Greedy causal structure learning of maximal arid graphs

Sebastiaan Jans, 6222668
Supervisor: Thijs van Ommen
Second examiner: Silja Renooij

February 14, 2024

Master's thesis in Artificial Intelligence
Utrecht University

Abstract

Causal knowledge is often modelled in directed acyclic graphs (DAGs) where an directed edge between variables, like $A \to B$, indicates that one (A) influences another (B). Many algorithms attempt the difficult task of finding such DAG models given only data – *causal discovery* – though fewer attempt to do it in the presence of unmeasured confounding variables. To represent a confounder between A and B, we can use a bidirected edge: $A \leftrightarrow B$. This gives us acyclic directed mixed graphs (ADMGs). We develop three algorithms that are variants of greedyBAP, a causal discovery algorithm that greedily searches through the space of bow-free acyclic path diagrams (BAPs) which are acyclic ADMGs with no bows (variable pairs with both a directed and bidirected edge). The modified algorithms restrict the search space from BAPs to arid graphs, that are everywhere identifiable, and maximal arid graphs, which allow each ADMG to map to a unique nested Markov equivalent arid graph. Because these arid models are smooth, the asymptotic results of the BIC model score justifying its use on graphical models holds for them. This is not the case for BAPs or unmeasured variable models in general. However, we do not find empirical evidence of improved performance of these algorithms over greedyBAPs in our simulation studies, so we conclude that they offer little to no advantage in practice.

Contents

1 Introduction

We first outline our research questions rather quickly. Concepts introduced here, as well as the basics of graphical causal analysis and causal discovery, are explained in more detail in Section 2.

Learning graphical causal models from data (causal discovery) is a difficult problem, certainly when we take into account the possibility of unmeasured confounding variables. However, a solution to it would contribute greatly to science and society in general, by giving us causal insights we may not have been able to gain otherwise.

In this work, we focus on learning from a dataset an acyclic directed mixed graph (ADMG) representation of the causal relations of the variables involved. We restrict the search space more specifically to (maximal) arid graphs (MArGs) (Shpitser, Evans, & Richardson, 2018), a class of ADMG that gives everywhere identifiable models that imply a nested Markov constraint (Richardson, Evans, Robins, & Shpitser, 2023) for each missing edge. We base this method on the greedyBAPs algorithm for learning bow-free acyclic path diagrams (BAPs) (bow-free ADMGs) by Nowzohour, Maathuis, Evans, and Bühlmann (2017), who also introduce useful notions of approximate distributional equivalence of ADMG models. Because we base it on greedyBAPs, our implementation uses linear-Gaussian models, but the same algorithm could in theory be applied equally well to discrete data.

Our research intends to answer the question of whether restricting the search space of a greedy algorithm to maximal arid graphs (MArGs) provides a worthwhile improvement over restricting it only to BAPs (a strict superclass of MArGs). Improvements may be improvements in retrieval of the correct graph, or in computational time. We also want to test how the difference in performance – if there is one – differs with the amount of variables/nodes, or with the amount of causal relations/edges.

We analyse graph retrieval performance by using the method from Nowzohour et al. (2017) for comparing the minimal absolute causal effects matrix of graphs, rather than the graphical structures themselves, because those are unreliable for ADMGs.

As mentioned, all relevant background concepts are explained in Section 2. The algorithms and their implementations are discussed in Section 3 and simulation studies and their results are discussed in Section 4, followed by a discussion of the results and some limitations of this study in Section 5 and a conclusion in Section 6.

2 Background

In this section we will describe the current state of the literature relevant for our work. We have attempted to make the text accessible also to those who are not already well-versed in graphical statistical (causal) modelling.

2.1 Intro to causality

2.1.1 Basic causal analysis

Causality is a concept that is fundamental not only to science, but to everyone's perception of the world. Everyone has an intuitive sense of what it means for one thing to cause another. Unfortunately, much of science is based on a treatment of statistics which avoids the explicit treatment of causality. Since causality is so important – we want to know whether a drug *causes* an improvement, not whether it is *associated* with one – causality is still dealt with, but informally. Standard hypothesis testing methods, like t-tests and ANOVAs, are meant to discover causal relations, though they are described statistically only in terms of association. We get causal information from them implicitly through our experimental design. This is truly unfortunate, since this informality makes causal conclusions less precise, and less reliable.

Fortunately, it is possible to treat causality more formally, with graphical causal models. The basic idea is to represent causal relations as a directed acyclic graph (DAG). This is a *graph* (network) with *directed edges* (connections or arrows) and no cycles. For our causal purposes, the nodes of the graph represent variables, and a directed edge $X \to Y$ means that variable X is a *direct cause* of variable Y. The first work on this formal treatment of causality is dealt with most comprehensively and most up-to-date in the book *Causality* (Pearl, 2009).[1]

We will demonstrate the usefulness of formal causal analysis using DAG models using an example adapted from the introductory textbook *Causal inference in statistics: A primer* (M. Glymour, Pearl, & Jewell, 2016). Say that we have observational data on a group of patients of some disease. That is, data from simple data collection: we have not performed an experiment to generate the data. We know for each of the patients whether they underwent a treatment (X), whether they recovered from the disease (Y), and whether they were under 30 years old (Z). All three are binary variables that have a value of 1 when true and 0 when false.

We use statistical associations (correlations) in the data and our understanding of the situation to model the causal relations of these variables graphically as a DAG, shown in Figure 1a. We see a positive correlation between the treatment and recovery, and we think that this is a causal effect, so we add an edge $X \to Y$. We also see that there is a correlation between age and recovery: more younger people recover. Since recovery from a disease does not influence one's age, we think that being younger helps with recovery, so we add $Z \to Y$. We also see that younger people are more likely to get the treatment. Of course, treatment also does not affect age, so $X \to Z$, like $Y \to Z$, is rejected, and we add $Z \to X$.

We would argue that for this clear visual representation of causation alone, as opposed to less clear textual descriptions, graphical representation of causality

[1] A more informal treatment for a (somewhat) general audience with a reasonable grasp of secondary school mathematics can be found in *The Book of Why* (Pearl & Mackenzie, 2018)

(a) The true model (b) The manipulated model

Figure 1: A causal DAG. X is a treatment, Y is recovery, and Z represents whether the patient is under 30. All variables are binary.

is worthwhile. There is however more to this. The fact that the age of patients influences their probability of getting a treatment is scientifically unfortunate. We would like to know what the effect of the treatment on the outcome actually is. But given only this observational data, we can't know this. We see in the data that young people recover more often, and that people who undergo the treatment also recover more often, but we cannot tease these causes apart. We don't know to what degree young people recover more often because they are more likely to get the treatment, or simply because of their youth.

Now, we will see that we can use the properties of our causal model beyond just using it as a nice visual representation. The DAG that we have constructed translates our assumptions about the causations into a more constrained causal model. If we make no assumptions about the data, and we want to use our model for predicting the probabilities of certain outcomes, we have to estimate the probability of every possible configuration of the variables. That is, we need to estimate the full joint probability $P(\mathbf{V})$, which for this example is $P(X, Y, Z)$. For binary variables, there are $2^{|\mathbf{V}|}$ possible configurations, such as $(X = 0, Y = 0, Z = 0)$, $(X = 0, Y = 0, Z = 1)$, etc., where $|\mathbf{V}|$ is the number of variables. The number of possible configurations therefore grows very quickly as the variable set $\mathbf{V}$ grows, and then many configurations may never even show up in the data, making it even harder to estimate their probabilities.

If we make no assumptions, with our variable set $\mathbf{V} = \{ X, Y, Z \}$ we get that

$$P(\mathbf{V}) = P(X, Y, Z) = P(X)P(Y \mid X)P(Z \mid X, Y),$$

using the fact that $P(A, B) = P(A)P(B \mid A) = P(B)P(A \mid B)$ for all variables or sets of variables A and B. The variables may be reordered in that expression, since joint probability is symmetric, but the probability of a configuration will always be a function of all the other variables. Note that the above expression is not a specific probability, but the definition of a probability distribution. It defines a function that takes a specific configuration and outputs a concrete probability, such as

$$P(X = 0, Y = 1, Z = 0) = P(X = 0)P(Y = 1 \mid X = 0)P(Z = 0 \mid X = 0, Y = 1).$$

Using our DAG model, however, we encode assumptions about the data. This simplifies our model significantly. DAG-based statistical models like this one are also known as Bayesian networks (BNs). The only difference between

BNs in general and causal models like this one are the causal implications we have given the model. In a BN model, we have that the joint probability simplifies significantly to

$$P(\mathbf{V}) = \prod_{X \in \mathbf{V}} P(X \mid \text{pa}(X)), \tag{1}$$

where $\text{pa}(X)$ is the set of *parents* of X, or nodes that have an arrow pointing to X. This puts a *constraint* on the model: the constraint that the joint probability must have this form. We can use the formula to find that the model in Figure 1a factorises as $P(X, Y, Z) = P(Z)P(X \mid Z)P(Y \mid X, Z)$.

This can dramatically decrease the number of probabilities we need to estimate, because of the following. In the model without assumptions, we had to estimate $P(X = 0 \mid \mathbf{V} \setminus X)$. That is, for each variable, X, we need to estimate for each configuration of the remaining variables what its probability is. Now, in the BN model, we only need for each variable one estimate for each configurations of its parents, which is usually much fewer. An estimate like this is called a *parameter* of the model. For example, a parameter of the assumptionless model might be $P(X = 0 \mid Y = 1, Z = 0) = 0.2$, while a parameter of the BN model might be $P(X = 0 \mid Z = 0) = 0.3$. Remember that in an unconstrained model, we need to estimate $2^{|\mathbf{V}|}$ parameters: the parameter count is exponential in the total number of variables. In a BN, the parameter count is only exponential in the maximum *in-degree*: the number of parents of the variable with the most parents.

More formally, we can identify the probability of a patient recovering *given that* they received the treatment: $P(Y = 1 \mid X = 1)$. But what we want is to identify the probability of a patient recovering in the hypothetical situation in which age does not influence the probability of getting the treatment, or equivalently, the probability of recovering if we *forced* someone to undergo the treatment. In terms of the model, this means that we don't observe the distribution of X from the data, but set the value ourselves. Let's call the value we set it to x. This would correspond to the causal representation in Figure 1b, with the edge $Z \to X$ removed, since X now has no causes except for our decision. The corresponding distribution, which we will call P_m, factorises as $P_m(X, Y, Z) = P_m(Y \mid x, Z)P_m(Z)$. The factor corresponding to X has been crossed out, or more technically, the distribution has been divided by the factor $P_m(X \mid Z)$. This makes sense intuitively, because there is no longer any meaningful probability of the value of X: we have set the value of X, so the probability of it having that value is 1.

To be able to identify causal effects like the probability of recovery given one if forced to undergo treatment, normally we use the scientific practice of *interventions*: we essentially do force some people to undergo the treatment, and some not to, in randomised controlled trials, and other experimental designs. But these trials are costly and time-consuming, and often even impossible because of ethical reasons. We cannot, for example, force people to smoke for the sake of an experiment. Luckily, though, it turns out that using graphical causal

analysis, we *can* identify this *interventional* probability without actually performing a costly intervention. We call this probability $P(Y = 1 \mid do(X = 1))$, using the so-called *do*-operator. Now, we use the manipulated model in Figure 1b, and we see that $P(Y = 1 \mid do(X = 1)) = P_m(Y = 1 \mid X = 1)$. In words: the probability of recovery ($Y = 1$) when forced to undergo treatment ($X = 1$) in the original model corresponds to the probability of recovery *given* that one underwent treatment in the manipulated model. Note that probabilities for P can be estimated from the data, because the data was generated from it, while probabilities for P_m cannot be estimated, since it did not generate the data.

M. Glymour et al. (2016) show that our probability of interest comes out to

$$\begin{aligned} P(Y = 1 \mid do(X = 1)) &= P_m(Y = 1 \mid X = 1) \\ &= \sum_z P_m(Y = 1 \mid X = 1, Z = z) P_m(Z = z) \\ &= \sum_z P(Y = 1 \mid X = 1, Z = z) P(Z = z), \end{aligned}$$

using an average over the possible values z of Z (in this case 0 and 1). The last step is allowed because the factors involved stay the same between the two different models. We have now expressed an interventional probability completely in terms of ordinary conditional probabilities that we can estimate from observational data. In other words, we can use (observational) data from the world in which Figure 1a is the true model, and measure the effect of the treatment as if we had performed an intervention and Figure 1b was the true model.

Since we have identified the causal effect $P(Y = 1 \mid do(X = 1))$, we can say that this effect is *identifiable*. The specifics of how to identify arbitrary causal effects is beyond the scope of this thesis, but we will note that it is possible to identify any effect that is identifiable, by using the *do*-calculus (Pearl, 2009) or the ID algorithm (Tian & Pearl, 2002). In DAG models, like these ones, all thinkable effects are identifiable.

