From the Institute of Theoretical Computer Science
of the University of Lübeck
Director: Prof. Dr. math. K. Rüdiger Reischuk

Algorithmics of Identifying Causal Effects in Graphical Models

Dissertation
for the Fulfillment
of Requirements for
the Doctoral Degree
of the University of Lübeck
from the Department of Computer Sciences

Submitted by
Benito van der Zander
from Düsseldorf

Lübeck, 2020

First referee	Prof. Dr. Maciej Liśkiewicz
Second referee	Prof. Dr. Ralf Möller
Date of oral examination	2020-08-31
Approved for printing	

Abstract

Graphical causal models represent the relationships between random variables and can predict the outcome of experiments from observed data. They are an important tool in fields like economics, social sciences, and epidemiology, where randomized experiments are impossible or unethically to perform, but a vast amount of observed data is available. Although many theoretical results are known, for many tasks involving graphical causal models no efficient algorithms are known, hindering the use of these models in artificial intelligence and for the analysis of big data.

In this thesis, we present efficient algorithms to predict the causal effect of experiments from a graphical causal model and observed data. We investigate two separate approaches for this purpose: covariate adjustment and instrumental variables.

We show that the sound and complete adjustment criterion by Shpitser et al. for DAGs becomes equivalent to a d-separation test – a conditional independence test – after a simple graph modification. From this, we derive efficient algorithms to verify, construct, and enumerate adjustment sets as well as minimal and minimum adjustment sets.

We generalize the adjustment criterion from DAGs to maximal ancestral graphs (MAGs), which represent ancestral relationships between variables and remain valid when the data is confounded or biased by unknown variables. We also generalize it to completed partially directed acyclic graphs (CPDAGs) and restricted chain graphs (RCGs), which represent an entire (sub)class of Markov equivalent DAGs at once. This allows our algorithms to be used for incomplete models that do not represent all variables or models in which the direction of causal influences is not known.

Prior to developing the algorithms for adjustment sets, we develop the corresponding algorithms for separating sets. Thereby we improve the runtime of the previously best-known algorithms for DAGs and generalize them to MAGs, CPDAGs, and RCGs.

Afterward, we study the algorithmics of instrumental variables (IVs). IVs are frequently used with linear structural equation models (SEMs) to estimate the causal effect under the assumption that all influences in the model are linear. This additional assumption allows one to calculate causal effects that cannot be calculated by adjustment or in non-linear models. We show that it is NP-complete to decide whether a given variable is a conditional IV. Nevertheless, we present an efficient algorithm to find at least one conditional IV if one exists in a given model. Furthermore, we investigate special cases in which conditional IVs are also efficiently verifiable.

We obtain similar results for instrumental sets, a generalization of instrumental variables. It is NP-complete to verify a given generalized instrumental set, but certain simple cases of instrumental sets can be found efficiently.

Zusammenfassung

Graphische, kausale Modelle repräsentieren Zufallsvariablen mitsamt ihren gegenseitigen Einflüssen als Graphen, und können die Ergebnisse von Experimenten aus rein beobachteten Daten vorhersagen. Diese Modelle haben große Bedeutung in Forschungsbereichen wie Epidemiologie, der Wirtschaftswissenschaft und der Sozialwissenschaft, in denen Zufallsexperimente unmöglich sind oder unethisch wären, jedoch große Datenmengen zur Verfügung stehen. Obwohl graphische, kausale Modelle schon intensiv erforscht wurden, sind die meisten Ergebnisse theoretischer Natur und es fehlen Algorithmen, um die Modelle in künstlicher Intelligenz oder zur Analyse von Big Data anzuwenden.

In dieser Arbeit entwickeln wir effiziente Algorithmen, um die kausalen Effekte von Experimenten aus gegebenen beobachteten Daten und dem dazugehörigen graphischen kausalen Model zu berechnen. Wir verwenden dazu zwei unterschiedliche Ansätze: das Adjustieren für Störfaktoren und die Instrumentvariablenmethode.

Wir zeigen, dass das korrekte, vollständige Adjustierungskriterium für DAGs von Shpitser et al. äquivalent zu einem d-Separation-Test in einem modifizierten DAG ist. Daraus folgt, dass d-Separationsalgorithmen erkennen können, welche Variablen für die Adjustierung geeignet sind, und wir erhalten effiziente Algorithmen, um diese Variablen zu finden und aufzulisten. Wir entwickeln weiterhin Algorithmen, um minimale und kleinstmögliche Mengen von Variablen für die Adjustierung zu finden.

