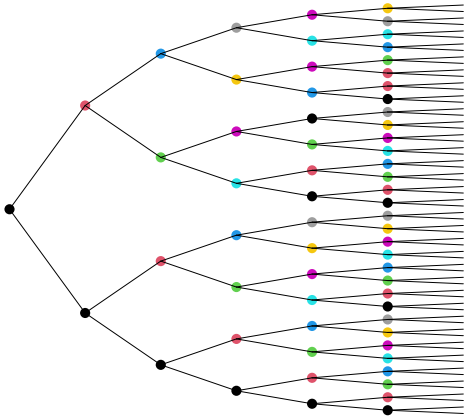
Hill Climbing



Backwards hill climbing



Backwards joining



Hierarchial clustering

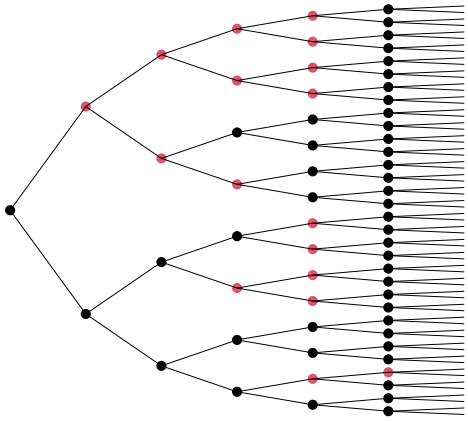
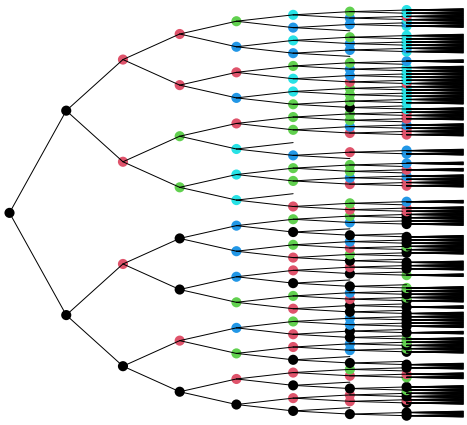
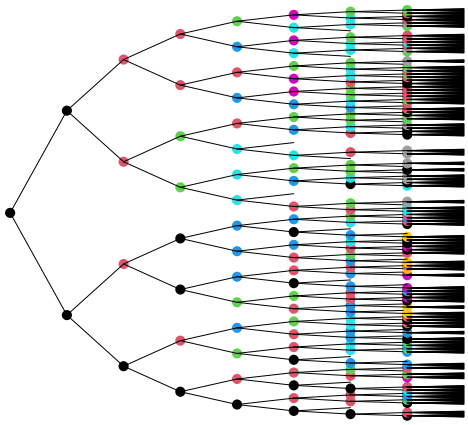


Figure 8.6: BIC Optimal staged trees learnt from the coronary dataset using the ordering 534126.

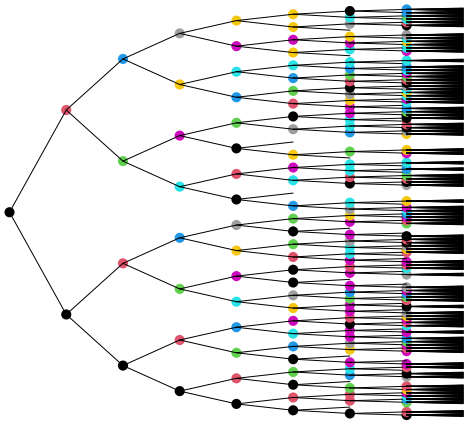
Hill Climbing



Backwards hill climbing



Backwards joining



Hierarchial clustering

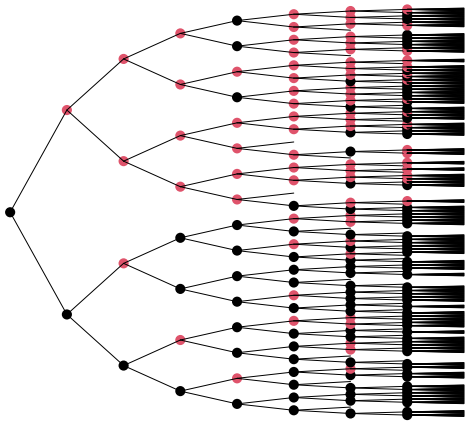
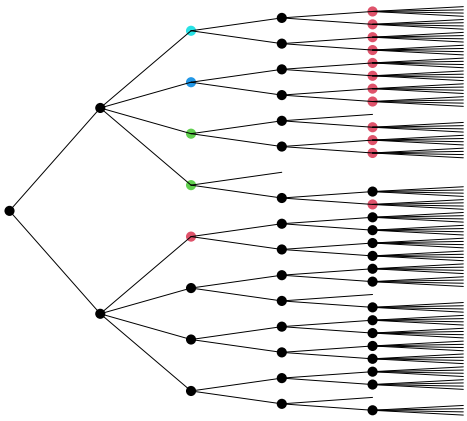
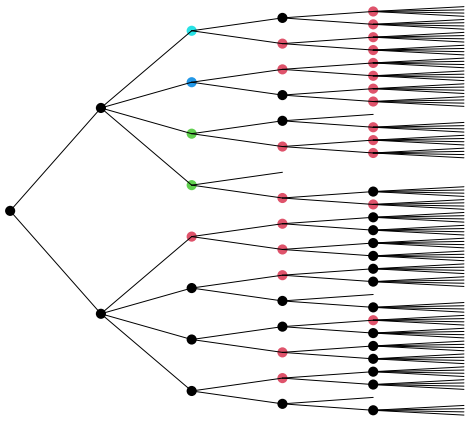


Figure 8.7: BIC Optimal staged trees learnt from the mice protein expression dataset using the ordering 32765148.