2.1.2 Unmeasured confounders

Sometimes we know or suspect that data is generated by a process where a variable is involved that we have not measured or cannot measure and that has a causal effect on several of the measured variables in our model, so that it obscures the causal relations between them. We call these variables *unmeasured confounders*, or also *latent confounders*. These confounders can equally well be modelled by a DAG. We can take as an example again the models in Figure 1. Let's say now that Z is not age, but some genetic factor that we cannot measure. This factor, just like age, has an effect both on probability of recovery and on the probability of undergoing the treatment. Interpreted in this way, Figure 1 now shows *latent DAGs*.[2]

[2]Formally, latent DAGs are often defined as having separate sets of measured nodes ($\mathbf{V}$) and unmeasured nodes ($\mathbf{U}$).

In these latent DAGs, it is not always possible to identify all causal effects, because some of the (observational) probabilities that we would need cannot be estimated, since the relevant variables are not measured in our data. Graphs in which we *can* identify all effects are called *everywhere identifiable*, which is a very desirable property.

2.1.3 Linear-Gaussian models

Up until now we have – for ease of exposition – only used binary variables. Causal analysis over categorical variables with more than two possible values works mostly the same, it is generally only a bit more notationally cumbersome. We also have to estimate a few more parameters, since it no longer suffices to estimate the probability of 0 and infer the possibility of 1, but we must estimate the probability of $k-1$ values, if the number of possible values of the variable is k.

Many variables, however, are inherently numerical. Age, for example, is really a continuous variable, being able to take any real value greater than or equal to 0. We could imagine continuous alternatives for our other two example variables as well. For example, X could be dosage of the drug, and Y might be some numerical measure of quality of life after treatment. Structural causal modelling is also possible with continuous variables, but it does work differently in some important ways.

Apart from assuming binary variables, we have made no parametric assumptions about the types of distributions of the variables, nor about the way in which the variables are related to each other. This is standard when dealing with categorical variables. When dealing with continuous variables, such assumptions *are* very often made, though it is not required. This is because not making assumptions would make the space of parameters absolutely immense. A very commonly studied case, therefore, is the *linear-Gaussian* case, in which each variable individually follows a normal (Gaussian) distribution, and the value of each variable can be written as a linear equation in terms of its parents. This is the type of model we investigate in this work, and it is the only type of continuous causal model we will discuss. See Section 2.2.3 for an example of the definition of a linear-Gaussian model.

An important advantage of linear-Gaussian models over non-parametric categorical models is that the number of parameters involved is much smaller. All that is necessary to fully define a linear-Gaussian probability is the distributions (mean and standard deviation) of each of the individual variables, and for each edge an "edge weight" specifying how much one variable influences another. Remember that in the categorical case, the number of parameters is exponential in the maximum in-degree of the DAG model.

2.1.4 What actually *is* causality?

At the end of this introductory section on causal analysis, there remains one very important omission: we have not yet given a clear definition of what we

actually *mean* when we say that one variable causes another. This can be quite a deep philosophical question. For the purposes of this work, however, we will work with a rather simple probabilistic notion of causality. Intuitively, we say that a variable X is a *cause* of another variable Y, or *influences* Y, if forcing the value of X to some value (intervening on X) (significantly) changes the probability distribution of variable Y. Formally, X is a cause of Y if and only if $P(Y) \neq P(Y \mid do(X))$ in the true distribution.

2.2 Causal discovery

In a lot of cases, it is quite clear which variable causes which. We know that rain causes the road to be wet, and that road wetness does not cause rain. Similarly, if we see a correlation between age (below 30 or not) and recovery, as we did in the previous section, we can safely assume that it is youth that causes better chances at recovery, albeit possibly in some indirect way. Recovering from a disease does not make someone younger. This leads us clearly to the model in Figure 1a. Many other causations, however, are not so clear. Does mindful meditation improve professional performance? If we have no data, we don't know. If we do have data, and we see a positive correlation, we might still ask ourselves whether perhaps the more professionally successful are more likely to get involved in mindfulness. Perhaps because they can afford to take more spare time, reversing the direction of the causality, or perhaps there is some other variable that causes both: a confounder.

Firstly, of course, a graphical model can be useful simply by giving a visual representation of our current hypotheses. This can be especially useful if our domain contains more than, say, three variables, because a textual description then becomes hard to conceptualise. But a DAG model also provides the useful concept of d-separation. This is a property of variables that we can read from the graph that *should* correspond to statistical conditional independence. That is: if we see in the graph that X is d-separated from Y given Z, we expect to see that X is statistically independent of Y given Z (written $X \perp Y \mid Z$). If this independence does not hold in the data, then our model is not a good representation of the distribution of the data and must be tweaked (Verma & Pearl, 1990). We can then test whether it is a good representation again and again incrementally, until we are satisfied with our model.

If this assumption that d-separation implies (conditional) independence holds, we say that the graph is *Markov* to the data, or that the graph is an independence map or *I-map* of the true distribution. The other way round we say that the data is *faithful* to the graph if every (conditional) independence in the data has a corresponding d-separation in the graph. We can also say that the graph is a dependence map or *D-map*. (Some authors define faithfulness as the graph being both a D-map and an I-map – a perfect map, or P-map – but that is not the definition we will use.) We do not need the directions of the edges to always be in the direction of causation for them to model the dependences and independences in the data well. This is why Bayesian networks in general do not require causal edge directions.

To allow the reader to understand d-separation, we will now introduce some terminology. The important thing to remember about d-separation, however, is that it is a way to read independence from graphs, where variables in the "conditioning set" can block paths between the variables that are independent. If we have in a graph $A \to B$, we say that A is the *parent* of B, while B is the *child* of A. Since the nodes share an edge, we say that they are *adjacent*. A sequence of nodes that are adjacent to each other we call a *path*. (A path may involve the same node more than once, if it does not, it is a *simple* path.) If all edges on a path are in the same direction, it is a *directed* path. A node is a *collider* on a path if it has two incoming edges on that path. For example, on the path $A \to B \to C \leftarrow D$, C is a collider. This makes (B, C, D) together a *collider triple*. The set of *descendants* of a variable X, $\text{de}(X)$, is the set of variables that can be reached from it through a directed path by following the direction of the edges. This includes the singleton path with no edges: any node is also its own descendant. The set of *ancestors* of X, $\text{an}(X)$, is defined analogously, but the edges are followed against their direction. Now, we say that a set of variables $\mathbf{A}$ is *d-separated* from set $\mathbf{B}$ given $\mathbf{C}$ in graph $\mathcal{G}$ if every path between $\mathbf{A}$ and $\mathbf{B}$ contains a non-collider variable that is in $\mathbf{C}$ or the path contains a collider that has no descendants in $\mathbf{C}$. We write this as $\mathbf{A} \perp_{\mathcal{G}} \mathbf{B} \mid \mathbf{C}$. If $\mathbf{C}$ is the empty set, we simply write $\mathbf{A} \perp_{\mathcal{G}} \mathbf{B}$.

By comparing d-separations to dependences in the data and using expert knowledge, we can construct a causal graph "by hand". While this is often a good approach, certainly when much is already known about the domain of the data, we may sometimes wish for more automation. Especially when little is known about the interactions of variables in a domain this may become necessary. For this reason, algorithms have been designed that completely automate this task of *causal discovery*. These algorithms take in a dataset as input, and output a graphical representation of the causal structure, often in the form of a DAG or a related type of graph. It is this algorithmic causal discovery that interests us for this work. More specifically, it is causal discovery with latent (unmeasured) confounders that interests us, which we will explain now.

2.2.1 Unmeasured confounders in causal discovery

Many approaches to causal discovery assume that the measured variables account for all causal relations in the data. This is the assumption of *causal sufficiency*. This assumption is unfortunately often quite unrealistic. Unmeasured confounders are the norm rather than the exception in real-world data: there are an incredible amount of variables in the world that can influence the variables that you are interested in, and you are very unlikely to fully measure all of them in most cases. There are, therefore, also many methods for dealing with unmeasured confounders, as opposed to ignoring them. A well-known approach is the fast causal inference (FCI) algorithm (Spirtes, Glymour, & Scheines, 2001), which is an extension of the classical independence-testing based PC algorithm that can find between which variables confounding happens by defining different edge types for possible confounding relations. Huang, Low,

Xie, Glymour, and Zhang (2022) use rank constraints to find not only where confounding happens, but to actually find the confounding variables, and even a hierarchical structure between several unmeasured confounders. In quite the opposite direction there is the approach of Frot, Nandy, and Maathuis (2019), which does not attempt to find the confounders at all, but does first remove their influence, so that we can search for causal relations over only the observed variables without violating causal sufficiency.

The type of representation we are interested in lies somewhere between these approaches. We are interested in representing causal models with acyclic directed mixed graphs (ADMGs). These are a superclass of the traditional directed acyclic graphs (DAGs), with the addition of bidirected edges. An ADMG can capture the causal relations between the observed variables in the traditional way with directed edges, and represents confounding between two variables with a bidirected edge. ADMG models of causality are based on the assumption that the true causal situation can be represented by a DAG that explicitly contains the unmeasured confounders as additional nodes in the graph: a latent DAG.[3] Note that it is possible for there to be both a directed edge $A \to B$ *and* a bidirected edge $A \leftrightarrow B$. Such a structure is called a *bow*. As an example, Figure 3b contains two bows.

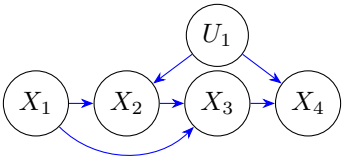
Every latent DAG can be associated with an ADMG through the operation of *latent projection* (Verma & Pearl, 1990). The idea of latent projection is to remove unmeasured (latent) variables from the graph, and replace the information they provide about confounding by a simple bidirected edge. Take as an example the DAG in Figure 2a and its latent projection shown in Figure 2b. The confounding variable U_1 is removed from the resulting latent projection, and the confounding information is represented in the bidirected edge between X_2 and X_4.

Definition 2.1. *A latent DAG $\mathcal{G}$ can be associated with its* latent projection *$\mathcal{G}^*$, which is the ADMG that we find when we apply the following procedure:*

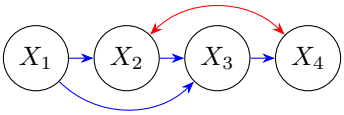
- *Initialise $\mathcal{G}^*$ with the nodes and directed edges of $\mathcal{G}$.*
- *Remove all unmeasured variables and their incoming and outgoing edges from $\mathcal{G}$.*
- *For any path of the form $A \to \cdots \to B$ in $\mathcal{G}$, with all intermediate variables being unmeasured, add an edge $A \to B$ to $\mathcal{G}^*$.*
- *For any path of the form $A \leftarrow \cdots \to B$ in $\mathcal{G}$ with all intermediate variables being unmeasured non-colliders, add a bidirected edge $A \leftrightarrow B$ in $\mathcal{G}^*$.*

Using this definition, we see that the latent DAG in Figure 2c also maps to the same ADMG in Figure 2b, with the chain of two confounders corresponding to the same single bidirected edge. The concept of d-separation in DAGs extends naturally to mixed graphs, only it is called *m-separation*. For m-separation,

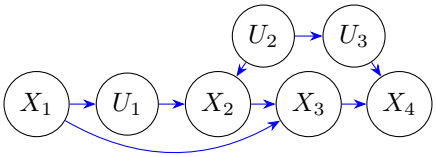
[3] Note that this assumption fails if the true causal structure contains feedback loops, since cycles are by definition disallowed in DAGs (directed *acyclic* graphs).



(a) Latent DAG



(b) acyclic directed mixed graph (ADMG)



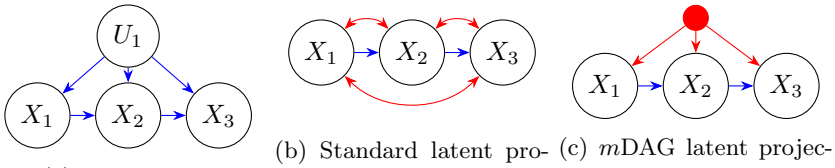
(c) Latent DAG with three unmeasured variables

Figure 2: Latent projection

bidirected edges also count as incoming edges for determining which nodes are collider nodes, so that B is also a collider on the paths $A \leftrightarrow B \leftarrow C$, $A \rightarrow B \leftrightarrow C$ and $A \leftrightarrow B \leftrightarrow C$. Otherwise the definition is the same. ADMGs are a superclass of DAGs, since DAGs can be seen as ADMGs without bidirected edges: DAGs $\subset$ ADMGs.