Wir verallgemeinern das Adjustierungskriterium von DAGs zu maximalen Ahnengraphen (MAGs), die Vorfahrrelationen statt Eltern-Kind-Relationen kodieren und deshalb auch gültig bleiben, wenn die Daten von unbekannten Störfaktoren beeinflusst werden, solange diese Störfaktoren die Vorfahrrelationen nicht beeinflussen. Weiterhin verallgemeinern wir das Kriterium zu vollständigen teilweise gerichteten azyklischen Graphen (CPDAGs) und eingeschränkten Kettengraphen (RCGs), welche eine (Teil-)Klasse von Markov äquivalenten DAGs repräsentieren. Daraus folgern wir, dass unsere Algorithmen auch gültige Ergebnisse liefern, wenn das kausale Modell nicht vollständig bekannt ist oder die Richtung von manchen kausalen Einflüssen unbekannt ist.

Eine nötige Grundlage für unsere Adjustierungsalgorithmen sind effiziente Algorithmen für d-Separation. Daher verbessern wir zu Beginn der Arbeit die Laufzeit der bekannten Separationsalgorithmen für DAGs und generalisieren sie zu MAGs, CPDAGs und RCGs.

Schließlich untersuchen wir die Instrumentvariablenmethode. Instrumentvariablen werden in linearen Gleichungsmodellen (SEMs) verwendet, um den kausalen Effekt unter der Annahme, dass alle Effekte linear sind, zu bestimmen. Instrumentvariablen können kausale Effekte bestimmen, die weder durch Adjustierung noch in nicht linearen Modellen bestimmt werden können. Wir zeigen, dass es ein NP-vollständiges Problem ist zu entscheiden, ob eine gegebene Variable eine bedingte Instrumentvariable ist. Trotzdem entwickeln wir einen effizienten Algorithmus, der eine bedingte Instrumentvariable findet, falls mindestens eine bedingte Instrumentvariable im Graphen existiert. Weiterhin untersuchen wir eine Einschränkung von bedingten Instrumentvariablen, die auch effizient zu erkennen sind.

Ähnliche Ergebnisse bekommen wir für Instrumentvariablenmengen, eine Generalisierung von Instrumentvariablen. Es ist NP-vollständig zu entscheiden, ob eine Menge von Variablen eine Instrumentvariablenmenge ist. Jedoch existieren Spezialfälle, in denen eine Instrumentvariablenmenge effizient gefunden werden kann.

Contents

1 Introduction

Many important questions like "Does smoking cause lung cancer?", "How can we prevent diabetes?", "Is global climate change caused by man-made carbon dioxide emissions?", or "Do lower taxes lead to more economic growth?" cannot be answered by randomized controlled trials. That is because the required trials would be unethically, too expensive, or flat-out impossible. For example, in a study of smoking, it would be unacceptable to expose a treatment group consisting of randomly chosen smokers and non-smokers to a potential carcinogen. Econometrists cannot change the taxes in random countries, and climatologists do not have multiple Earths to compare the effects of different levels of carbon dioxide emissions.

Hence, such questions need to be answered from observed data. Graphical causal models are a popular method to obtain these answers in fields like econometrics [AP08; Imb14], social sciences [Elw13], and epidemiology [RGL08]. Such models represent the mechanisms that are assumed to generate an observed joint probability distribution. They reveal correlations and conditionally independence relationships between the variables of the model and predict how the relationships would be changed by an *intervention* that changes some variables experimentally. *Directed acyclic graphs* (DAGs) are a standard model that uses directed edges between nodes to represent cause and effect [Pea09].

In this thesis, we study the *identification* of the causal effect of variables $\mathbf{X}$ on variables $\mathbf{Y}$, i.e., predicting the outcome of an experiment changing $\mathbf{X}$ from the graphical model and observed data. An experiment like a randomized controlled trial investigates the effect of variables $\mathbf{X}$ on variables $\mathbf{Y}$ by performing an intervention that assigns fixed values to the variables $\mathbf{X}$ and then measures the resulting changes in the variables $\mathbf{Y}$. If we only have access to the graphical model and observed data, we cannot change any variables and need to rely on the mechanism encoded in the graphical model to predict the outcome of the intervention. We will focus on two most prominent methods of identifying the causal effect: *covariate adjustment* and *instrumental variables*.

