Karlsruhe Institute of Technology

(Active) Learning of Causation

Proseminar of

Brandon Schühlein

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Advisor: M.Sc. Bela Böhnke

21. October 2021 – 03. March 2022

Abstract

Learning cause-effect relationships is fundamentally ingrained in human nature and virtually all fields of science. Because when we know them, we can adjust our actions to achieve the desired effect, and we gain a deep understanding of the underlying processes. One usually learns these cause-effect relationships by conducting experiments; however, that is difficult or even impossible in many cases. It is then necessary to discover cause-effect relationships by analyzing the statistical properties of purely observational data. We aim to give an introduction to and a overview of the algorithms for learning cause-effect relationships from observational as well as experimental data. In addition, we further look at brand-new improvements upon the basic algorithms and compare them. We found that, focusing on only one of the two cases is insufficient, and combining them leads to good results.

Contents

1 Introduction

1.1 Motivation

It is hard to deny that our understanding of the world is linked to the cause-effect relationships we discover in it. We identify and experiment with these relationships from an early age: crying long enough gets the parents' attention, putting our hand on the stove leads to pain, etc. Knowing these cause-effect relationships allows us to adjust our actions (interventions) in a way that gives us the desired effect. Therefore, Causation is fundamentally ingrained in human nature. However, learning causal relationships is at the heart of many fields of science as well. Furthermore, usually one goes about learning them in one of two ways, either by intervening in the system by some experiment or by observing the system itself - both are valid approaches. This paper is about how computer science can aid the process of causal discovery by reviewing efficient and robust algorithms for both observational and interventional cases.

Even though this field of research is not brand-new anymore (necessary groundwork was published in the late 90s and early 2000s), it has gained more traction over the last couple of years. Most notably, the explosion in the number of algorithms, especially for application in biology [7]. Because of this wide range of algorithms, we will mainly focus on the fundamental underlying types and improvements. Before we look at the algorithms, we need to introduce some basic concepts.

1.2 What is Causation?

"Causation is influence by which one event (a cause) contributes to the production of another event (an effect) where the cause is partly responsible for the effect". Therefore, we have some relationship between two events: the cause and the effect.

We look at an example to understand these cause-effect relationships. Suppose we measure atmospheric pressure X with a barometer, which indicates a value Y. It is evident that X and Y are closely related, if not identical (if the barometer is perfect). For our example, we assume that the barometer has some bias b:

$$Y = X + b \tag{1.1}$$

With that we can easily infer the actual pressure:

$$X = Y - b \tag{1.2}$$

These equations are algebraically equivalent, but are there any reasons why we should prefer one over the other? We realize that 1.1 describes a cause-effect relationship, whereas 1.2 does not. Therefore, 1.1 carries more information than 1.2, and that is why we should prefer it. The property of such a cause-effect relationship is, that it is invariant under manipulation [27]: If

we were able to set the value of the atmospheric pressure X - a so-called intervention - we would still find that 1.1 accurately describes how the barometer value Y would respond to the change. In contrast, 1.2, where we intuitively expect that a change in our barometer value Y will not predict the change in atmospheric pressure X.

In its simplest form, a deterministic cause–effect relationship from a cause X to an effect Y can be written as follows:

$$Y := f(X) \tag{1.3}$$

where the function f can be seen as "Nature's thousands of little ears that constantly listen to X" [14] and determine the value of Y - which is irreversible.

This gives us the intuition needed to look at how we model these cause-effect relationships.

1.3 Causal Models

There are multiple ways of modeling cause-effect relationships in the literature, depending on the algorithm and assumptions. For simplicity reasons, we focus on the two most essential models and add nuances as we progress through the different algorithms.

1.3.1 Causal Bayesian Networks

As the name suggests, Causal Bayesian Networks inherit most of their properties from Bayesian Networks. However, in contrast to the well-known Bayesian Networks, the Causal Bayesian Networks models causal relationships.

Definition 1 (Causal Bayesian Network) *Let M be the set of variables in our data set, and $G = (V, E)$ a DAG, where V is the set of vertices/nodes and E is the set of directed edges. We define $V := M$ and $E \subseteq V \times V$. In addition, E needs to fulfill the following two assumptions:*

- *Local Markov Assumption: Given its parents in the DAG, a node X is independent of all its non-descendants.*
- *Causal Edges Assumption: In a directed graph, every parent is a direct cause of all children.*

1.3.2 Structural Causal Models

As already described in 1.2, the mathematical equal sign is symmetric, so saying A = B is the same as saying B = A. But since we want to express that A is a cause of B, not the other way around, we need an asymmetric expression. For this purpose, we introduce the structural equation:

$$B := f(A), \tag{1.4}$$

where f is a function that maps A to B. We deliberately choose :=, because it conveys asymmetry. Since in reality, we will most likely miss some relevant variables to write B as a deterministic function of A, we introduce some noise into the equation:

$$B := f(A, U), \tag{1.5}$$

where U is some unobserved random variable (noise).

Definition 2 (Structural causal model) *A set of equations in the form of:*

$$X_i := f_i(pa_i, U_i), i = 1, ..., n \tag{1.6}$$

where each variable has a distinct equation, in which it appears on the left-hand side (called the dependent variable) then the model is called a structural causal model or a causal model for short [15].

Note: So far, we do not assume any particular form of f, although that will become relevant in Section 2.4.

2 Causal Discovery from Observational Data

This chapter explains how it is possible to learn the structure of a causal graph from observational data. We introduce needed notation and further required knowledge, then show the standard algorithms to perform this task. Thereby we present two different approaches: independence-based and semi-parametric. We end this chapter by discussing improvements upon the standard algorithms and comparing these.

2.1 Introduction

How do we learn about cause-effect relationships using observational data without expert knowledge? We remember the old adage that says, "correlation is not causation." There is no such thing as a statistical test of causation. So how can we say anything about the causal structure of a given system, given observational data?

We need to learn about statistical independence and conditional independence. Although there is no test of causation, there are tests of (conditional) independence. These turn out to be the closest general criterion we can use to learn about causation in an observational study [17].

2.2 From (conditional) independencies to graphs

Knowing that (conditional) independencies are a "proxy" for causation, we now need to use them to build a causal graph. Before we can do that, we need to create a connection between the "data realm", and the "graph realm". We make the following assumptions and introduce new concepts, like the d-separation and the Markov equivalence class.