A latent DAG is more informative than its corresponding ADMG, but it is harder to find the correct one. Let's assume that the true model generating our data is a latent DAG model. The distribution that we obtain from the data is then the distribution with the unmeasured variables marginalised out. For any such marginal distribution, an infinite number of latent DAGs could have produced it, so it is hard to find the true one, at least without making additional parametric assumptions (assumptions about the types of distributions and relations between the variables, i.e. that variables are Gaussian, or non-Gaussian, or linearly related or not).

If we decide that we do not particularly care about how many confounding variables there are and where they should be in the model, we can still represent at least between which variables any confounding happens by searching only for an ADMG, which is more feasible. Intuitively, this information is already sufficiently interesting. A more general concept of latent projection that we will unfortunately not investigate, but that we think is still worth mentioning, is captured in marginalised directed acyclic graphs (mDAGs) (Evans, 2016). These more elegantly accommodate, for example, the possibility of one unmeasured confounder influencing more than two observed variables.



(a) Latent DAG (b) Standard latent projection (c) mDAG latent projection

Figure 3: mDAGs more elegantly capture confounding by one unmeasured variable on several measured variables.

As an example, take Figure 3a. The unmeasured variable U_1 influences all three measured variables. Standard latent projection would yield the ADMG in Figure 3b, where a single variable (U_1) corresponds to three bidirected edges. From this ADMG, it is impossible to see that these three bidirected edges had anything to do with each other. The mDAG latent projection, however, uses a single so-called "bidirected hyper-edge" to connect the three measured variables, as shown in Figure 3c. This makes it clear that there are not three separate (sets of) confounders, but that they share a confounder. mDAGs are a generalisation of ADMGs, so we have DAGs $\subset$ ADMGs $\subset$ mDAGs.

2.2.2 Formal graphical definitions

We will now define DAGs and ADMGs formally, though they have already been informally introduced. We will also define some important genealogical relations between nodes.

Definition 2.2. *A* directed acyclic graph *(DAG)* $\mathcal{G}(\mathbf{V}_\mathcal{G})$ *is a graph with a set of nodes* $\mathbf{V}_\mathcal{G}$ *and an edge set* $\mathbf{E}_\mathcal{G}$ *which contains ordered pairs of nodes which represent directed edges in the graph. If* $\mathbf{E}_\mathcal{G}$ *contains the ordered pair* (A, B)*,* $\mathcal{G}$ *contains an edge from* A *to* B*, which we may also write more visually as* $A \to B \in \mathbf{E}_\mathcal{G}$*. A DAG may not contain directed cycles. That is: there is no directed path from any node to itself except the trivial path involving no edges.*

To avoid notational clutter, we will leave out the subscript $\mathcal{G}$ when it is clear from context which graph we are referring to. In this work, we will refer to nodes in a graphical model and variables interchangeably with capital letters. That is, A, B, and C can be the nodes in the DAG $A \leftarrow B \rightarrow C$ as well as the variables that those nodes represent. ADMGs are defined formally as follows.

Definition 2.3. *An* acyclic directed mixed graph *(ADMG)* $\mathcal{G}(\mathbf{V}_\mathcal{G})$ *is a graph with a set of nodes* $\mathbf{V}_\mathcal{G}$*, a set of directed edges* $\mathbf{E}_\mathcal{G}$ *and a set of bidirected edges* $\mathbf{B}_\mathcal{G}$*. The set* $\mathbf{E}_\mathcal{G}$ *contains ordered pairs, as in DAGs, while the set* $\mathbf{B}_\mathcal{G}$ *contains unordered pairs which represent bidirected edges: edges that have arrowheads at both sides. For example, the unordered pair* $\{ A, B \} \in \mathbf{B}_\mathcal{G}$ *would represent the bidirected edge* $A \leftrightarrow B$*. As in DAGs, the directed edges may not form cycles.*

We say that two nodes are *adjacent* if there is some type of edge between them. A *path* is a sequence of nodes wherein each node is adjacent to the next

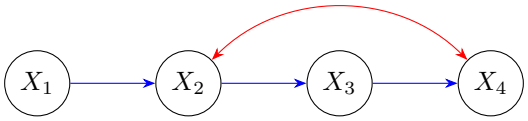


Figure 4: A BAP.

node. A trivial path containing only one node and no edges is also a path. A path is *directed* if all edges along the path point in the same direction. A path is *bidirected* if all edges along the path are bidirected edges. All nodes along a bidirected path are said to be *bidirected-connected*.

We use the standard genealogical sets of variables, which we define below in Definition 2.4. (Some of these have already been introduced.) In order, these are the *parents*, *children*, *ancestors*, *descendants*, *siblings*, and the *district* of A. Note that, because there is always a trivial path from A to A itself, with no edges, A is included in its set of ancestors and its set of descendants.

Definition 2.4. *For any variable A, we define the following genealogical sets:*

$$\begin{aligned}
\mathrm{pa}_{\mathcal{G}}(A) &=_{\mathrm{def}} \{\, B \in \mathbf{V}_{\mathcal{G}} \mid B \to A \in \mathbf{E}_{\mathcal{G}} \,\} \\
\mathrm{ch}_{\mathcal{G}}(A) &=_{\mathrm{def}} \{\, B \in \mathbf{V}_{\mathcal{G}} \mid A \to B \in \mathbf{E}_{\mathcal{G}} \,\} \\
\mathrm{an}_{\mathcal{G}}(A) &=_{\mathrm{def}} \{\, B \in \mathbf{V}_{\mathcal{G}} \mid \text{there is a directed path from } B \text{ to } A \,\} \\
\mathrm{de}_{\mathcal{G}}(A) &=_{\mathrm{def}} \{\, B \in \mathbf{V}_{\mathcal{G}} \mid \text{there is a directed path from } A \text{ to } B \,\} \\
\mathrm{sib}_{\mathcal{G}}(A) &=_{\mathrm{def}} \{\, B \in \mathbf{V}_{\mathcal{G}} \mid A \leftrightarrow B \in \mathbf{B}_{\mathcal{G}} \,\} \\
\mathrm{dis}_{\mathcal{G}}(A) &=_{\mathrm{def}} \{\, B \in \mathbf{V}_{\mathcal{G}} \mid \text{there is a bidirected path from } A \text{ to } B \,\}
\end{aligned}$$

We allow for these definitions to be applied disjunctively to sets, such that for example

$$\mathrm{pa}(\mathbf{A}) =_{\mathrm{def}} \bigcup_{A \in \mathbf{A}} \mathrm{pa}(A).$$

The induced subgraph $\mathcal{G}_{\mathbf{S}}$ of $\mathcal{G}$, with $\mathbf{S} \subseteq \mathbf{V}_{\mathcal{G}}$, is the graph containing only the nodes in $\mathbf{S}$ and only the edges that have both end-points in $\mathbf{S}$. If $\mathcal{G}_{\mathbf{S}}$ contains only one district, we say that the set $\mathbf{S}$ is bidirected-connected in $\mathcal{G}$.

2.2.3 Linear-Gaussian acyclic path diagrams

The algorithms we develop are based on the greedyBAPs algorithm by Nowzohour et al. (2017), so it is convenient to show how they define linear-Gaussian models. For each variable X_i in the graph $\mathcal{G}$, its value is given by

$$X_i = \sum_{X_j \in \mathrm{pa}_{\mathcal{G}}(X_i)} B_{ij} X_j + \epsilon_i.$$

The graph in Figure 4 would be represented as follows:

$$\begin{aligned} X_1 &= \epsilon_1 \\ X_2 &= B_{21}X_1 + \epsilon_2 \\ X_3 &= B_{32}X_2 + \epsilon_3 \\ X_4 &= B_{43}X_3 + \epsilon_4. \end{aligned}$$

Here, each variable is a linear combination of its parents (only singular parents in this case) and an ϵ term that includes its variance and the covariance with other variables caused by bidirected edges. Means are assumed to be normalised to 0 (though that is not a feature of linear-Gaussian models in general). We have then that $(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4)^T \sim \mathcal{N}(\mathbf{0}, \Omega)$, with

$$\Omega = \begin{pmatrix} \Omega_{11} & 0 & 0 & 0 \\ 0 & \Omega_{22} & 0 & \Omega_{24} \\ 0 & 0 & \Omega_{33} & 0 \\ 0 & \Omega_{24} & 0 & \Omega_{44} \end{pmatrix}.$$

To phrase this in a perhaps more intuitive way, for each i-th and j-th node, the covariance $\mathrm{cov}(\epsilon_i, \epsilon_j) = \Omega_{ij}$.[4]

Since all means are zero, the only parameters are $\Omega_{11}, \Omega_{22}, \Omega_{33}$ and Ω_{44}, representing the variances of the variables, B_{21}, B_{32} and B_{42}, representing the directed edges, and Ω_{24} representing the bidirected edge. The B parameters can, like the Ω parameters, be seen as entries in a B matrix. The set of parameters for a graph $\mathcal{G}$ we call θ_G. We can obtain from these parameters the covariance matrix of the observed variables

$$\Sigma = (I - B)^{-1}\Omega(I - B)^{-T},$$

which at each entry Σ_{ij} contains the covariance of variables X_i and X_j. From such a model, we can obtain the *total causal effect* of a variable X_j on a variable X_i, $E_{ij} = \frac{\partial}{\partial x}\mathbb{E}[X_i \mid do(X_j = x)]$, by taking

$$E_{ij} = ((I - B)^{-1})_{ij}.$$

This gives us the causal effects matrix E.

This type of model definition is called a linear structural equation model (SEM), because the relations are linear, and it defines a structural model through a set of equations. More generally, causal models defined by equations, as opposed to the Bayesian network definition we saw before, are called structural causal models (SCMs). Variables in SCMs in general need not be normally distributed or be related linearly, they only need to be defined by equations in terms of their parent variables. This work concerns only the (very common) case of linear SEMs.

[4]Note that this works for the variances of the individual variables as well, since the covariance of a variable with itself is its variance: $\mathrm{cov}(X, X) = \mathrm{Var}(X)$ for all variables X. The variances are therefore on the diagonal of Ω.

2.2.4 Constraint-based methods

There are two main types of approaches within causal discovery: constraint-based approaches and score-based approaches. Constraint-based approaches such as PC and FCI (Spirtes et al., 2001) generally start a search at a complete undirected graph (with an undirected edge between every pair of variables), and remove and orient edges to arrive at some type of partially directed acyclic graph (a DAG in which some edges have no defined direction) with as many edges oriented as we can tell from the data. To do this, they use constraints in the form of conditional independence tests, and they exploit structural properties of the graph. In a sense, these constraint-based methods are similar to the approach of constructing a graph by hand mentioned above in that they both examine conditional independences. They can also take expert knowledge into account. These methods check for relevant conditional independences more systematically.

2.2.5 Score-based methods

There are also score-based algorithms, which are the focus of our research. These algorithms try to maximise a *score function* which represents how well the graph captures the data. Many of them do this searching through the space of graphs by adding edges, deleting edges or switching edge directions. The most important part of the score function is often the log likelihood of the data given the graph. The likelihood of the data is (informally) the probability that the current model would have generated the given data: $P(data \mid model)$. The log likelihood is just the logarithm of likelihood, with any base you like, for computational reasons.[5] Score functions also incorporate a penalty term on complexity to avoid overfitting with complex models. If two models explain the data equally well, we prefer the simpler model, with fewer edges, and a simpler model is often preferable even if its likelihood is slightly lower than a more complex model. Two important examples of such scores are the Akaike information criterion (AIC) (Akaike, 1974) and the Bayesian information criterion (BIC) (Schwarz, 1978). A more extensive overview of score functions can be found in Vowels, Camgoz, and Bowden (2022).

Local search Essentially, many score-based algorithms define the problem of causal discovery as an instance of *local search*. They do this, because the number of DAGs (or superclasses of dags) with d nodes is super-exponential in d, so exhaustive search is infeasible. For local search, we need a domain of solutions, in our case, the set of graphs of the type we are interested in, a way to evaluate solutions, in our case, the score function, and a neighbourhood

[5] Specifically, since probabilities often multiply, they tend to differ from each other exponentially rather than linearly. Because of this, they often become very very small, such that the floating point representation computers use for real numbers cannot capture these probabilities well. Taking a logarithm makes the differences linear and makes the probabilities representable by floating point. In fact, one generally works only with logarithms of probabilities in computational settings, as opposed to just taking the logarithm after the calculation.

relation or function that defines local moves from one solution to another. This is a function mapping one graph to another of the same type. For DAGs, we usually say that $\mathcal{G}' \in \text{neighbours}_{\text{DAG}}(\mathcal{G})$ if and only if $\mathcal{G}'$ is also acyclic, and either

- $\mathcal{G}'$ contains one edge that $\mathcal{G}$ does not, but is otherwise identical (edge addition), or
- $\mathcal{G}'$ is missing one of the edges $\mathcal{G}$ has, but is otherwise identical (edge deletion), or
- $\mathcal{G}$ and $\mathcal{G}'$ differ only in the direction of one edge (edge change).