For example, the model in Figure 1.1 can be used to study the influence of education (represented as variable *LE*) on diabetes risk (*D*) in the hope of preventing diabetes by improving education [TL11; ZLT19; RGL08, Chapter 12]. The model assumes there are three direct causes of diabetes risk: the genetic risk of the mother to develop diabetes (*MR*), her actually developing diabetes (*MD*), and the level of education (*LE*). Neither *MR* nor *MD* affect *LE*. A fourth variable, family income (*FI*), is assumed to affect *MD* and *LE*, but to have no direct effect on *MR* and *D*. It affects *D* through its influence on *MD* indirectly.

In a randomized controlled trial, one could assign random levels of education to different groups of people and measure the resulting probability that they develop diabetes. This probability is also called the *total causal effect* of LE on D. We denote it as $P(D = d \mid do(LE = le))$,

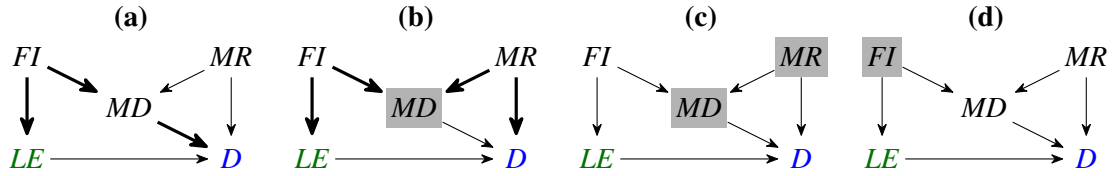


Figure 1.1: Causal DAG [TL11; ZLT19; RGL08, Chapter 12] describing the effect of low education (*LE*) on diabetes risk (*D*) with the covariates: family income (*FI*), mother's genetic risk to develop diabetes (*MR*), and mother's diabetes (*MD*). In cases (a) resp. (b) the biasing path resulting from no adjustment resp. adjusting *MD* is highlighted. Cases (c) and (d) show possible adjustments that prevent all bias between *LE* and *D*.

meaning the probability that variable D takes value d after an experimental intervention that forces variable LE to have value le.

Such an intervention removes all influences of other variables on LE as if there were no edges into LE in the graphical model. Thus, the causal effect of LE on D can predict whether the risk of diabetes is lowered by improving education.

Identifying causal effects via adjustment sets. In an observational study, we only know how many people exist for various combinations of variable values, i.e., the *joint probability distribution* P of all variables. From the joint distribution, one can obtain the *conditional probability* $P(D = d \mid LE = le)$, the probability that variable D takes value d given that variable LE has value le. Observing that people with low education are more likely to have diabetes than people with high education might imply that improving education could prevent diabetes.

However, the conditional probability is not the same as the causal effect. If some other factors are causing both low education and diabetes, these factors influence the conditional probability, but they are not supposed to influence the causal effect. Indeed, the model includes such a *confounding* factor: the family income, which affects *LE* as well as *MD* and thus, indirectly, *D*. The DAG encodes this indirect influence as *biasing path* $LE \leftarrow FI \rightarrow MD \rightarrow D$, which is highlighted in Figure 1.1 (a). In a randomized trial, there would be no influence of *FI* on *LE*, so, if one would use the conditional probability to estimate the causal effect of *LE* on *D*, the estimate would be biased by this influence.

To remove this bias we *adjust* for *MD*, which means to sum the conditional probability $P(D \mid LE, MD)$ over all possible values of *MD* separately as $\sum_{md} P(D \mid LE, MD = md)P(MD = md)$. Since the value of *MD* is fixed, *MD* is no longer affected by *FI*, so the biasing path $LE \leftarrow FI \rightarrow MD \rightarrow D$ is blocked. However, the adjustment opens another biasing path $LE \leftarrow FI \rightarrow MD \leftarrow MR \rightarrow D$ as highlighted in Figure 1.1 (b). This path is not open without the adjustment for *MD* since *MD* is an effect of both *FI* and *MR*, so *FI* and *MR* cannot influence each other through *MD*. However, *FI* and *MR* can become correlated given a fixed value of *MD*. For instance, if the income is high and the mother has diabetes, it is more likely that she has a high genetic risk. If the income is low and she does not have diabetes, one can conclude that she has a low genetic risk. This path can be blocked by also adjusting for *MR* as shown in Figure 1.1 (c)[1].