2.2.1 d-separation

To understand the upcoming assumptions, we first need to understand d-separation, which is a measure of the independence of nodes in a graph. Before we can define d-separation, we need to define the concept of a "blocked path". We illustrate this concept in Fig. 2.1 and Fig. 2.2. Here we see, the behaviour of unblocked paths in graphs like described in the first step and second step of the definition below. Thereby, also observing what changes, when we intervene on different nodes.

Definition 3 (Unblocked Path) *A path between nodes X and Y is blocked by a (potentially empty) conditioning set Z if either of the following is true:*

1. *Along the path, there is a chain $\cdots \to W \to \cdots$ or a fork $\cdots \leftarrow W \to \cdots$, where W is conditioned on ($W \in Z$).*

2. *There is a collider W on the path that is not conditioned on ($W \notin Z$) and none of its descendants are conditioned on ($de(W) \nsubseteq Z$) [13].*

Then, an unblocked path is just the complement; an unblocked path is a path that is not blocked. The graphical intuition is that association flows along unblocked paths, and association does not flow along blocked paths. Now, we are ready to introduce a fundamental concept: d-separation.

Definition 4 (d-separation) *Two (sets of) nodes X and Y are d-separated by a set of nodes Z if all of the paths between (any node in) X and (any node in) Y are blocked by Z [16].*

We will use the notation $X \perp\!\!\!\perp_G Y \mid Z$ to denote that X and Y are d-separated in the graph G when conditioning on Z. Similarly, we will use $X \perp\!\!\!\perp_P Y \mid Z$ to denote that X and Y are independent in the distribution P when conditioning on Z.

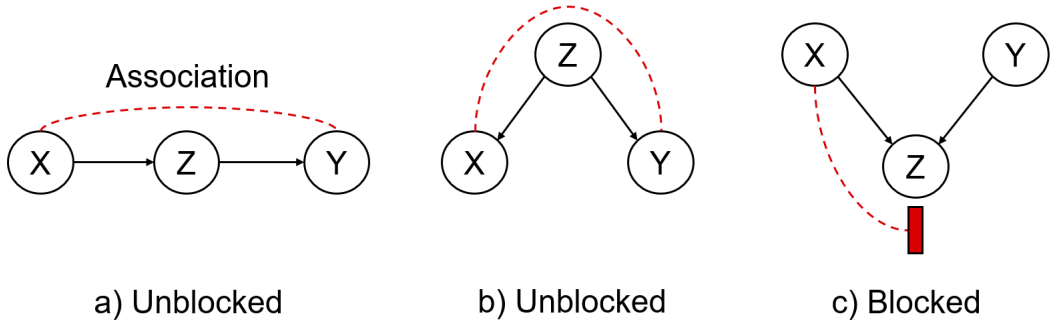

Figure 2.1: Identifying blocked paths by looking at association flow

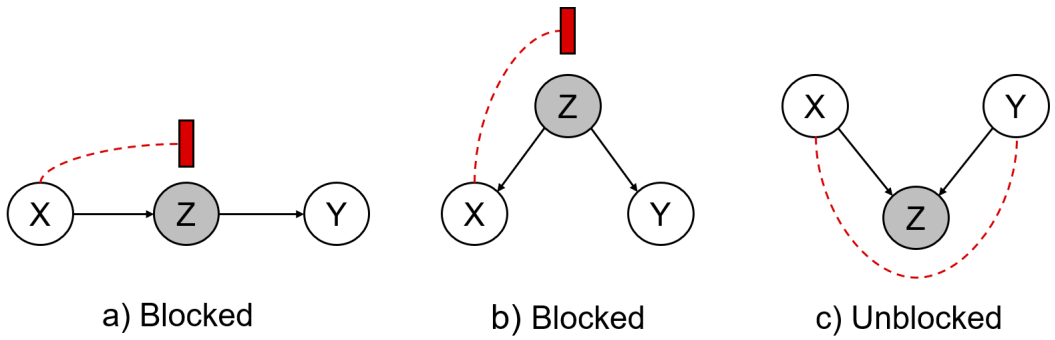

Figure 2.2: Blocked paths when conditioning on Z

2.2.2 (Global) Markov Assumption

The Global Markov Assumption follows directly from d-separation and shows why it is such an important concept.

Definition 5 (Global Markov) $X \perp\!\!\!\perp_G Y \mid Z \implies X \perp\!\!\!\perp_P Y \mid Z$

So d-separation implies conditional independence, which is why it is so important. Finally, we see the connection between the "data realm" and the "graph realm".

Note: Since the local and global Markov assumptions are equivalent, we will refer to them as "Markov assumption" [11].

2.2.3 Faithfulness Assumption

The Markov assumption tells us that if variables are d-separated in the graph G, they are independent in the distribution P, but we need the converse. We need to get from conditional independencies in the data, which we can detect, to d-separations in the graph. That converse is known as the faithfulness assumption.

Definition 6 (Faithfulness) $X \perp\!\!\!\perp_G Y \mid Z \iff X \perp\!\!\!\perp_P Y \mid Z$ *[13]*

2.2.4 Causal Sufficiency Assumption

One of the most popular assumptions is that the data is causally sufficient, which means that no unobserved variable affects more than one observed variable. In practice, this is a very strict condition, because it is hard to know if one measured all the variables influencing the observed phenomenon. And if such unobserved variables are present they generate confounding bias. The general question that arises is then how much of the observed behavior of the system is truly causal, or whether it is due to some external, unobserved forces [1].