The simplest local search algorithm is *greedy search*. We will explain greedy search precisely, because our work is based on it. Pseudocode for general greedy search in the space of graphical statistical models is shown in Algorithm 1. The graph to start from, $\mathcal{G}_{\text{start}}$, is often the empty graph, with no edges between any of the variables, though this is not required. A neighbourhood function needs to be provided, as well as a score function that takes a graph and a set of parameters and outputs a score. The greedy search algorithm then, at every step, produces all neighbours of the current graph, scores them, and continues on from the best-scoring graph, until no score improvement is possible anymore.

Algorithm 1 The general form of greedy graph search algorithms. $\mathcal{G}_{\text{start}}$ is the graph to start the search from, score is the score function, neighbours is the specific neighbourhood function to use and ε is the threshold specifying the minimum score difference for improvement. Either the dataset $\mathbf{X}$ can be provided (in the categorical case) or the covariance matrix Σ (in the linear-Gaussian case).

procedure GreedyGraphSearch($\mathcal{G}_{\text{start}}, \mathbf{X}||\Sigma, \text{neighbours}, \text{score}, \varepsilon$)
  $i \leftarrow 1$
  $\mathcal{G}_i \leftarrow \mathcal{G}_{\text{start}}$
  $\Delta_{\text{score}} \leftarrow \infty$
  while $\Delta_{\text{score}} > \varepsilon$ **do**
    $Candidates \leftarrow \text{neighbours}(\mathcal{G}_i)$
    for all candidates $\mathcal{G}_c \in Candidates$ **do**
      $\hat{\theta}_{\mathcal{G}_c} \leftarrow$ EstimateParameters($\mathcal{G}_c, \mathbf{X}||\Sigma$)
    $\mathcal{G}_{i+1} \leftarrow \arg\max_{\mathcal{G}_c \in Candidates \cup \{\, \mathcal{G}_i \,\}} \text{score}(\mathcal{G}_c, \hat{\theta}_{\mathcal{G}_c})$
    $\Delta_{\text{score}} \leftarrow \text{score}(\mathcal{G}_{i+1}, \hat{\theta}_{\mathcal{G}_{i+1}}) - \text{score}(\mathcal{G}_i, \hat{\theta}_{\mathcal{G}_i})$
    $i \leftarrow i + 1$
  return $\mathcal{G}_i$

Greedy search can often work quite well for its simplicity. However, it is prone to getting stuck in local maxima in the score function. It may reach a graph $\mathcal{G}$ that has no neighbours $\mathcal{G}'$ that score better than it, while there does exist a graph $\mathcal{H}$ that scores (much) better than $\mathcal{G}$, but it simply is not reachable from the $\mathcal{G}$ through local moves and greedy choices.

Score functions While our main research question is largely agnostic of whether we use categorical models or linear-Gaussian models, it should be noted that there are some important differences for causal discovery between the two. The difference lies mainly in the score function. As mentioned, score functions generally have the form of

$$\text{score}(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) = \mathcal{L}(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) - \text{parameter penalties},$$

with $\mathcal{L}$ the log-likelihood function and $\hat{\theta}_{\mathcal{G}}$ a set of parameters. For example, the well-known BIC score (Schwarz, 1978) is defined as

$$\begin{aligned} BIC(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) &=_{\text{def}} -2\mathcal{L}(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) + |\hat{\theta}_{\mathcal{G}}| \\ -BIC(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) &=_{\text{def}} 2\mathcal{L}(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) - |\hat{\theta}_{\mathcal{G}}|. \end{aligned}$$

The first line is the usual definition here, because lower BIC scores are better, but in this work we will use negative score functions where a higher (less negative) value is better, so the second line shows this form better.[6]

The hat above $\hat{\theta}_{\mathcal{G}}$ is because parameters do not come with the graph, their values have to be estimated from the data. The graph and associated type of statistical model dictate which parameters have to be estimated exactly. For example, in our example from Figure 1a, using an ordinary Markov model, we need to estimate $P(Z = 0), P(X = 0 \mid Z = 0), P(X = 0 \mid Z = 1), P(Y = 0 \mid X = 0, Z = 0), P(Y = 0 \mid X = 0, Z = 1)$, et cetera. For discrete models, as mentioned, the number of parameters to estimate is exponential in the maximum in-degree (2 in this case), which can be quite a lot. We also need the whole dataset to estimate this every time. For linear-Gaussian models, this is quite different. We don't need the data, we only need the covariance matrix, which we can generate from the data once, after which we don't need the data anymore. This is why in Algorithm 1, either the data $\mathbf{X}$ can be passed in, or the covariance matrix Σ.

Decomposition Fitting parameters can be quite an expensive operation, and therefore scoring a graph can be expensive. This is a problem for score-based causal discovery methods, because we generally need to assess the score of many graphs before we reach our best-scoring graph. Luckily, we don't need to re-score every graph from the ground up every time we make a small change. We can make use of the factorisation property in (1). Because an ordinary Markov model is defined in terms of individual factors for each node, the likelihood function can also be expressed as a combination (sum or product) of individual terms, each term corresponding to a single node and its parents. That means that when an edge is changed, at most two terms have to be recalculated, while the rest can be cached, so we don't have to re-calculate the score for the entire graph after every edge change. For models defined over ADMGs this works slightly differently: generally the terms correspond to districts, not singular

[6] The score is negative, because the likelihood $P(\mathcal{G} \mid \text{data})$ is a probability, so in the range [0, 1], and a logarithm of a value between 0 and 1 is always negative.

nodes. This means that a bit less computation will be saved, but it's still a big difference, as long as districts don't get too large.

2.2.6 Continuous optimisation methods

A more modern development is the rise of continuous optimisation approaches to causal discovery. Standard score-based methods enforce DAGness by having a discrete acyclicity constraint (or other-type-of-graphness with different constraints): a graph is either a DAG or it is not. Continuous optimisation methods instead transform this constraint into a continuous numerical one (a number that captures how "DAGgy" a graph is, further from 0 is less DAGgy) so that continuous optimisation can be used to find the best graph instead of local search. Often, these methods use neural networks. For more comprehensive overviews of causal discovery methods, see C. Glymour, Zhang, and Spirtes (2019), who focus on traditional constraint-based and score-based methods, and Vowels et al. (2022), who focus on continuous optimisation methods.

2.2.7 Model equivalence

One of the most important problems for causal discovery from observational data is the concept of *model equivalence*. In score-based methods, we want to find a graph that scores best given the data, based on its likelihood ($P(\text{data} \mid \text{graph})$). Unfortunately, the likelihood of different graphs is often the same. Then, if their penalty terms are also the same (because they imply the same number of parameters), we cannot distinguish them with our score function. This happens if the two graphs are equivalent according to the type of model that we use the graph to define. If we find two (or more) equivalent graphs as our best graph, then either of them explains the data equally well statistically, and we have only our own judgement to decide which is really most likely to have generated the data.

A graphical causal model uses a graph to define a model (as the name implies). A model puts constraints, such as (conditional) independences, on the joint distribution. The model also specifies a set of parameters of which the value needs to be estimated. Two models are equivalent if they impose the same constraints. It is possible to define different types of models over the same graph. We will go into these differences in this section.

Ordinary Markov equivalence Perhaps the simplest case is that of ordinary Markov equivalence over DAG models with categorical variables. It is the case we described in Section 2.1.1. The constraints that an ordinary Markov model implies are the (conditional) independences that we read from the d-separations in the DAG. If these are the same between two graphs, the models are equivalent.

For DAGs, we have a simply checkable graphical characterisation of this equivalence. Verma and Pearl (1990) show that two DAGs are Markov equivalent if and only if they have the same skeleton and the same v-structures.

For a graph $\mathcal{G}'$ to have the same skeleton as a graph $\mathcal{G}$ means that if and only if there is an edge between two nodes in $\mathcal{G}$, there is also an edge between those nodes in $\mathcal{G}'$, whatever its direction. Essentially, the graphs have the same edges if we ignore the direction of the edges. *v-structures*, sometimes also called *unshielded colliders*, are collider triples where the two parents are not adjacent. For example: the collider triple $A \to B \leftarrow C$ is also a v-structure, unless an edge $A \to C$ or $C \to A$ is also present. (That would turn the 'v' into a triangle.)

A complete set of DAGs that are Markov equivalent to each other, a Markov equivalence class (MEC), can be efficiently represented by a complete partial directed acyclic graph (CPDAG). The CPDAG has a directed edge where all DAGs in the equivalence class agree on the direction, and an undirected edge where some DAGs disagree on the direction. Of course, two different DAGs in the same MEC do not have the same causal interpretation, but we would not be able to tell which interpretation is correct from the data. CPDAGs are a generalisation of DAGs in a very similar but still different way from ADMGs, so we have CPDAGs $\supset$ DAGs $\subset$ ADMGs $\subset$ mDAGs.

For DAGs, it is possible to use equivalence and CPDAGs to our advantage. This is done by the well-known greedy equivalence search (GES) algorithm, which was originally developed by Meek (1997) and proved to find the optimal solution (under certain conditions) by Chickering (2003). The algorithm searches in the space of CPDAGs (or equivalently the space of MECs). The precise way in which the neighbourhood relation is defined is a bit involved, but the general idea of it is that all DAGs in the equivalence class are instantiated, an edge addition or removal is performed on them, resulting in a new DAG, and the final result is the CPDAGs corresponding to that new DAG. GES actually first has a forward phase, where edges are added until a local maximum of the score function is reached, and then a backward phase where edges are removed until a local maximum is reached again. Variants of the GES algorithm that are more efficient in different ways are introduced in Chickering and Meek (2015) and Chickering (2020). The moves defined by the neighbourhood relation given by Chickering (2003) can also be used in different local search procedures, for example with the ant-colony optimisation meta-heuristic (Daly & Shen, 2009).

Nested Markov equivalence ADMG models can encode ordinary Markov constraints through m-separation. However, it has long been known that they can imply more constraints on the joint probability distribution (Robins, 1986, Verma and Pearl, 1990). Shpitser, Evans, Richardson, and Robins (2014) and Richardson et al. (2023) (among others) call these constraints *nested Markov constraints*, and use them to define *nested Markov models*, along with parameterisations[7] for those models. Essentially, a nested Markov constraint is an independence that holds in a version of the original graph that has been altered: some number of the nodes have been "fixed", an operation similar to

[7]Definitions of the parameter set, and how to map from the parameters to a probability distribution.

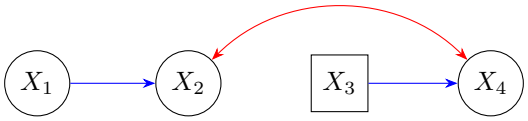


Figure 5: The conditional acyclic directed mixed graph (CADMG) resulting from fixing X_3 in Figure 2b.

do-operator interventions like in Figure 1b.[8]

To explain nested Markov constraints, we will come back to Figures 2a and 2b (page 12). We see that no d-separations are encoded among the observed variables in the DAG Figure 2a, and that (therefore, as intended by the definition of latent projections) no m-separations are encoded in the ADMG in Figure 2b. These graphs therefore encode no regular (conditional) independences. However, it turns out that these graphs do still encode some constraints. For example, if we "fix" X_3 in Figure 2b, i.e. give it a set value (equivalent in this case to intervention), we get the graph in Figure 5 (the fixed node is indicated as a square).[9] In this graph, X_1 is independent of X_4 (given X_3, though also marginally). We can observe such "post-truncation independences" in observational data corresponding to such a graph. The authors call an ADMG with fixed nodes like the one in Figure 5 a conditional acyclic directed mixed graph (CADMG).[10] A post-truncation independence like the one described above is an example of a nested Markov constraint. If two graphs impose the same nested Markov constraints on the distribution, we say that they are *nested Markov equivalent*. Section 2.4 goes into more depth on fixing and other nested Markov definitions, such as when a node is allowed to be fixed.

Unfortunately, no graphical characterisation of nested Markov equivalence exists, as we do have for ordinary Markov models in the form of the requirement of having the same skeleton and v-structures. Of course, this means that no equivalent of the CPDAG, which elegantly shows the equivalence class, exists for this model. This is a long-standing open problem.