[1] A path is blocked if one adjusts for a variable that is a cause of one of its adjacent nodes on the path. Since *MD*

Thus, all biasing paths are blocked by adjusting both MD and MR, and we can calculate the causal effect as

$$P(D = d \mid do(LE = le)) = \sum_{md,mr} P(D = d \mid LE = le, MD = md, MR = mr)P(MD = md, MR = mr).$$

Rather than adjusting for *MR* and *MD*, one can also adjust for *FI* alone as shown in Figure 1.1 (d). This also removes all bias since all biasing paths between *LE* and *D* contain *FI* as a cause. So far, we have assumed that all variables are *observed*. It may happen that some variables $\mathbf{U}$ are not observed, and we do not know their probability $P(\mathbf{U})$. If *FI* was unobserved, e.g., because it was not recorded during the study, the only possible adjustment would be *MR* and *MD*. If *MR* was unobserved, e.g., because the researchers did not have the equipment to measure genetic risk, we must adjust at *FI*. If both *FI* and *MR* were unobserved, it would be impossible to calculate the causal effect by adjustment.

This example shows that choosing a set of variables for adjustment is not an easy task. We will investigate the following problem: given a causal structure as a DAG and sets of variables $\mathbf{X}$ and $\mathbf{Y}$, find a set of variables $\mathbf{Z}$ such that for all probability distributions compatible with the DAG, $P(\mathbf{Y} = \mathbf{y} \mid do(\mathbf{X} = \mathbf{x})) = \sum_{\mathbf{z}} P(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}, \mathbf{Z} = \mathbf{z})P(\mathbf{Z} = \mathbf{z})$ holds. A set $\mathbf{Z}$ that satisfies this equation is called an *adjustment set*. Since there are infinitely many probability distributions compatible with the DAG, we want to find the set $\mathbf{Z}$ without reasoning about probability distributions by only considering properties of the DAG. We describe this problem more formally in Section 2.3 and Chapter 4.

Pearl's *back-door criterion* [Pea93; Pea09] is the standard method of choosing an adjustment set $\mathbf{Z}$ in DAGs. However, his criterion is not complete and might falsely tell us that adjustment is not possible in a certain model. Shpitser et al. have discovered a sound and complete criterion [SVR10], i.e., a criterion that is satisfied by a set of variables $\mathbf{Z}$ if and only if $\mathbf{Z}$ is an adjustment set. A crucial condition of their criterion is that $\mathbf{Z}$ blocks all biasing paths of a certain kind. This condition is difficult to handle in algorithms since a straightforward implementation would enumerate all biasing paths and test whether each path is blocked, which requires an exponential amount of time.

We will derive a *constructive back-door criterion* from Shpitser's criterion, which changes the condition from blocking biasing paths to blocking all paths in a new DAG obtained by removing the first edge of causal (i.e., directed) paths from the nodes $\mathbf{X}$ to $\mathbf{Y}$. All causal paths contain such an edge, so all paths remaining in the new DAG are biasing. Nodes on causal paths can be calculated as the intersection of ancestors and descendants, so the edges can be removed efficiently without enumerating causal paths.

Determining whether all paths are blocked in a DAG is simpler than determining whether all biasing paths are blocked since the algorithms only need to store which nodes are reachable by any path rather than tracking the paths themselves. Reachability searches are a standard problem and there exist many algorithms to solve them in causal DAGs [ADC96; Sha98; TPP98; TL11]. These algorithms solve tasks like deciding if all paths are blocked, finding a set $\mathbf{Z}$ that blocks all paths, finding a minimal or minimum set $\mathbf{Z}$, or enumerating all such sets. We will generalize these algorithms to allow additional constraints like excluding certain nodes, which is a necessary constraint when using the algorithms for adjustment sets. We will also improve the runtime of finding minimal sets.

is an effect on the path in Figure 1.1 (b), the path is not blocked by *MD*.