Definition 7 (Causal Sufficiency) *There are no unobserved confounders of any of the variables in the graph [13].*

2.2.5 Markov Equivalence

Having made all these assumptions so far, one might think we are finally ready to identify the causal graph by looking at conditional independencies completely. Still, unfortunately, that is not the case. Even having made all these assumptions (acyclicity implicitly by assuming a DAG), we can only identify the causal graph up to a certain point, which we call the "Markov equivalence class" (MEC). A markov equivalence class can be defined as follows:

Definition 8 (Markov equivalence class) *Given a graph G, we refer to its Markov equivalence class as the set of graphs that encode the same conditional independencies.*

We also say that two graphs are Markov equivalent if and only if they are in the same MEC. To get some intuition, let us look at Fig. 2.3. Here we have three distinct graphs, but they all share the same set of (conditional) independencies/dependencies - assuming they fulfill the Markov and faithfulness assumption. We know that a node X is independent of all its non-descendants, given its parents (1). We see that all graphs fulfill $X \perp\!\!\!\perp_P Y \mid Z$. Now taking into account faithfulness as well, we further get $X \not\perp\!\!\!\perp_P Z \mid \emptyset$, $Y \not\perp\!\!\!\perp_P Z \mid \emptyset$ and $X \not\perp\!\!\!\perp_P Y \mid \emptyset$. In contrast to these three graphs, the immorality (Fig. 2.4, which is charaterized by two nodes collding onto one other) is alone in its MEC, which motivates the following finding of Verma and Pearl [26]:

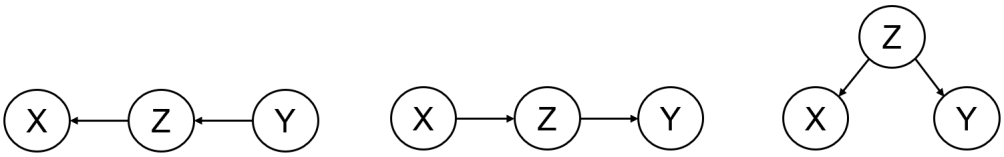

Figure 2.3: Three Markov equivalent graphs

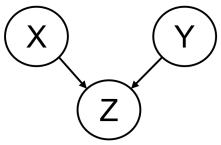

Figure 2.4: Immoralities are their own Markov equivalence class

Theorem 1 *Two graphs are Markov equivalent if and only if they have the same skeleton and same immoralities.*

where the skeleton of a graph is just the graph itself, but every edge is made undirected.

2.3 Independence-based

We can finally start looking at some algorithms. We start with the so-called independence-based algorithms. As the name already suggests, they work by checking for (conditional) independencies in the data to construct the graph. So exactly what we have motivated so far. To be a little more precise, they usually start with a complete undirected graph, where the nodes are our variables, and prune that graph until we get the MEC. Note that this type of graph differs from our definition of a causal graph, which is directed and acyclic. That is where the forementioned nuances come in. In this "new" definition, we say that all undirected edges tell us that the variables are dependent and that there is a causal connection between them in one of the two possible directions.

In the following, we will look at the most well-known algorithm of this class, improvements upon it, and give an outlook to further algorithms of this class.

2.3.1 PC-Algorithm

The PC algorithm [24], named after its two inventors (Peter Spirtes and Clark Glymour), is the most well-known algorithm of this class, mostly because of the following reasons: being one of the earliest algorithms in its category and most other algorithms (in this class) build upon it.

2.3.1.1 Assumptions

- Acyclicity
- Causal Sufficiency

- Markov
- Faithfulness

2.3.1.2 Algorithm Outline

The PC algorithm directly follows from the finding of Verma and Pearl (1).

1. Identify skeleton
2. Identify immoralities and orient them
3. Orient qualifying edges

It is important to note that the conditioning sets get larger with each iteration in the second step.

2.3.1.3 Example

To get a feeling for how the PC algorithm works, we need to know what the true graph looks like, our ground truth so to speak: see f) of (2.5) We start with a), a complete undirected graph.

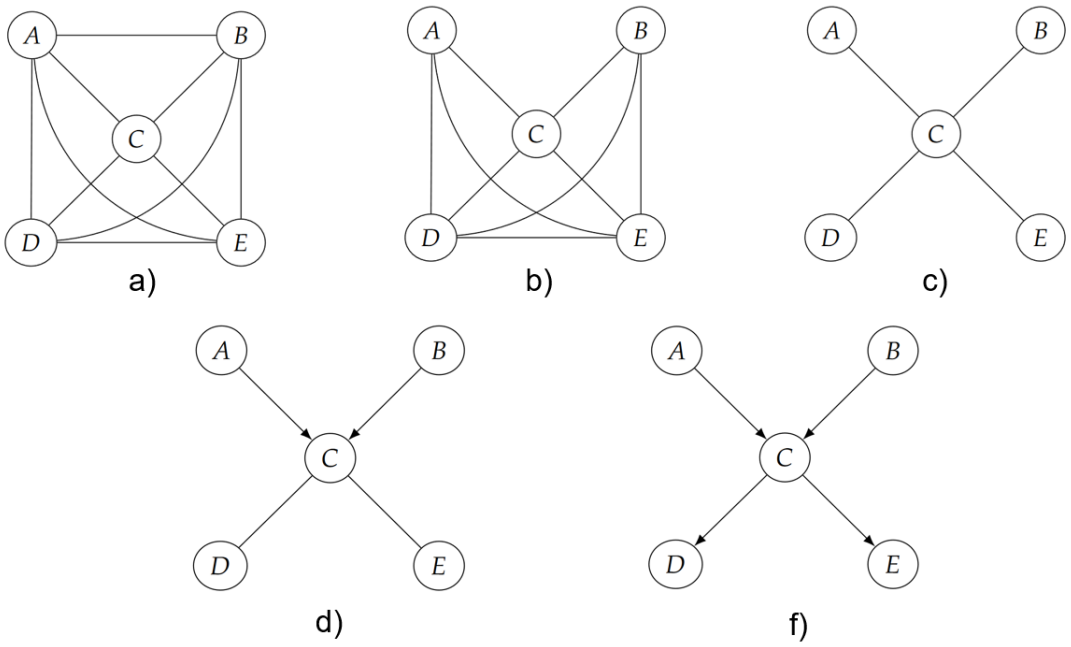

Figure 2.5: Example for PC execution [13].

Afterward, we start with step one: Identifying the skeleton. We do that by incrementally increasing the size of the conditioning set. Starting with size zero (empty conditioning set) resulting in b), which remains after removing $X - Y$ edges where $X \perp\!\!\!\perp Y \mid \emptyset$ and X,Y are nodes of the graph: $X, Y \in \{A, B, C, D, E\}$. Increasing the size of the conditioning set to one (here: it contains only the node C) results in c), where we remove all edges $X - Y$ if $X \perp\!\!\!\perp Y \mid Z$. As we

can see, we now identified the skeleton of our true graph and can move on to step two. After orienting all immoralities, we get the graph d). In the third and final step, we take advantage of the fact that we might be able to orient more edges since we know we discovered all of the immoralities in the previous step. In our case, that means that we can orient both of the undirected edges left in the graph to what we see in f). That is because if any of the edges were in the opposite direction, there would be an extra immorality, which we would have found in step two. As it turns out, we are lucky that we can orient all the edges, but it is important to understand that it is not the case in general. We have seen earlier that we can only identify the graph up to its MEC in general. In this case, the MEC of our true graph contains one graph.