Shpitser, Richardson, Robins, and Evans (2012) estimate that about 25% of ADMGs with four nodes have such post-truncation independences, so taking them into account may help us distinguish models we could not have distinguished through ordinary Markov (non)equivalence in a significant amount of cases. Shpitser et al. (2018) show that linear SEMs also obey the nested Markov property. That is, all nested Markov constraints over a graph $\mathcal{G}$ are also present in a linear-Gaussian model over that graph $\mathcal{G}$, so nested Markov models are not only relevant in the discrete case.

Evans (2018) shows that nested Markov models capture all equality con-

[8] Specifically, fixing generalises marginalizing out a variable and *do*-operator intervention by dividing out the factor corresponding to a variable from the factorisation.

[9] There are rules about which nodes are "fixable" at which time, but they involve rather complex graphical definitions that we will not reproduce here.

[10] CADMGs are a strict generalisation of ADMGs: every ADMG is a CADMG, just with no nodes fixed.

straints for discrete models, showing that there is no more fine-grained notion of equivalence possible for discrete model, as long as we don't want to consider inequality constraints, which for the purposes of this work, we don't.[11]

Distributional equivalence For linear-Gaussian models, as opposed to discrete models, there are more fine-grained notions of equivalence available. For example, van Ommen and Mooij (2017) make progress on algebraic equivalence constraints, which are polynomial constraints on the covariance matrices of linear structural equation models.

Nowzohour et al. (2017) discuss the case of BAPs: ADMGs without bows. Let $\mathcal{S}_\mathcal{G}$ be the set of possible covariance matrices over a BAP $\mathcal{G}$, and $\bar{\mathcal{S}}_\mathcal{G} \subset \mathcal{S}_\mathcal{G}$ the set of normalised covariance matrices over $\mathcal{G}$, where $\bar{\mathcal{S}}_\mathcal{G} = \{\Sigma \in \mathcal{S}_\mathcal{G} \mid \Sigma_{ii} = 1 \text{ for all } X_i \in \mathbf{V}_\mathcal{G}\}$. They define two BAPs $\mathcal{G}$ and $\mathcal{G}'$ to be *distributionally equivalent* if $\bar{\mathcal{S}}_\mathcal{G} = \bar{\mathcal{S}}_{\mathcal{G}'}$. This is yet more fine-grained than algebraic equivalence.

As with nested Markov equivalence, no graphical characterisation of distributional equivalence is known. Nowzohour et al. (2017) do, however, provide some necessary and some sufficient conditions for distributional equivalence. They prove that any two BAPs with the same skeleton and the same collider triples must be equivalent (sufficient condition) and that two BAPs cannot be equivalent if they do not have the same skeleton and v-structures (necessary condition). This means concretely that for any BAP $\mathcal{G}$, all other graphs with the same skeleton and collider triples are definitely equivalent, and more graphs may be equivalent, as long as they have the same v-structures.

They use these results to restrict the search space in a search for *empirically equivalent* graphs. That is, they find a graph that fits the data, then to find equivalent graphs, they start by adding all graphs with the same skeleton and collider triples to the empirical equivalence class. They then greedily make small changes to these graphs that do not result in a difference of v-structures, and score these models. If they score the same (or at least very similarly) as the original found graph, they are added to the empirical equivalence class. This algorithm is the first of its kind for this type of graph, and provides some clarity in the absence of a full characterisation. It is unfortunately a lot slower and less reliable than the quickly constructed and mathematically proven CPDAGs.

2.3 Limitations of causal discovery

We think it is important to give the reader a bit of perspective on the usefulness of causal discovery, since in these explanations of the theoretical possibilities we run the risk of overselling the actual possibilities. Current methods for causal discovery can certainly be useful, but their use absolutely has its limits. If causal

[11] All constraints that we have considered up to now, and that we will consider later, have been and will be equality constraints: constraints that can be written as equalities. For example, the post-truncation independence we will discuss in the next paragraph can also be written as $\frac{\partial}{\partial x_1} \sum_{x_2} P(x_4 \mid x_1, x_2, x_3) P(x_2 \mid x_1) = 0$. Models can however also imply constraints that can only be written as inequalities. Relatively little is known about these, and we will not consider them further for this work.

discovery were a completely solved problem, that would practically mean that much of science would be solved. This would be amazing, but unfortunately, causal discovery is not a solved problem. It is an incredibly hard problem. In many cases, the necessary information is simply not there in the data. And even if the information is there, many algorithms are intractable at larger numbers of variables, and still may be slow at smaller numbers of variables. Algorithms can also be very sensitive to parameter tuning. So causal discovery is useful, but we cannot expect to just feed in a bunch of data to an algorithm that then outputs the complete and true causal structure of the variables any time soon, if ever.

Another important warning to give to anyone intending to use graphical causal models is that the main assumption we make is not a trivial one. We assume that the data we observe is generated by a process that can be modeled as a DAG, but that is not always the case. For example, sometimes, real-world processes involve feedback loops where one variable influences another, which in turn influences the first. This would correspond to a cycle in a graphical model, which is disallowed in DAGs because they are hard to deal with. It is in general not always the case that a DAG exists that is Markov and faithful to the data, though we often assume that there is such a graph in causal discovery.

2.4 Maximal Arid Graphs

In this study, we develop greedy search algorithms over the class of maximal arid graphs (MArGs), a subclass of ADMGs, introduced by Shpitser et al. (2018). The *arid* part of its name implies that it is everywhere identifiable. The *maximal* part means that any missing edge in a MArG implies a nested Markov constraint. The authors show that every ADMG $\mathcal{G}$ has an associated MArG $\mathcal{G}^\dagger$ through the operation of *maximal arid projection*. This MArG $\mathcal{G}^\dagger$ is also nested Markov equivalent to $\mathcal{G}$. Since, as mentioned, Evans (2018) showed that nested Markov models capture all equality constraints for discrete models, it can be argued that, for a discrete model, instead of any $\mathcal{G}$, we may as well use its maximal arid projection $\mathcal{G}^\dagger$, since it is equivalent, and everywhere identifiable. For those familiar, MArGs are analogous in a way to maximal ancestral graphs (MAGs). What MAGs are to ancestral graphs, MArGs are to ADMGs in general. The maximal arid projection can also preserve much more of the causal structure of a graph than its corresponding MAG.

To define the maximal arid projection, we need a few more graphical concepts relevant to nested Markov models. That is, for the maximal arid projection, we need the reachable closure, and for the reachable closure, we need to specify what fixing orders are allowed. Firstly, we properly define a CADMG $\mathcal{G}(\mathbf{V}, \mathbf{W})$ to be an ADMG with a set of random nodes $\mathbf{V}$ and a set of fixed nodes $\mathbf{W}$.[12] The siblings of fixed vertices do not overlap with their parents: $\text{sib}_\mathcal{G}(W) \cap \text{pa}_\mathcal{G}(W) = \emptyset$ for all $W \in \mathbf{W}$. A node $V \in \mathbf{V}$ is *fixable* if and only if its district and

[12]To update the overview of graph classes, we now have CPDAGs $\supset$ DAGs $\subset$ MAGs $\subset$ MArGs $\subset$ BAPs $\subset$ ADMGs $\subset$ CADMGs.

descendants do not overlap (except for itself): $\text{dis}_{\mathcal{G}}(V) \cap \text{de}_{\mathcal{G}}(V) = \{ V \}$. We define a fixing operation $\text{fix}_X(\mathcal{G})$ that takes a CADMG $\mathcal{G}(\mathbf{V}, \mathbf{W})$ and a fixable node $X \in \mathbf{V}$ and returns a new graph $\mathcal{G}(\mathbf{V} \setminus \{ X \}, \mathbf{W} \cup \{ X \})$ where all edges with arrowheads at V are removed.[13] Sometimes, one node cannot be fixed before another node is fixed. This means that not all *fixing orders* over a set of nodes to fix are valid. Shpitser et al. (2018) do prove, however, that any fixing order over the same set of nodes will always result in the same CADMG. If, for some set of nodes $\mathbf{X} \subseteq \mathbf{V}$, there exists a fixing order for the remaining nodes $\mathbf{V} \setminus \mathbf{X}$, that, if fixed, leaves only $\mathbf{X}$ unfixed/random, we say that $\mathbf{X}$ is a *reachable set*. To illustrate: we may have a CADMG $\mathcal{G}(\{ A, B, C, D \}, \emptyset)$, where C cannot be fixed before A and B are. Then (A, B, C) or (A, C, B) would be valid fixing orders, while (A, C, B), for example, is not. Fixing the nodes in the two valid fixing orders will however result in the same CADMG. From this we know that, for example $\{ C, D \}$ and $\{ D \}$ are reachable, but $\{ A, D \}$ is not.

Every set of variables $\mathbf{S}$ in a CADMG $\mathcal{G}(\mathbf{V}, \mathbf{W})$ has a unique *reachable closure* $\mathbf{C}$, where $\mathbf{S} \subseteq \mathbf{C} \subseteq \mathbf{V}$, that is the minimal set (w.r.t. inclusion) that is a subset of $\mathbf{S}$ that is reachable. For example, in the paragraph above we described a graph where C was not fixable before both A and B were fixed. Let's say that D is fixable at any time. We have that the set $\{ A, C \}$ is not reachable. We need to add B to make the set reachable. Since we made a minimal addition to the set, and it is sufficient, we know that $\{ A, B, C \}$ is the reachable closure of $\{ A, C \}$.

Now, an ADMG $\mathcal{G}(\mathbf{V})$ is *arid* if and only if the reachable closure of each node $V \in \mathbf{V}$ node contains only itself, or more precisely $\langle \{ V \} \rangle_{\mathcal{G}} = \{ V \}$. This is important, because the SEM for an ADMG is everywhere identifiable if and only if it is arid (Shpitser et al., 2018). The term *arid* is used, because an induced subgraph $\mathcal{G}_{\langle \{ V \} \rangle}$ is also called a *V-rooted C-tree* or an *arborescence converging on V*, and trees don't grow well in arid environments. It is these C-trees or arborescences that cause problems in identifiability.

Now we will define the property of *maximality*, for which we need the notion of *densely connected* nodes.

Definition 2.5. *A pair of nodes $A \neq B$ is* densely connected *if one of the following holds:*

- $A \in pa_{\mathcal{G}}(\langle \{ B \} \rangle_{\mathcal{G}})$*, or*
- $B \in pa_{\mathcal{G}}(\langle \{ A \} \rangle_{\mathcal{G}})$*, or*
- $\langle \{ A, B \} \rangle_{\mathcal{G}}$ *is a bidirected-connected set.*

A CADMG is maximal *if every pair of densely connected vertices is adjacent.*

Then we finally come to the definition of the maximal arid projection:

[13] With the graphical operation of fixing also comes a probabilistic operation on the set of distribution the graphical model defines, but we will not go into those definitions.

Definition 2.6. *The* maximal arid projection *of a CADMG* $\mathcal{G}(\mathbf{V}, \mathbf{W})$*, which we will denote* $\mathcal{G}^\dagger$*, is the graph that contains the same vertices* $\mathbf{V}$ *and* $\mathbf{W}$*, but only contains the following edges. For each* $A, B \in \mathbf{V}$*,*

- *there is an edge* $A \to B$ *if* $A \in pa_{\mathcal{G}}(\langle \{ B \} \rangle_{\mathcal{G}})$*,*
- *and there is an edge* $A \leftrightarrow B$ *if*
  - *neither*
    - $A \in pa_{\mathcal{G}}(\langle \{ B \} \rangle_{\mathcal{G}})$
    - *nor* $B \in pa_{\mathcal{G}}(\langle \{ A \} \rangle_{\mathcal{G}})$
  - *and* $\langle \{ A, B \} \rangle_{\mathcal{G}}$ *is a bidirected-connected set.*

It is important for our purposes to note the following:

Lemma 2.1. *Every arid graph is bow-free.*

Proof. We prove the equivalent contrapositive statement: if a graph is not bow-free, then it is not arid. If a graph is not bow-free, it contains a bow, i.e. a construction for some $A, B \in \mathbf{V}$ such that $A \neq B$, and $A \to B$ and $A \leftrightarrow B$. Then, $\mathrm{dis}(A) \subseteq \{ A, B \}$, and $\mathrm{de}(A) \subseteq \{ A, B \}$, so $\mathrm{dis}(A) \cap \mathrm{de}(A) \subseteq \{ A, B \} \neq \{ A \}$. That means that A cannot be fixed before B is. Therefore, the reachable closure of $\{ B \}$ must include A as well, which means that the graph is not arid. □

It is also important that the converse is not true:

Lemma 2.2. *Not every bow-free CADMG is arid.*

Proof. We prove this by providing a counterexample that is bow-free, but not arid. We see that the ADMG in Figure 6a has no bows. However, since the graph is fully bidirected-connected, the only district is the district containing all nodes, and all nodes are ancestors of C. That means that for each node $X \in \mathbf{V} \setminus \{ C \}$, C is both in its descendants and in its district, which in turn means that none of them can be fixed before C. The reachable closure of C, therefore, contains all the nodes in the graph, instead of just C: $\langle \{ C \} \rangle = \mathbf{V} \{ A, B, C, D \} \neq \{ C \}$. That means that the graph is not arid. □

This means that restricting the search space to MArGs is actually meaningful, compared to only restricting it to BAPs.