Adjustment sets in generalized graphical causal models. Many other factors besides the variables in Figure 1.1 might lead to diabetes, like a bad diet or a lack of exercise. Causal models encoded as a DAG are only valid if all relevant variables are included in the graph, which is called the assumption of causal sufficiency. But one can see that an adjustment for *MR* and *MD* would remain valid if another variable (confounder) is inserted as a common cause of *MR* and *D* or of *LE* and *MD* to the model. It does not remain valid if a confounder between *FI* and *D* or *MD* and *D* is added. Adjusting *FI* is more stable since it remains valid in all these cases except in the case of a confounder between *LE* and *MD*.

Ancestral graph models have been designed to handle such additional unknown variables. These models encode ancestral relationships and conclusions drawn from them are valid for an infinite number of DAGs that have the same ancestral relationships [RS02]. For example, the ancestral relationships in Figure 1.1 imply that *FI* is an ancestor of *D*, but not of *MR*. So if we treat the graph of the figure as an ancestral graph, it also represents every DAG that extends the graph with additional confounders as common causes of two variables. But extending it with a variable that is a child of *FI* and a parent of *MR* is not allowed, since such a variable would make *FI* an ancestor of *MR*.

Maximal ancestral graph models (MAGs) additionally assume the graph contains a maximal number of edges, i.e., if it is not possible to block all paths between two variables, these variables must be connected by an edge. The graph of Figure 1.1 can be treated as a MAG. It still allows an unknown, confounder between *MR* and *D*. However, inserting a confounder U between *LE* and *D* is not allowed, because then there would be two paths $FI \to LE \leftarrow U \to D$ and $FI \to LE \to D$. The latter path can only be blocked at *LE*, which would open the former path that cannot be blocked by any variable in the model. Inserting this confounder U would be allowed if there was an edge $FI \to D$ in the MAG.

In the DAG model, a correlation between *LE* and *D* under the right adjustment means low education is causing diabetes. How can we be certain that it is not diabetes causing low education? Well, because there is an arrow from *LE* to *D* in the model, and we assume the model has been chosen correctly by a domain expert[2].

However, often it is only known that one of two variables is causing the other, but not which variable is the cause and which one is the effect. This uncertainty arises especially if the model has not been curated by a human, but learned automatically from correlations in observed data. It can be represented graphically by connecting the correlated variables with an undirected edge. From a model containing an undirected edge, one can then derive two DAGs, one in which the undirected edge has been replaced by a directed edge in one direction and one in which the directed edge points in the opposite direction. A model with multiple undirected edges would thus represent a large, finite set of DAGs with a DAG for each possible edge orientation. Some of these orientations might not be consistent with the observed data or be otherwise invalid, so we only consider sets of DAGs that are statistically indistinguishable from each other by forbidding certain edge orientations, e.g., we do not allow orientations that transform an undirected cycle to a directed cycle in the DAG. These graphs with additional undirected edges are called *chain graphs* [LW89; Fry90]. Requiring further that the set of represented DAGs is not empty, i.e., not all orientations are forbidden, leads to the class of *restricted chain graphs*.

We will generalize all our results like the constructive back-door criterion and our algorithms from DAGs to maximal ancestral graphs and restricted chain graphs, enabling one to find adjustment sets in those models efficiently. For this generalization, we investigate which kinds

[2]The expert could, for example, rely on temporal orderings since a cause occurs before its effects.

of paths can exist in the various graphical models and show that some paths are syntactically equivalent between all models. Thus, using these paths, the same separation algorithms can be used for all these models.

Identification in SEMs. In practice, researchers often assume that the relationships between all variables are linear [Bol89], using so-called *structural equation models* (SEMs). A linear SEM encodes the influence of a variable X on a variable Y by a single parameter that describes how much the mean of variable Y is changed by a unit change of variable X. This parameter is called the *direct causal effect* of X on Y. Each variable can only influence its children directly, so if Y is not a child of X, the direct causal effect of X on Y is zero. Hence, from the values of all parents of Y, one can calculate the expected mean of variable Y, and the probability that Y has value y depending on the current values of its parents is given by a normal distribution centered at the mean.

Building a SEM requires much less data than building a non-linear model since one only needs a single parameter for each edge rather than probabilities for exponentially many combinations of variable values. This makes it feasible to gather enough data to create larger models consisting of more variables.