2.3.1.4 Runtime

The bulk of the required time lies in the amount of independence tests we have to perform. Which results in a worst case upper bound of: $O(\frac{n^2*(n-1)^{k-1}}{2})$ where $k = max\{deg(v) \mid v \in V\}$ [29].

2.3.1.5 Problems

There are a few problems with the PC algorithm. As we have seen, the runtime is exponential in the degree of the graph. In addition to that, it is unstable [24], which means that errors made in the process tend to get larger and larger as we progress and order-dependent [2].

2.3.2 Improved versions of PC-Algorithm

Having seen which problems PC has, lets look at some major improvements. Here is a small list with new variants, what they improved and where to find them.

- Stable-PC: Fixed the instability problem [2]
- Parallel-PC: Parallelized the independence tests [29] [12] [28]
- PC-reverse: Start pruning with large conditioning sets and move down [29]

Note that these variants can and actually have been combined into a single one, so basically, the “PC-stable-reverse-parallel”. In the following figure (Fig. 2.6), you can see a comparison of different versions. Here we see, that the PC-reverse algorithm can achieve about a 6-fold speedup compared to the PC algorithm in graphs with 85+ nodes and the parallel versions improve upon that even more with a 825-fold speedup [29].

2.3.3 Further Independence-based algorithms

The FCI-Algorithm is very similar to the PC-Algorithm; one might even argue it should also fall into the list of improved PC-Algorithms. Nevertheless, it is historically the first to drop the Causal Sufficiency assumption, why we decided to put it in this category.

- Fast Causal Inference (FCI): Dropping Causal Sufficiency [24]

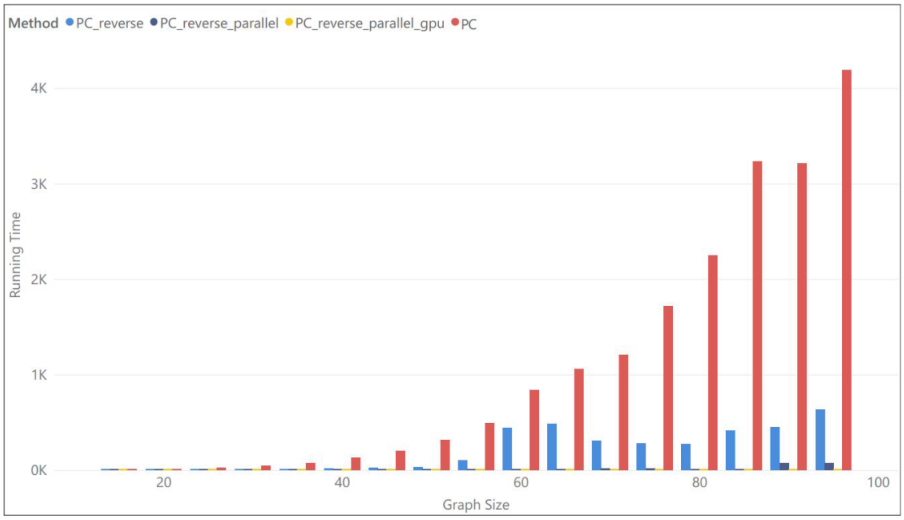

Figure 2.6: Runtime of different algorithms in seconds [29].

- Cyclic Causal Dependency (CCD): Dropping Acyclicity [18] [19]
- Cyclic Causal Inference (CCI): Dropping both Causal Sufficiency and Acyclicity [25]

2.3.4 Limits of Independence-based models

We have seen some significant improvements upon the original independence-based algorithms. However, even the improved versions still share the same problem: they can only identify the graph up to the MEC. The intuition behind that is clear; when we only look at (conditional) independencies and different graphs have the same, we can not differentiate between them.

2.4 Semi-Parametric

To solve the problem of not identifying the graph further than its MEC, we will look at the Semi-Parametric models. In essence, they are a particular case of the structural causal models (1.3.2), where we make assumptions about the type of function f and the corresponding noise U. So we have two hyper-parameters motivating the name of these models. Like the independence-based methods, we will cover one of the more essential and easy-to-understand algorithms, look at improvements upon it, and finish by giving an outlook on further algorithms.

2.4.1 (ICA-) Linear Non-Gaussian Acyclic Models

2.4.1.1 Assumptions [22]

- Acyclicity
- Causal Sufficiency

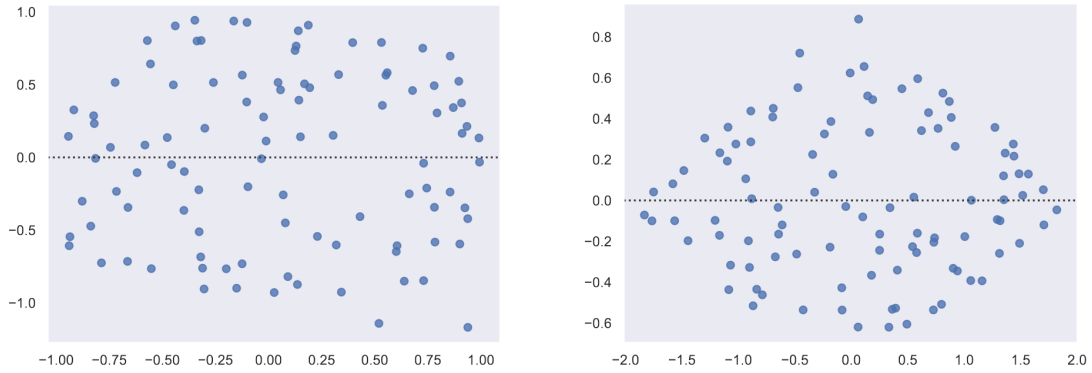

Figure 2.7: Residuals of linear models (in both directions) fit the linear non-Gaussian data. Left: Causal direction; Right: Anti-Causal direction [13].