3 Algorithms

3.1 greedyBAP

The implementation of the algorithms developed here are based mainly on the greedyBAPs algorithm by Nowzohour et al. (2017),[14] which we will now describe. It should be noted that the idea of restricting the search space to MArGs

[14] The code for greedyBAP, and the experiments run by Nowzohour et al. (2017), is available at https://github.com/cnowzohour/greedyBAPs.

does not depend on the specific implementation of greedyBAP, nor on the fact that greedyBAPs searches for linear-Gaussian models. In theory, it should be applicable to other types of variables and other types of models as well.

What makes greedyBAPs unique is its neighbourhood relation, which ensures that we search only in the space of BAPs, which are bow-free ADMGs. For compatibility with our own research we will formulate this a function that maps a graph to its neighbours. For this case, $\mathcal{G}' \in \text{neighbours}_{\text{BAP}}(\mathcal{G})$ if and only if $\mathcal{G}'$ is a BAP and either

- $\mathcal{G}'$ contains one edge that $\mathcal{G}$ does not, but is otherwise identical (edge addition), or
- $\mathcal{G}'$ is missing one of the edges $\mathcal{G}$ has, but is otherwise identical (edge deletion), or
- $\mathcal{G}$ and $\mathcal{G}'$ differ only in the direction or type of one edge (edge change, the options for edges are $\rightarrow$, $\leftarrow$, and $\leftrightarrow$).

We may also choose to only allow a subset of these conditions, to get forward search or backward search.

The score function employed by greedyBAPs is the following:

$$s(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) =_{\text{def}} \frac{1}{n}\left(\mathcal{L}(\mathcal{G}, \hat{\theta}_{\mathcal{G}}) - (|\{nodes\}| + |\{edges\}|)\log n\right)$$

It differs from the BIC score only in that the penalty for nodes and edges is twice as large, which the authors found to work well in testing. Of course, in the actual implementation, the score is decomposed into score parts per district. See the paper for details on that decomposition. Partial scores are cached, so that scoring slightly different graphs takes very little time.

To prevent getting stuck in local maxima, greedyBAPs uses multiple random restarts from uniformly sampled random BAPs, using a self-defined sampling procedure based on Markov chain Monte Carlo steps.

3.2 Arid algorithms

We compare greedyBAPs to three different algorithms, which we will describe here.[15]

Arid limiting search The first algorithm limits the search space to arid graphs. For a move to be allowed, the new graph must not only be acyclic and bow-free, but also arid (which of course also implies bow-freeness). This can be seen as a "correct" version of greedyBAP, because of the following.

As Shpitser et al. (2014) point out, the rationale for using BIC, given with its definition by Schwarz (1978), is that it is an approximation of the model posterior ($P(\text{graph} \mid \text{data})$). The proofs for this assume that the parameter map

[15] The code for all of the algorithms, and also for the experiments and the generated datasets, can be found on https://github.com/Sebastiaan-Jans/greedyBAPs.

(mapping parameters to distributions) of the model in question is smooth. This does not hold for models with unmeasured variables in general. As Shpitser et al. (2018) show, only SEMs associated with arid graphs are known to be guaranteed to represent a so-called *curved exponential family*, and therefore to have a smooth parameter map. Using BIC on BAPs is therefore not entirely theoretically justified. The same, and what follows, also holds for the score used by Nowzohour et al. (2017), which is only a slight variant of BIC.

Arid graphs are, as mentioned before, everywhere identifiable, while BAP models are only almost everywhere identifiable: the set of unidentifiable parameters has measure zero in the space of possible parameters. The probability that, if we were to randomly generate a parameter for an edge, we generate a parameter that would be unidentifiable for a parameter estimation procedure, is zero. In the same way, the probability that the "true" parameter of an edge is such that not all parameters of the model can be identified is zero. Still, such parameters may exist. This works like a line on a two-dimensional plane. The line is there, and you can name points that lie on the line if you know where the line is, but the probability of hitting it by randomly choosing a point on the plane that the line goes through it is zero.

However, though these unidentifiable parameters should not generally cause problems for parameter identification, they can cause the parameter map to become non-smooth. As Drton (2009) points out, this may cause the likelihood function to behave strangely near those unidentifiable parameters, which can be a problem for BIC scoring.

Aside from being in a sense more correct, the restriction to arid graphs is also without loss of generality (within the class of nested Markov models), since Shpitser et al. (2018) show that SEMs associated with arid graphs represent all Gaussian nested Markov models. A restriction of the search space can be useful simply because we may need to score fewer graphs, and we may prevent the search from reaching an unfavorable local maximum.

We now prove some relevant facts for the implementation of this algorithm. For example, to check aridity of a resulting graph after a local move, we need to check that it is still the case that the reachable closure of each node contains only itself. It turns out that we do not need to check the reachable closures of each node in the node set ($\mathbf{V}$). This may be quite helpful, because finding the reachable closure can be a relatively expensive operation, being quadratic in the number of nodes in the worst case,[16] and it may not be uncommon to get situations close to that worst case. That would make checking the reachable closure of all nodes in a set $\mathbf{S}$ on the order of $|\mathbf{S}|^3$.

Specifically, when we add an edge $A \leftrightarrow B$, we only need to re-check the

[16]To compute the reachable closure of a set $\mathbf{S}$, we try to fix all nodes in $\mathbf{V}$ outside $\mathbf{S}$, until we find that we cannot fix any more. In the best case where no nodes are fixable, we only need one pass. If some nodes are fixable, we may still need only two passes: one to fix the nodes, another to check that no more can be fixed. In the worst case, all of them might be fixable, but every we time that we check all of them, all of the ones we check – except the last one – cannot be fixed before the last one we check. This way, we need to make an order $|\mathbf{V} \setminus \mathbf{S}|$ pass for each node, which makes it quadratic.

reachable closures of nodes in the district of A and B in the new graph (which they are both in), and when we add an edge $A \to B$, we need only re-check the reachable closures of nodes in the descendants of B (including B, of course).

Lemma 3.1. *Let $\mathcal{G}$ be an arbitrary arid graph with node set $\mathbf{V}$ which contains nodes A, B, and possibly others, where there is no bidirected edge $A \leftrightarrow B$. Let $\mathcal{G}'$ be the graph that results from adding the bidirected edge $A \leftrightarrow B$ to $\mathcal{G}$, which shares the node set $\mathbf{V}$. Then, for all $V \notin dis_{\mathcal{G}'}(A) (= dis_{\mathcal{G}'}(B) = dis_{\mathcal{G}'}(\{ A, B \}))$, we have $\langle \{ V \} \rangle_{\mathcal{G}'} = \{ V \}$.*

Proof. For a node $V \in \mathbf{V}$ to have $\langle \{ V \} \rangle_{\mathcal{G}'} \neq \{ V \}$, there must be one or more nodes $T \in \langle \{ V \} \rangle_{\mathcal{G}'} \setminus \{ V \}$. Then, we must have $T \in \mathrm{dis}_{\mathcal{G}'}(V)$ and $T \in \mathrm{an}_{\mathcal{G}'}(V)$, because otherwise T would be fixable in $\mathcal{G}'_{\langle \{ v \} \rangle_{\mathcal{G}'}}$ and it would not be in $\langle \{ v \} \rangle_{\mathcal{G}'}$. The only changes in genealogical relations from the addition of $A \leftrightarrow B$ are that $\mathrm{dis}_{\mathcal{G}'}(A) = \mathrm{dis}_{\mathcal{G}'}(B) = \mathrm{dis}_{\mathcal{G}'}(\{ A, B \}) = \mathrm{dis}_{\mathcal{G}}(A) \cup \mathrm{dis}_{\mathcal{G}}(B)$. The addition of a bidirected edge affects no ancestor relations. Therefore, if $V \notin \mathrm{dis}_{\mathcal{G}'}(A)$, its ancestor sets and districts are the same in $\mathcal{G}'$ as in $\mathcal{G}$. Since we know that there was no such node T in $\mathcal{G}$ – because $\mathcal{G}$ is arid – there cannot now be such a node T in $\mathcal{G}'$, and we have $\langle \{ V \} \rangle_{\mathcal{G}'} = \{ V \}$. □

Lemma 3.2. *Let $\mathcal{G}$ be an arbitrary arid graph with node set $\mathbf{V}$ which contains nodes A, B, and possibly others, where there is no directed edge $A \to B$. Let $\mathcal{G}'$ be the graph that results from adding the bidirected edge $A \to B$ to $\mathcal{G}$, which shares the node set $\mathbf{V}$. Then, for all $V \notin de_{\mathcal{G}'}(B) = de_{\mathcal{G}}(B)$, we have $\langle \{ V \} \rangle_{\mathcal{G}'} = \{ V \}$.*

Proof. The only genealogical relations affected here are that $\mathrm{an}_{\mathcal{G}'}(B) = \mathrm{an}_{\mathcal{G}}(B) \cup \mathrm{an}_{\mathcal{G}}(A)$ and $\mathrm{de}_{\mathcal{G}'}(A) = \mathrm{de}_{\mathcal{G}}(A) \cup \mathrm{de}_{\mathcal{G}}(B)$. The only way this can affect fixabilities is if there is a node $X \in \mathrm{an}_{\mathcal{G}}(A)$ and a node $Y \in \mathrm{de}_{\mathcal{G}}(B)$, such that $X \in \mathrm{dis}_{\mathcal{G}}(Y)$. Then, Y becomes a descendant of X in $\mathcal{G}'$, so X cannot be fixed before Y in $\mathcal{G}'$, which means that the reachable closure of Y in $\mathcal{G}'$ will include X. This means that the only nodes for which the reachable closure can be different in $\mathcal{G}'$ than in $\mathcal{G}$ are those in $\mathrm{de}(B)$ (including B itself). For nodes $V \notin \mathrm{de}_{\mathcal{G}'}(B)$, we already have that $\langle \{ V \} \rangle_{\mathcal{G}'} = \{ V \}$ because that same property holds in $\mathcal{G}$ – since it is arid – and their reachable closures are unchanged. □

Maximal arid limiting search The second algorithm limits the search to maximal arid graphs. For a move to be allowed, the new graph must be acyclic, (bow-free,) and maximal arid. Unfortunately, we have found no way to make this more efficient than simply checking whether a graph's maximal arid projection equals the graph itself. Both are operations on the order of $|\mathbf{V}|^4$, since to check maximality, we need to check the reachable closure of all pairs of nodes. The only slight implementation difference is that we do not need to make a copy of the graph to check if it is maximal arid, though we did not implement that optimisation.

Maximal arid projection search The last algorithm, and perhaps the most interesting one, defines moves slightly differently. We will refer to this algorithm as simply the *projection algorithm*. It makes sense to define the neighbourhood function for this algorithm, which we will simply call the *projection algorithm*, in a more algorithmic style. Essentially, we make all edge additions/changes/edges according to $\text{neighbours}_{\text{BAP}}$, and then take the maximal arid projection of all those graphs. Then we filter out those graphs whose maximal projection goes back to being $\mathcal{G}$ again. More formally, the neighbours of $\mathcal{G}$ are

$$\text{neighbours}_{\text{projection}}(\mathcal{G}) =_{\text{def}} \left\{ \mathcal{G}'^{\dagger} \mid \mathcal{G}' \in \text{neighbours}_{\text{BAP}}(\mathcal{G}) \wedge \mathcal{G}'^{\dagger} \neq \mathcal{G} \right\},$$

where † is the maximal arid projection operator as before.

The projection can make moves that affect more than one edge at once. This can be illustrated by using the graph in Figure 6a. Let's call it $\mathcal{G}$, and let's call the graph that results from removing the edge $D \to C$ from it G^{-}, shown in Figure 7. This graph is maximal arid. If this graph is the current state in the projection algorithm, a possible move would be to add the edge $D \to C$. That results in the graph $\mathcal{G}$, which as we know is not arid. The projection algorithm then takes the maximal arid projection, resulting in the graph in Figure 6b. As we see, the addition of the edge $D \to C$ also causes the edge $A \leftrightarrow C$ to become directed: $A \to C$, so it causes a change in an edge that involves none of its own end-points. Note that this G^{-} is just a concrete example, the same goes for any other graph G^{-} resulting from an edge removal from $\mathcal{G}$.