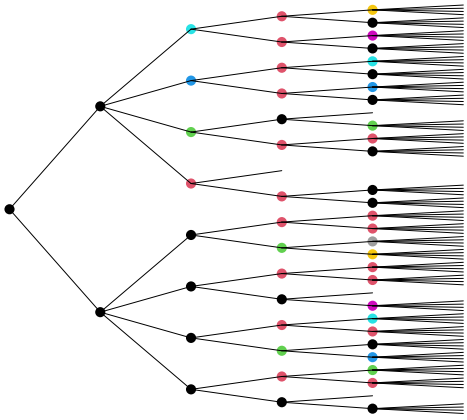
Hill Climbing



Backwards hill climbing



Backwards joining



Hierarchial clustering

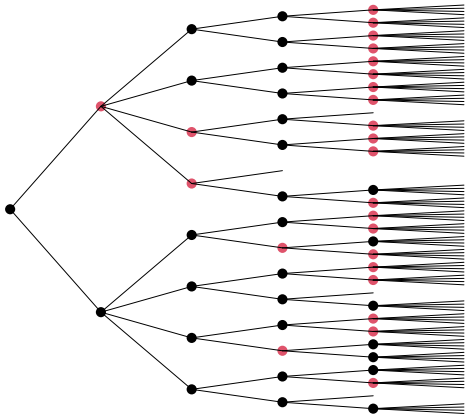
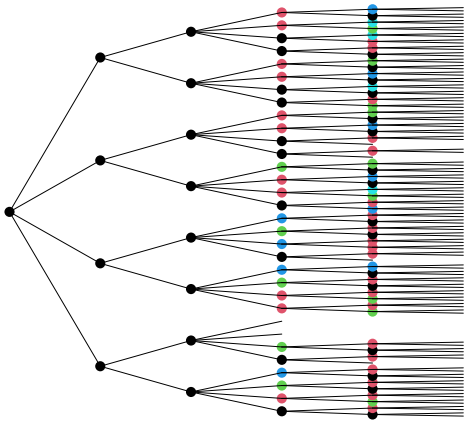
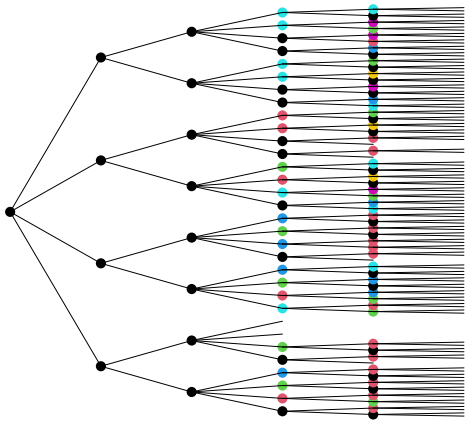


Figure 8.8: BIC Optimal staged trees learnt from the Vitamin D dataset using the ordering 53421.

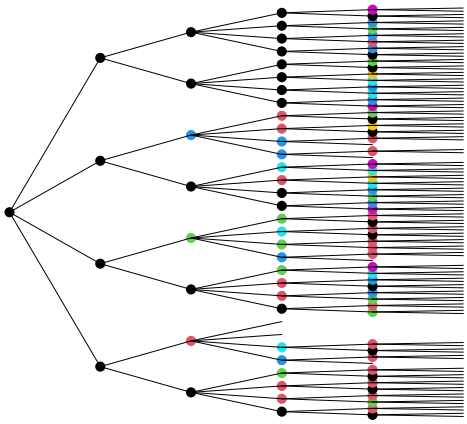
Hill Climbing



Backwards hill climbing



Backwards joining



Hierarchial clustering

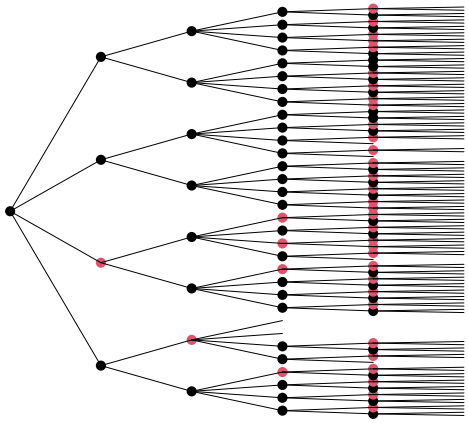


Figure 8.9: BIC Optimal staged trees learnt from the Vitamin D dataset using the ordering 32145.