- f is a linear function
- Non-Gaussian Noise

2.4.1.2 Example

Looking at the simple case of trying to determine in the two-variable case, in which direction the causal association goes. Remember that this is not solvable with independence-based models (1). There are two possible SCMs (1.3.2) we can create:

1. $X_1 := f_1(X_2) + U_1$
2. $X_2 := f_2(X_1) + U_2$

In both cases, we assume that the noise is independent of the dependent variable: $X_1 \perp\!\!\!\perp U_1$ and for the second SCM $X_2 \perp\!\!\!\perp U_2$. Let us now assume that 1. describes the true causal relationship. When we now perform a linear regression in both cases, we will find that our assumption of independence is violated in the second case (Anti-Causal direction). Therefore we can identify which the true causal direction is, as we can see in Fig. 2.7.

This two-variable approach can easily be generalized: first identify the variables which are independent, then use the two variable approach on each pair.

2.4.1.3 Problems

The main problems of the (ICA-) LiNGAM are [20]:

- Getting stuck in local optima
- Scale-dependent calculations (runtime grows significantly as the number of variables increases)
- Estimating dense graph even for sparse ground truth causal graphs

2.4.2 Improvements upon LiNGAM and further algorithms

- DirectLiNGAM: Guaranteed to converge to the right solution within a small fixed number of steps if all the model assumptions are met, and the sample size is infinite. We can provide prior knowledge to the algorithm [23].
- ParaLiNGAM: Parallel version of DirectLiNGAM, which outperforms the latter by 500x - 4600x depending on the exact algorithm [28]. [20]
- Nonlinear Additive Noise Model: Represents the effect as a non-linear function of the cause plus independent error [7].
- Post-nonlinear Causal Model (PNL): As a direct extension of LiNGAM and non-linear additive noise model, the PNL takes into account the non-linear influence from the cause, the noise effect, and the possible measurement distortion in the observed variables Therefore, our assumptions are weaker, but full identifiability is not guaranteed like with LiNGAM [7] [30].

2.5 Comparison

	PC-reverse-parallel [1]	CCI [2]	ParaLiNGAM[3]	Post-nonlinear[4]
Markov	✓	✓	✓	✓
Faithfulness	✓	✓	✗	✗
Acyclicity	✓	✗	✓	✓
Causal Sufficiency	✓	✗	✓	✓
Non-Parametric	✓	✓	✗	✗
Deterministic	✓	✓	✓	✓
Time Complexity	825x speedup	?	4600x speedup	Model dependent
Data type	?	?	Continuous	?
Identifiability	MEC	MEC	Fully	Model dependent
Other assumptions			f Linear	Model dependent
			Noise Non-Gaussian	

Table 2.1: Comparison of algorithms for observational data. Algorithms: [1]2.3.2, [2]2.3.3, [3]2.4.2, [4]2.4.2

3 Causal Discovery from Interventional Data

So far, we have seen how to learn cause-effect relationships from observational data alone without any interventions. This chapter explains how we learn cause-effect relationships by intervening in the underlying system. We begin by introducing single- and multi-node interventions and how we utilize them to identify the causal graph fully. Here we look at the naive and more sophisticated algorithms, thereby answering three problems we might face when performing interventions. We end this chapter by comparing the algorithms to each other.

3.1 What are Interventions?

There are two types of interventions: structural and parametric. The idea behind structural interventions is to set our target variable/node Y to a given value. With that eliminating the influence of the initial parents, that is why they are called structural. On the other side, we have parametric interventions. Here we have parameters that influence how each parent influences Y. So we change the conditional distribution of Y given its parents. Formally that works by introducing weights for every parent which we put together to a set Θ, as seen in Fig. 3.1. A practical example, would be that we put the weight of every parent to the exact same value, therefore resulting in the original structure of not having intervened at all (left image in Fig. 3.1). Note that parametric interventions, as described above, form a superset of structural interventions. Still, there is also the definition that the set of parametric interventions does not contain the set of structural interventions in literature [13].

For simplicity reasons, we will refer to structural interventions as interventions.

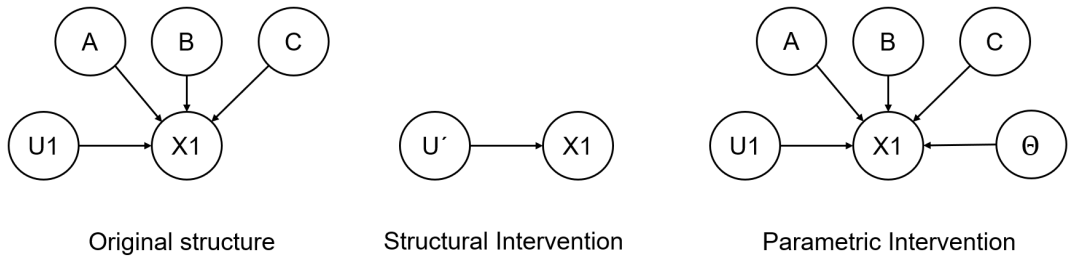

Figure 3.1: Comparison of structural and parametric intervention. Θ represents the set of parameters.

3.2 Single-Node Interventions

Single-Node interventions refer to intervening on maximally one variable at a time. To understand the basic idea, we first look at the two-variable case and afterward generalize our findings. Assuming we have two nodes X_1 and X_2, which are dependent, we want to determine whether the causal direction is from X_1 to X_2 or the other way around. Note that this is not possible by purely looking at the observational data since both graphs are in the same MEC. We model these two options with the following Structural Causal Models (1.3.2):

1. $X_1 := U_1, X_2 := f_2(X_1, U_2)$
2. $X_2 := U_2, X_1 := f_1(X_2, U_1)$

We now perform an intervention on either X_1 or X_2; here, we choose to go with X_2. This results in both Structural Causal Models changing:

1. $X_1 := U_1, X_2 := U_2'$
2. $X_2 := U_2', X_1 := f_1(X_2, U_1)$

As we can see, X_1 and X_2 are now independent in the first case and still dependent in the second case. We can differentiate between these two cases by looking for dependency after the intervention.