The ability to make changes to multiple edges at once has some advantages and some disadvantages. The limiting algorithms would simply disallow the move adding $D \to C$ to G^{-} because it results in a non-arid graph, while the projection algorithm does allow it, and "fixes" the non-aridity by taking the maximal arid projection. Permitting more moves may allow the algorithm to explore a larger part of the search space, which might increase the chance of escaping a local maximum. A disadvantage would be performance. When making local changes, it is quite straightforward to work out which districts – and corresponding parent sets – have changed, and should have their (partial) score recomputed, so that only that partial score needs to be recomputed. It is not obvious what the effects of a local move followed by a maximal arid projection on the districts may be on the districts of a graph, so the whole graph will have to be re-scored after each non-local move in the projection algorithm, after the set of districts are re-calculated. This is not catastrophic, since partial scores can be cached, and not too many parts should have to actually be re-scored. It does mean that some look-ups need to be performed to retrieve the partial scores.

(a) A bow-free, non-arid ADMG. (b) Its maximal arid projection.

Figure 6: Maximal arid projection.

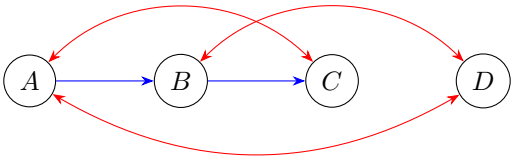


Figure 7: The graph $\mathcal{G}^-$, which results from removing the edge $D \to C$ from the graph in Figure 6a.

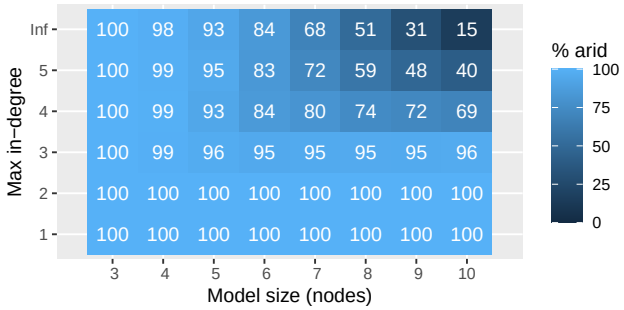
4 Experiments and results

4.1 Number of (maximal) arid BAPs

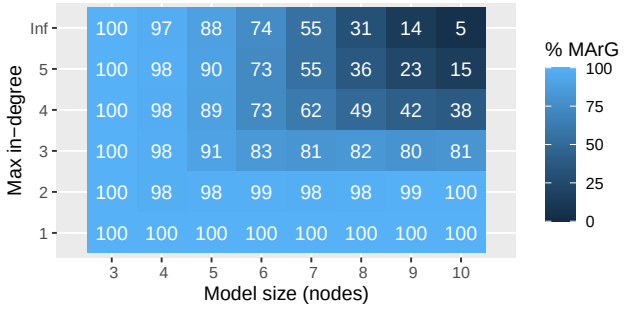
To gain insight into the space through which the algorithms search, we tested how much the search space restrictions actually restrict the search space. For different values of the maximum in-degree α and model size d (node count), we used the algorithm for generating uniformly distributed random BAPs provided by Nowzohour et al. (2017) to generate 1000 random BAPs. We then checked which of these BAPs were arid, and which were maximal arid. The results are shown in Figure 8a. As can be seen, for low values of α and d, very few to no BAPs are not (maximal) arid. For larger values, however, few BAPs *are* (maximal) arid. Of course, fewer BAPs are maximal arid than arid. It should be noted that a uniform sample from the space of all d-node BAPs with maximum in-degree α will include many more models with many edges than models with few edges, because there are simply more models with many edges: there are more ways to arrange many edges than there are ways to arrange few edges. Indeed, from visually inspecting a sample of generated BAPs with unrestricted in-degree ($\alpha = \infty$), it became clear that most of them have edges between most node pairs. This means that a uniform sample over all possible BAPs is not a perfect representation of the type of graphs that a greedy search algorithm starting from the empty graph is actually likely to encounter.

4.2 Greedy search experiments

To test the algorithm, we ran it on generated data. The testing procedure was as follows.

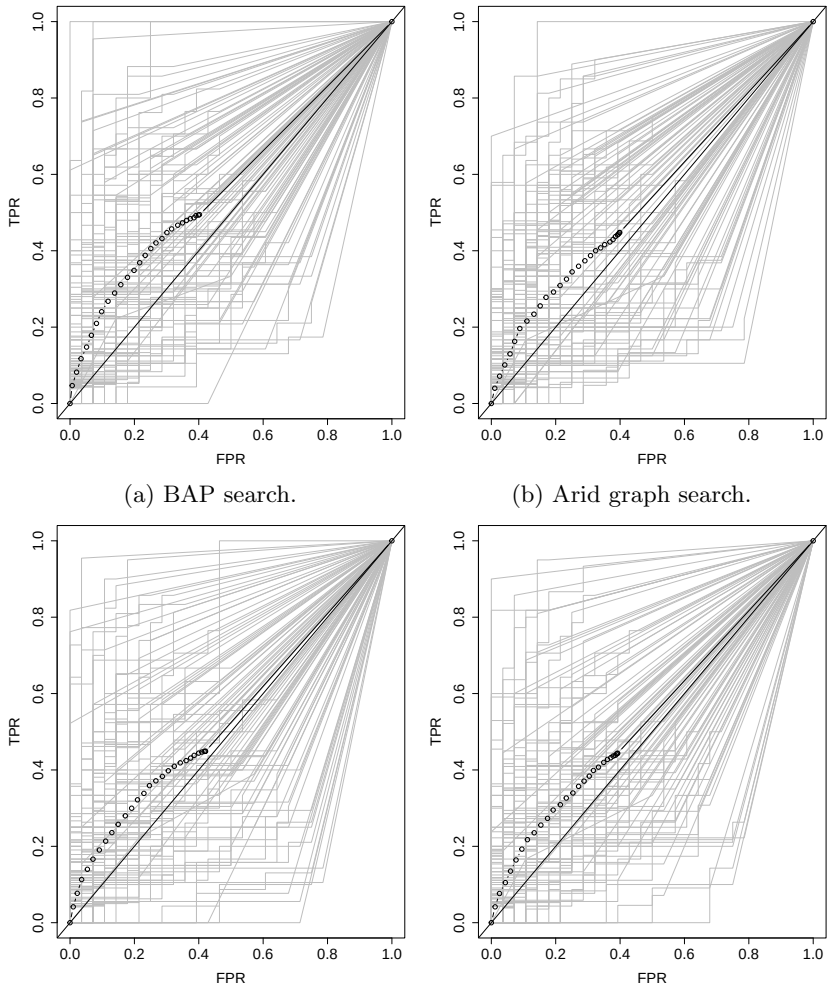


(a) Percentage of arid BAPs.



(b) Percentage of maximal arid BAPs.

Figure 8: Percentage of (maximal) arid BAPs in a sample of 1000 per model size and max in-degree.



(a) BAP search.

(b) Arid graph search.

(c) Maximal arid limiting search.

(d) Maximal arid projection search.

Figure 9: ROC curves of the minimal absolute causal effect predictions from the experimental setting $d = 8, \alpha = 4, N = 100$. The grey curves are the $N = 100$ resulting ROC curves, and the black curve is the point-wise average of those ROC curves, with each point showing the average TPR against the average FPR for a specific threshold.

Experimental setup After some preliminary examination of the algorithms, we chose the following three experimental settings for larger samples. We generate random ground truth graphs $\mathcal{G}$ for the settings shown in Table 1, where we also show the means and standard deviation of edge count of the generated ground truth graphs, and the maximum possible edge count for a graph with d nodes. We used the uniform BAP sampling procedure developed by Nowzohour et al. (2017), and set the model size (number of nodes) to d. The maximum allowed in-degree for a generated graph was set to α, and we generated N graphs for each setting. We then took the maximal arid projection of these graphs. Note that it can be argued that this puts the greedyBAPs and arid limiting algorithms at a slight disadvantage, since the ground truth is known to be a MArG.

Experimental setting/graph type	Mean(SD) of edges	Possible edges
$d = 8, \alpha = 4, N = 100$	17.95(2.86)	28
$d = 8, \alpha = 3, N = 100$	18.13(3.08)	28
$d = 6, \alpha = 6, N = 200$	9.26(1.48)	15

Table 1: The three experimental settings/graph types.

We chose these settings based on the results from the previous section, which showed that a significant fraction of BAPs in these regions are not arid, or maximal arid. We balanced this against the fact that large graphs with many edges and large districts are difficult for greedy search and parameter fitting. We also took into account that models with very high in-degree (say, 5 or more) are generally not very informative causal models. We then randomly generated parameters $\theta_{\mathcal{G}} = (B, \Omega)$ for each BAP $\mathcal{G}$, and simulated $n = 10000$ data points from each $\theta_{\mathcal{G}}$. See Nowzohour et al. (2017) for some details on parameter generation and data simulation.

Then, for each of the graph sets corresponding to the experimental settings, we ran the four algorithms, to get estimate graphs $\hat{\mathcal{G}}$. Within each experimental setting, each algorithm was run on the same set of N graphs. Of course, between the settings, the graphs are different. Though greedyBAPs is intended to be run with random restarts from randomly generated BAPs – Nowzohour et al. (2017) run it with 100 restarts – we did not use restarts. This is because our aim is not to find an algorithm with optimal performance, but to see if the aridity modifications have an effect on a greedy search. A greedy search in its purest form is run just once. To make as few assumptions as possible, we start this search at the empty graph (with no edges).

Evaluating results Evaluating a causal discovery algorithm over ADMGs is notoriously difficult, because no simply calculable graphical characterisation of equivalence exists for most models associated with ADMGs.[17] More concretely,

[17] A notable exception is MAGs, for which a greedy equivalence search algorithm was recently developed by Claassen and Bucur (2022), but as mentioned MAGs cannot represent causal structure as flexibly as MArGs (or superclasses of MArGs).

in this linear-Gaussian application, when we run the algorithm on data generated from a ground truth graph $\mathcal{G}$, and we obtain an estimate $\hat{\mathcal{G}}$, we cannot simply say that the estimate is correct if it equals $\mathcal{G}$, and incorrect if it doesn't. This is because $\hat{\mathcal{G}}$ may well in fact be correct, in that it is in the distributional equivalence class $DEC(\mathcal{G})$ of $\mathcal{G}$. Since most types of equivalence over ADMGs cannot be computed simply, measures like Structural Hamming Distance (SHD) are often employed. The SHD is simply the number of node pairs ($(A, B) \in \mathbf{V} \times \mathbf{V}$) where $\mathcal{G}$ and $\hat{\mathcal{G}}$ disagree on what edge is there, or if there is an edge. This does not take equivalence into account at all, however, and is therefore still a rather poor estimate of the prediction quality.

This is where the greedy empirical equivalence class search algorithm by Nowzohour et al. (2017) becomes very useful. They not only use the empirical equivalence class in the evaluation, but also do not compare edges, but identifiable causal effects. They reasonably claim that this is often more relevant in practice than actual edges. They estimate the causal effects matrix $\hat{E}$ for each graph $\mathcal{G}' \in \widehat{DEC}(\mathcal{G})$ (the estimated distributional equivalence class), which contains the causal effect of node X_j on X_i in each element $\hat{E}_{ij}$. They then take absolute values of all these effects, and take the element-wise minimum over all causal effects matrices in the equivalence class to get $\hat{E}^{min}_{\mathcal{G}}$. Now if $\hat{E}_{ij} > 0$ there is an effect of X_j on X_i in each of the graphs in $\widehat{DEC}(\mathcal{G})$. The same is done for $\hat{\mathcal{G}}$, to get $\hat{E}^{min}_{\hat{\mathcal{G}}}$.

Now we simply have two matrices to compare, that represent their empirical distributional equivalence classes. This is now an instance of binary classification: if there is an effect (a non-zero entry) in $\hat{E}^{min}_{\mathcal{G}}$, we want to predict that there is an effect in $\hat{E}^{min}_{\hat{\mathcal{G}}}$, and if there isn't an effect, we don't want to predict it. Instead of taking a non-zero entry to imply an effect immediately, we can also set a threshold. We can plot a receiver operating characteristic (ROC) curve, that for each value of the threshold calculates the true positive rate [18] and the false positive rate [19] and plots them against each other. The area under that curve (AUCs) is often a good measure of the performance of a binary classifier in general, regardless of a particular threshold.

The average AUC and standard deviaton for each algorithm and experimental setting is shown in Table 2, as well as the average time taken by the algorithm per graph.[20] The ROC curves for the setting $d = 8, \alpha = 4, N = 100$

[18] Thre true positive rate is given by $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$, where TP is the number of true positives, FN the number of false negatives, and P the number of ground truth positives: effects in $\hat{E}^{min}_{\mathcal{G}}$. This is also known as recall or sensitivity.