Generalizing this approach is quite straightforward; it thereby suffices to look at the three variable cases. Here we need two interventions to identify the graph completely; we choose to intervene on X_3 and X_2. By combining the information we gain through both interventions, we completely identify the graph; we go through the entire process because that is not obvious. (see: Fig. 3.2)

1. Find independencies by not intervening on anything: We find that all variables are dependent on each other.
2. Intervene on X_3: We find that now X_3 is independent of X_1 and X_2.
3. Intervene on X_2: We find that now X_2 is independent of X_1
4. Combine the previous information: X_1 and X_2 are parents of X_3 since they are dependent in the first step and independent in the second step. X_1 is a parent of X_2, same reasoning.

That was the naive approach, where $n-1$ interventions are necessary in the worst case (complete skeleton) [4].

3.3 Multi-Node Interventions

In contrast to single-node interventions, multi-node interventions allow us to intervene on multiple variables simultaneously. The most general form is that there is no limit on the size of the intervention set, but some algorithms try to answer what happens when we limit the

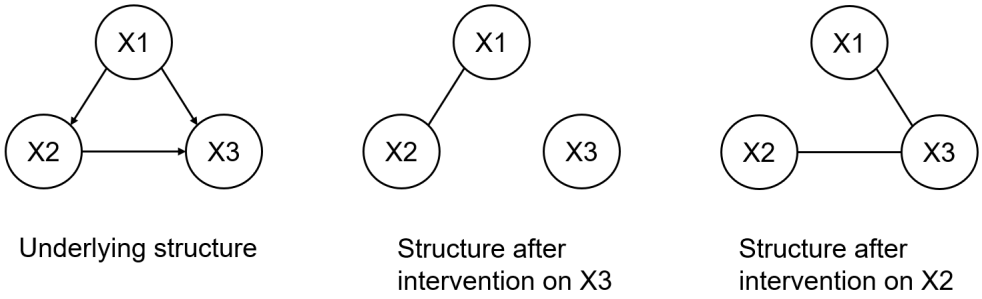

Figure 3.2: Effects of interventions in a three-variable setting.

size, as we will see in 3.4. Using multi-node interventions, the time complexity drops to only $\lfloor log_2(n)) \rfloor + 1$, which is sufficient and necessary in the worst case (complete skeleton) [5].

However, we can use a trick to speed it up further respectively give a tighter upper bound: Identifying the MEC before doing interventions. This results in a maximum of (and worst-case necessary) $\lceil log_2(c) \rceil$ interventions, where c is the size of the largest clique in the graph [3] [8]. Note that this is not significantly faster in the worst case since the largest clique in a complete graph is n. Besides that, one also needs to pay attention to the time it takes to find the MEC.

3.4 Advanced methods

Having a logarithmic time complexity is generally considered good, but interventions could be costly or unethical. Consider, for example, trying to discover the causes of a human illness, a common scenario in modern medical science. Performing interventions with possibly harmful consequences on humans is unethical, so scientists often perform experiments on animals instead of knowing full well that the causal relationships discovered in animals may not generalize to humans.

From that, two questions naturally arise:

1. How can we reduce the number of interventions?
2. What if we have a limited budget of interventions?

Aside from that, we answer the pressing practical problem of handling unobserved confounders, which we did not dedicate much attention to so far.

3.4.1 Reducing number of interventions

3.4.1.1 Randomized algorithms

Hauser and Bühlmann [8] showed that the number of experiments required is logarithmic in the cardinality of the largest clique in the skeleton. However, their lower bounds assume that the experiment designer can not use randomization in selecting the experiments. Hu et al. [9] have proven that using randomization in the selection of the intervention sets leads to a lower bound of $O(log(log(n))$ interventions in expectation. In addition, they showed that this bound

can not be improved upon (in the worst-case). Still, looking at the average case, it is possible to only perform a constant amount of interventions under certain conditions.

So how does randomization help improve the lower bound of Hauser and Bühlmann? In essence, those lower bounds are based upon the causal graph being constructed by a powerful adversary. This adversary must pre-commit to the causal graph in advance but, before doing so, it has access to the entire list of experiments $S = \{S_1, S_2, \dots\}$ that the experiment designer will use; here $S_i \subseteq V; \forall i$. However, the experiment designer can trick the powerful adversary by randomizing the selection of the experiments.

He does that by selecting the experiments $\{S1, S2, \dots\}$ from a collection of probability distributions $P = \{P_1, P_2, \dots\}$, respectively, where distribution P_{i+1} may depend upon the results of experiments $1, 2 \dots, i$. Then, even if the adversary has access to the list of probability distributions P before it commits to the causal graph G, the expected number of experiments required to recover G falls significantly [9]. Note that the proposed algorithm assumes that we can perform multi-node interventions of unbounded size.

Regarding limited intervention size, Shanmugam et al. [21] showed that we require at least $\frac{n}{2k}$ k-size interventions in the deterministic case, and $O(\frac{n}{k} * log(log(n))$ in expectation - using a randomized algorithm.

3.4.1.2 Deterministic algorithms

We have seen that the lower bound for deterministic algorithms is $\lceil log_2(c) \rceil$, but that for itself is only a theoretical finding. One of the recently proposed algorithms needs $O(log(n) + l)$ interventions, where l is the length of the longest path in the graph. Before we look at the steps, we introduce the concept of the transitive closure of a graph. The transitive closure of a graph is its reachability matrix R, which describes if a node X_j is reachable from another node X_i, therefore $R \in V \times V$ and $R[i][j] \in \{0, 1\}$. The two-steps of the algorithm:

1. Discover the transitive closure of the graph in $O(log(n))$
2. Learn the observable graph (all edges between observed variables) in $O(l)$

An important property is that this algorithm also works when unobserved confounders are present, which is vital for the section about Learning unobserved confounders (3.4.3) [10].

3.4.2 Handling a limited budget of interventions

The problem we focus on now is: If the experimenter is allowed to perform k experiments, each of size 1, what portion of the graph could, on average, be reconstructed?

Ghassami et al. [6] showed that their greedy algorithm suffices to achieve a $1 - \frac{1}{e}$ approximation of the optimal value, where the objective function is the expected amount of edges recovered with a set of experiments $I; |I| = k$. The type of interventions is passive because all interventions need to be known beforehand and can not change in the process. A significant problem they faced is that computing the objective function is, in general, intractable for a given set. So they proposed an unbiased estimator, which has polynomial time complexity for graphs with bounded degree. They provided another efficient but slightly biased estimator for graphs

with a high degree. It is further necessary to note that they first perform an observational test, such as the PC algorithm, to identify the MEC before performing the interventions.

We show some of their results on synthetic and real graphs in Fig. 3.3. Here we see, that their greedy algorithm significantly outperforms the random (intervening on random nodes) as well as the maximum degree approach (intervening on the node with maximum degree) in their ability to orient as many edges as possible.

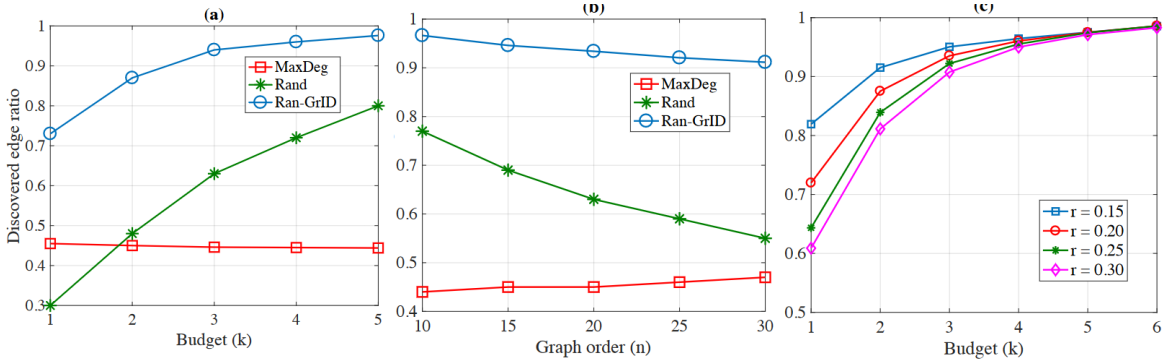

Figure 3.3: Discovered edge ratio versus (a) budget for n = 20, (b) graph orders for k = 3, (c) budget for n = 20 and different densities [6].

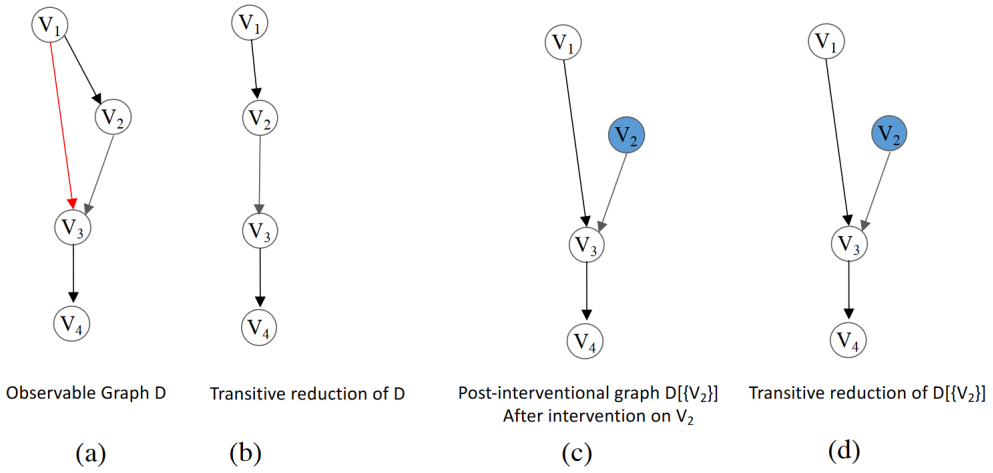

Figure 3.4: a) An example of an observable graph G without unobserved confounders (b): Transitive reduction of the graph G. The highlighted red edge (V_1, V_3) has not been revealed under the operation of transitive reduction. c) Intervention on node V_2 and its post interventional graph $G[\{V_2\}]$ d) Since all parents of V_3 above V_1 in the partial order have been intervened on, the edge (V_1, V_3) is revealed in the transitive reduction of $G[\{V_2\}]$ [10].

3.4.3 Learning unobserved confounders

Building upon the findings of Kocaoglu et al. [10], which we described in (3.4.1.2), we now introduce a randomized approach to step two, and how to find the latent variables - step three.

The basic idea behind the randomized algorithm is to intervene randomly, then compute the transitive closure of the post-interventional graph (after intervening on a set of variables), compute the transitive reduction (graph with minimum edges, which has the same transitive closure), and then accumulate all the edges found in the transitive reduction at every stage. This algorithm needs $O(d^2 * log(n))$ interventions in expectation. To better understand what a transitive closure and transitive reduction is, look at the following example: Fig. 3.4.

Regarding step three: learning unobserved confounders by looking at the observable graph, Kocaoglu et al. improved exponentially upon the naive baseline approach. Their algorithm only requires $O(d^2 * log(n))$ interventions, whereas the naive approach requires $\Theta(n^2)$. However, there is an additional assumption, which states that the observable graph has a constant degree. The algorithm itself is made up of two steps, which combined find all unobserved edges:

1. Find unobserved edges between Non-adjacent Nodes
2. Find unobserved edges between adjacent Nodes

Explaining this algorithm would break the bounds for this report; we still want to note that the second step needs a new approach called “do-see” test, which you can also find in [10].

3.5 Comparison

	LOD [1]	LOR [2]	LearnLatent [3]	RanGrID [4]
Markov	✓	✓	✓	✓
Faithfulness	✓	✓	✓	✓
Acyclicity	✓	✓	✓	✓
Causal Sufficiency	✗	✗	✗	✓
Find unob. confounders	✗	✗	✓	✗
Non-Parametric	✓	✓	✓	✓
Deterministic	✓	✗	✗	✗
Time Complexity	$O(log(n)+l)$	$O(dlog^2(n))$	$O(d^2log(n))$	$O(kNn^{\Delta+1})$
Data type	Discrete	Discrete	Discrete	?
Intervention type	Multi-Node	Multi-Node	Multi-Node	Single-Node
	Adaptive	Adaptive	Adaptive	Passive
Other assumptions	CI Oracle	CI Oracle	?	?
		Constant degree		

Table 3.1: Comparison of algorithms working with interventions. Algorithms: [1]3.4.1.2, [2]3.4.3, [3]3.4.3, [4]3.4.2

4 Conclusion

We found that focusing only on the interventional case is not sufficient since interventions can be expensive, unethical, and sometimes outright impossible. Therefore, one also needs to know about the purely observational case and a combination of both. In addition to that, we found that approaches that combine both cases also combine the respective strengths: observational approaches can be made automatically (since they do not require interventions), and interventional approaches identify the true causal graph fully.