[19] The false negative rate is given by $FPR = \frac{FP}{FP+TN} = \frac{FP}{N}$, where FP is the number of false positives, TN the number of true negatives, and N is usually the number of ground truth negatives. In this case, strictly speaking, we don't use the real FPR. Because the predictions aren't independent – we cannot have an effect from A to B as well as B to A – using the true number of non-effects as N in the denominator would result in false positive rates greater than 1. Instead, the number of possible effects ($\frac{d(d-1)}{2}$ with model size d) is used. The more widely known precision statistic ($\frac{TP}{TP+FN} = \frac{TP}{PP}$ with PP the predicted positives) can also be used instead of the false positive rate for ROC analysis, but this is less commonly done.

[20] Individual graph times were not recorded, so the standard deviations of time per graph

Graph type	Algorithm	Mean(SD) of AUC	Time (s/graph)
($d = 8$,	greedyBAPs	0.57(0.17)	7.4
$\alpha = 4$,	Arid limiting	0.55(0.17)	19.4
$N = 100$)	MArG limiting	0.54(0.19)	103.5
	MArG projection	0.54(0.18)	119.9
($d = 8$,	greedyBAPs	0.57(0.19)	7.8
$\alpha = 3$,	Arid limiting	0.55(0.18)	19.2
$N = 100$)	MArG limiting	0.52(0.18)	108.7
	MArG projection	0.55(0.18)	121.1
($d = 6$,	greedyBAPs	0.55(0.23)	4.3
$\alpha = 3$,	Arid limiting	0.55(0.22)	7.4
$N = 200$)	MArG limiting	0.55(0.24)	21.3
	MArG projection	0.55(0.23)	23.7

Table 2: AUC and time results of the various algorithms.

are shown in Figure 9. The thresholds were chosen in steps such that for the lowest threshold, all non-zero entries in $\hat{E}_{\mathcal{G}}^{min}$ are predicted as effects (top-right corner of the ROC curve), and for each next threshold value, one fewer effect is predicted, until for the last threshold, no effects are predicted (bottom-left corner of the ROC curve). Apart from the individual ROC curves per run, for each threshold value, the average TPR is plotted against the average FPR, in black, taken over all N runs. For interpretation: a diagonal line corresponds to chance performance and $AUC = 0.5$, to the left and above is better than chance and $AUC > 0.5$.

For each experimental setting, we performed a repeated measures ANOVA analysis, with the algorithm as the independent variable, and the AUC as the dependent variable. A repeated measures ANOVA is justified, because the different algorithms are run "within subjects": the graphs are the subjects, and the algorithms are run on the same set of graphs. We found no significant difference of AUC between the algorithms for the setting $d = 8, \alpha = 4, N = 100$ ($F(3, 297) = 1.68, p = .17$), nor for the setting $d = 8, \alpha = 3, N = 100$ ($F(3, 297) = 1.68, p = .17$), nor for $d = 6, \alpha = 3, N = 200$ ($F(3, 297), p = .22$).

Visual histogram inspections showed that the AUC was approximately normally distributed in all conditions except $d = 8, \alpha = 4, N = 100$, which showed a spike at $AUC \approx 1$ next to the otherwise normal distribution as shown in Figure 10. We speculate that this could be due to the greedy equivalence search algorithm. It may have incorrectly found the same equivalence class for the ground truth and the estimate many times. This could be because the relevant setting $d = 8, \alpha = 4$ is quite challenging: it is a very large search space.

could not be calculated.

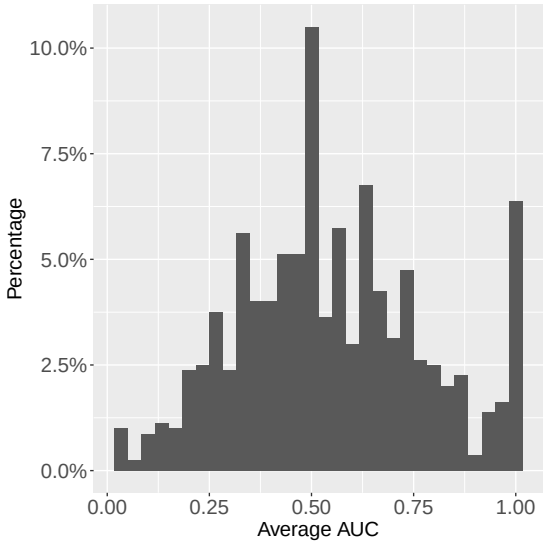


Figure 10: Distribution of average AUCs per graph for the condition $d = 8, \alpha = 4, N = 100$.

5 Discussion

Findings We found no improvements over greedyBAPs for any of the proposed algorithms. It is possible, of course, that the algorithms would show better performance over a different search space, perhaps of BAPs with even more nodes and edges, though with the minute differences in AUC we found it seems unlikely. The algorithm limiting the space to arid graphs may still be chosen, since it takes little longer than greedyBAPs, and is more theoretically sound. The algorithm limiting the space to maximal arid graphs, however, takes much longer. The maximal arid projection algorithm is more flexible in its moves, but still shows no performance benefit, and is as slow as the maximal arid limiting algorithm. Neither of the maximal arid algorithms therefore seems useful.

Limitations We previously intended to test the maximal arid projection version of greedy search on categorical variables with nested Markov models, using the fitting provided by the `ananke-causal` Python package.[21] However, parameter fitting turned out to be prohibitively slow, which makes sense because this package is intended for analysis of single models, when it does not matter that fitting takes a few seconds. In a greedy search over many variables, however, this does matter. Re-implementing nested Markov model fitting in a way that is more suited to model search would have been too time-consuming, with uncer-

[21] Available at https://gitlab.com/causal/ananke/.

tain results. The maximal arid projection does fit quite naturally with discrete nested Markov models, since it preserves nested Markov equivalence, which is complete with respect to equality constraints for discrete models, while it does not preserve distributional equivalence for linear-Gaussian models.

Theoretically, it makes sense to allow arbitrary (non-bow-free) ADMGs to result from the first step of the neighbours function, after which the maximal arid projection is taken, because not doing so may restrict the search space unnecessarily. Unfortunately, the implementation of greedyBAPs is not very modular (to make many precise optimisations possible), so it was not possible to replace that representation within the time-frame of this study.

6 Conclusion

In this study, we investigated whether restricting the search space of a greedy search algorithm over the space of BAPs (greedyBAP) to arid graphs or maximal arid graphs, or taking the maximal arid projection of each move, would improve performance. We argued that the theoretical backing for scoring only arid graphs with BIC, or a BIC-like score, is stronger. For this reason, one might argue for the use of the arid limiting algorithm instead of greedyBAPS. However, since none of the algorithms showed an increase in performance on the simulation studies, there seems to be no reason to choose the maximal arid limiting algorithm or the maximal arid projection algorithm over greedyBAPs.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*(6), 716–723. doi:10.1109/TAC.1974.1100705

Chickering, D. M. (2003). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, *3*, 507–554. doi:10.1162/153244303321897717

Chickering, D. M. (2020). Statistically efficient greedy equivalence search. In J. Peters & D. Sontag (Eds.), *Proceedings of the 36th conference on uncertainty in artificial intelligence (uai)* (Vol. 124, pp. 241–249). PMLR. Retrieved from https://proceedings.mlr.press/v124/chickering20a.html

Chickering, D. M., & Meek, C. (2015). Selective greedy equivalence search: Finding optimal bayesian networks using a polynomial number of score evaluations. *CoRR*, *abs/1506.02113*. arXiv: 1506.02113. Retrieved from http://arxiv.org/abs/1506.02113

Claassen, T., & Bucur, I. G. (2022). Greedy equivalence search in the presence of latent confounders. In J. Cussens & K. Zhang (Eds.), *Proceedings of the 38th conference on uncertainty in artificial intelligence (uai)* (Vol. 180, pp. 443–452). PMLR. Retrieved from https://proceedings.mlr.press/v180/claassen22a.html

Daly, R., & Shen, Q. (2009). Learning bayesian network equivalence classes with ant colony optimization. *Journal of Artificial Intelligence Research*, *35*, 391–447. doi:10.1613/jair.2681

Drton, M. (2009). Likelihood ratio tests and singularities. *The Annals of Statistics*, *37*(2), 979–1012. doi:10.1214/07-AOS571

Evans, R. J. (2016). Graphs for margins of bayesian networks. *Scandinavian Journal of Statistics*, *43*(3), 625–648. doi:10.1111/sjos.12194

Evans, R. J. (2018). Margins of discrete bayesian networks. *The Annals of Statistics*, *46*(6A), 2623–2656. doi:10.1214/17-AOS1631

Frot, B., Nandy, P., & Maathuis, M. H. (2019). Robust Causal Structure Learning with Some Hidden Variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *81*(3), 459–487. doi:10.1111/rssb.12315

Glymour, C., Zhang, K., & Spirtes, P. (2019). Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, *10*. doi:10.3389/fgene.2019.00524

Glymour, M., Pearl, J., & Jewell, N. P. (2016). *Causal inference in statistics: A primer*. John Wiley & Sons.

Huang, B., Low, C. J. H., Xie, F., Glymour, C., & Zhang, K. (2022). Latent hierarchical causal structure discovery with rank constraints. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (Vol. 35, pp. 5549–5561). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2022/file/24d2dd6dc9b79116f8ebc852ddb9dc94-Paper-Conference.pdf

Meek, C. (1997). *Graphical models: Selecting causal and statistical models* (Doctoral dissertation, Carnegie Mellon University). doi:10.1184/R1/22696393.v1

Nowzohour, C., Maathuis, M. H., Evans, R. J., & Bühlmann, P. (2017). Distributional equivalence and structure learning for bow-free acyclic path diagrams. *Electronic Journal of Statistics*, *11*(2), 5342–5374. doi:10.1214/17-EJS1372

Pearl, J. (2009). *Causality* (2nd ed.). Cambridge university press.

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

Richardson, T. S., Evans, R. J., Robins, J. M., & Shpitser, I. (2023). Nested Markov properties for acyclic directed mixed graphs. *The Annals of Statistics*, *51*(1), 334–361. doi:10.1214/22-AOS2253

Robins, J. (1986). A new approach to causal inference in mortality studies with a sustained exposure period—application to control of the healthy worker survivor effect. *Mathematical Modelling*, *7*(9), 1393–1512. doi:10.1016/0270-0255(86)90088-6

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*(2), 461–464. Retrieved from http://www.jstor.org/stable/2958889

Shpitser, I., Evans, R. J., & Richardson, T. S. (2018). Acyclic linear sems obey the nested markov property. In *Proceedings of the 34th conference on uncertainty in artificial intelligence (uai)*.

Shpitser, I., Evans, R. J., Richardson, T. S., & Robins, J. M. (2014). Introduction to nested markov models. *Behaviormetrika*, *41*, 3–39. doi:10.2333/bhmk.41.3

Shpitser, I., Richardson, T. S., Robins, J. M., & Evans, R. (2012). Parameter and structure learning in nested markov models. In *Proceedings of the causal structure learning workshop of the 28th conference on uncertainty in artificial intelligence (uai)*.

Spirtes, P., Glymour, C., & Scheines, R. (2001). *Causation, prediction, and search* (2nd ed.). MIT Press.

Tian, J., & Pearl, J. (2002). A general identification condition for causal effects. In *Eighteenth national conference on artificial intelligence* (pp. 567–573). Edmonton, Alberta, Canada: American Association for Artificial Intelligence.

van Ommen, T., & Mooij, J. M. (2017). Algebraic equivalence of linear structural equation models. In *Proceedings of the 33rd conference on uncertainty in artificial intelligence (uai)*, Sydney.

Verma, T., & Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the sixth conference on uncertainty in artificial intelligence (uai)* (pp. 220–227).

Vowels, M. J., Camgoz, N. C., & Bowden, R. (2022). D'ya like dags? a survey on structure learning and causal discovery. *ACM Computing Surveys*, *55*(4). doi:10.1145/3527154

A Glossary of abbreviations

DAG directed acyclic graph

BN Bayesian network

MEC Markov equivalence class

ADMG acyclic directed mixed graph

CADMG conditional acyclic directed mixed graph

CPDAG complete partial directed acyclic graph

SEM structural equation model

SCM structural causal model

m**DAG** marginalised directed acyclic graph

MArG maximal arid graph

MAG maximal ancestral graph

BAP bow-free acyclic path diagram

FCI fast causal inference

GES greedy equivalence search

AIC Akaike information criterion

BIC Bayesian information criterion

ROC receiver operating characteristic

AUC area under the (ROC) curve