We have seen that independence-based methods, like "PC", are limited in their identifiability because they can only identify the true causal graph up to its Markov Equivalence Class. To resolve that issue, we can use specific Semi-Parametric methods (e.g., LiNGAM), which give us full identifiability but in exchange for further assumptions upon our data. To avoid that trade-off, we should use interventional methods - given it is possible - or even combinations like "RanGrID".

However, there are still limiting factors, most notably the assumptions we have to make - especially the Causal Sufficiency assumption, which often does not hold in practice. Furthermore, we need a "proxy" for causality since there is no statistical test for it. The most general criterion in literature are tests for conditional independence, which are not entirely reliable in the case of limited data. Another limitation concerning observational approaches is that they often have a sub-optimal time complexity (squared, cubed, or even exponential). Recent papers focused on improving time complexity by creating a parallel version of the algorithms and achieved significant speedup ranging from $825x$ to $4600x$.

Further work could be to evaluate the algorithms with regard to practical applicability through experiments on a wide range of real-world datasets, find ways to reduce the strength of assumptions needed and make the already known algorithms more robust/consistent when data does not fulfill the assumptions.

Our Review of observational approaches and interventional ones is unique to the best of our knowledge but, more importantly, provides understanding and some of the best tools for application in a wide range of research fields, regardless of whether interventions are possible or not.

Bibliography

[1] Elias Bareinboim and Judea Pearl. "Causal inference and the data-fusion problem". In: *Proceedings of the National Academy of Sciences* 113.27 (2016), pp. 7345–7352.

[2] Diego Colombo, Marloes H Maathuis, et al. "Order-independent constraint-based causal structure learning." In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 3741–3782.

[3] Frederick Eberhardt. "Almost optimal intervention sets for causal discovery". In: *arXiv preprint arXiv:1206.3250* (2012).

[4] Frederick Eberhardt, Clark Glymour, and Richard Scheines. "N-1 experiments suffice to determine the causal relations among n variables". In: *Innovations in machine learning*. Springer, 2006, pp. 97–112.

[5] Frederick Eberhardt, Clark Glymour, and Richard Scheines. "On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables". In: *arXiv preprint arXiv:1207.1389* (2012).

[6] AmirEmad Ghassami et al. "Budgeted experiment design for causal structure learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1724–1733.

[7] Clark Glymour, Kun Zhang, and Peter Spirtes. "Review of causal discovery methods based on graphical models". In: *Frontiers in genetics* 10 (2019), p. 524.

[8] Alain Hauser and Peter Bühlmann. "Two optimal strategies for active learning of causal models from interventional data". In: *International Journal of Approximate Reasoning* 55.4 (2014), pp. 926–939.

[9] Huining Hu, Zhentao Li, and Adrian R Vetta. "Randomized experimental design for causal graph discovery". In: *Advances in neural information processing systems* 27 (2014).

[10] Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. "Experimental design for learning causal graphs with latent variables". In: *Advances in Neural Information Processing Systems* 30 (2017).

[11] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[12] Thuc Duy Le et al. "A fast PC algorithm for high dimensional causal discovery with multi-core PCs". In: *IEEE/ACM transactions on computational biology and bioinformatics* 16.5 (2016), pp. 1483–1495.

[13] Brady Neal. "Introduction to causal inference from a machine learning perspective". In: *Course Lecture Notes (draft)* (2020).

[14] Judea Pearl. "Causal inference as computational learning". In: Talk at the NIPS 08 workshop on Causality: objectives and assessment. 2008.

[15] Judea Pearl. *Causality*. Cambridge university press, 2009.

[16] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[17] Jean-Philippe Pellet. "Effective Causal Analysis: Methods for Structure Learning and Explanations". PhD thesis. ETH Zurich, 2010.

[18] T Richardson. "A discovery algorithm for directed cyclic graphs. Uncertainty in Artificial Intelligence". In: *Proceedings, 12th Conference, Morgan Kaufman, CA*. 1996.

[19] T Richardson and P Spirtes. "Automated causal discovery under linear feedback". In: *Computation, Causation, and Discovery* (1999), pp. 253–302.

[20] Amirhossein Shahbazinia, Saber Salehkaleybar, and Matin Hashemi. "ParaLiNGAM: Parallel Causal Structure Learning for Linear non-Gaussian Acyclic Models". In: *arXiv preprint arXiv:2109.13993* (2021).

[21] Karthikeyan Shanmugam et al. "Learning causal graphs with small interventions". In: *Advances in Neural Information Processing Systems* 28 (2015).

[22] Shohei Shimizu et al. "A linear non-Gaussian acyclic model for causal discovery." In: *Journal of Machine Learning Research* 7.10 (2006).

[23] Shohei Shimizu et al. "DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model". In: *The Journal of Machine Learning Research* 12 (2011), pp. 1225–1248.

[24] Peter Spirtes et al. *Causation, prediction, and search*. MIT press, 2000.

[25] Eric V Strobl. "A constraint-based algorithm for causal discovery with cycles, latent variables and selection bias". In: *International Journal of Data Science and Analytics* 8.1 (2019), pp. 33–56.

[26] T Verma and J Pearl. "Equivalence and synthesis of causal models (Technical Report)". In: *Cognitive Systems Laboratory, University of California at Los Angeles* (1991).

[27] James Woodward. *Making things happen: A theory of causal explanation*. Oxford university press, 2005.

[28] Behrooz Zarebavani et al. "cuPC: CUDA-based parallel PC algorithm for causal structure learning on GPU". In: *IEEE Transactions on Parallel and Distributed Systems* 31.3 (2019), pp. 530–542.

[29] Kai Zhang et al. "A Fast PC Algorithm with Reversed-order Pruning and A Parallelization Strategy". In: *arXiv preprint arXiv:2109.04626* (2021).

[30] Kun Zhang and Aapo Hyvarinen. "On the identifiability of the post-nonlinear causal model". In: *arXiv preprint arXiv:1205.2599* (2012).