For instance, a 10% increase in the mother's diabetes risk could increase the diabetes risk of the child by 2%. Then the direct causal effect of *MR* on *D* would be $2\%/10\% = 0.2$. *MR* also affects *D* through the path *MR* $\rightarrow$ *MD* $\rightarrow$ *D*. Let us assume for a moment that *MD* is not a binary variable and the direct causal effect of *MR* on *MD* is 0.5 and the effect of *MD* on *D* is 0.3, then the observed correlation between *MR* and *D* is $0.2 + 0.5 \cdot 0.3 = 0.35$.

In linear models, it is possible to calculate causal effects that cannot be identified in non-linear models. For example, the equation $0.2 + 0.5 \cdot 0.3 = 0.35$ involves three direct causal effects on the left-hand side and one correlation on the right-hand side. Since all correlations are known from the observed data, the equation allows us to identify one of the direct causal effects, e.g., 0.2, whenever the other two directed causal effects are already identified.

The identification problem in SEMs asks one to identify as many parameters of the SEMs as possible, e.g., calculate as many direct causal effects as possible from observed correlations. The identification problem is one of the most important problems in the theory and practice of SEMs [Fis66]. In general cases, the problem remains unsolved, although many methods have been developed to identify specific cases.

The most popular methods to identify direct causal effects in SEMs are *instrumental variables* (resp. *conditional instrumental variables*) [BT84; AIR96; Imb14; Pea01]. Thereby, one searches for a variable Z, such that the direct causal effect of X on Y is the quotient of the correlation between Y and Z and the correlation between X and Z (resp. after conditioning on other covariates). There exists a simple graphical criterion to decide whether a given variable is a (resp. conditional) instrumental variable, essentially Z needs to be (resp. conditionally) independent of Y while being correlated with X. The algorithmic complexity of conditional instrumental variables has not been investigated, so far, and it has been unclear how to find conditional instrumental variables efficiently.

We will show that testing whether the criterion for conditional instrumental variables is satisfied is an NP-complete problem because it is hard to choose the set of covariates for conditioning. Nevertheless, we provide an efficient algorithm to find a variable satisfying the criterion if any conditional instrumental variable exists in the graph. For that, we define a specialization of the criterion which can be verified efficiently, and prove that if a variable

satisfies the general criterion, then there always exists at least one variable satisfying the specialized criterion.

In some instances of SEMs, it is not possible to identify causal effects using a single (conditional) instrumental variable, but multiple instrumental variables together can identify it. Brito et al. have formalized this as so-called *instrumental sets* [Bri04; Bri10; BP02] and have given a criterion to recognize instrumental sets. However, they have left the algorithmic aspects of instrumental sets open. We will analyze the complexity of their criterion and show that it is NP-complete to decide whether a set is an instrumental set, even though each variable in the set has to fulfill a criterion that is a special case of our specialized criterion for conditional instrumental variables. We then derive a simpler criterion from the instrumental set criterion, which yields efficient algorithms in certain cases.

1.1 Publications

The results of this thesis have lead to the following publications:

[ZLT14] Benito van der Zander, Maciej Liśkiewicz, and Johannes Textor. "Constructing Separators and Adjustment Sets in Ancestral Graphs". In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*. (**IBM Best Student Paper Award**). AUAI Press, 2014, pp. 907–916

[ZTL15] Benito van der Zander, Johannes Textor, and Maciej Liśkiewicz. "Efficiently Finding Conditional Instruments for Causal Inference". In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. 2015, pp. 3243–3249

[ZL16b] Benito van der Zander and Maciej Liśkiewicz. "Separators and Adjustment Sets in Markov Equivalent DAGs". In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. 2016, pp. 3315–3321

[ZL16a] Benito van der Zander and Maciej Liśkiewicz. "On Searching for Generalized Instrumental Variables." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2016, pp. 1214–1222

[Tex+16] Johannes Textor, Benito van der Zander, Mark S Gilthorpe, Maciej Liśkiewicz, and George TH Ellison. "Robust causal inference using directed acyclic graphs: the R package 'dagitty'". In: *International Journal of Epidemiology* 45.6 (2016), pp. 1887–1894

[ZLT19] Benito van der Zander, Maciej Liśkiewicz, and Johannes Textor. "Separators and Adjustment Sets in Causal Graphs: Complete Criteria and an Algorithmic Framework". In: *Artificial Intelligence* 270 (2019), pp. 1–40

[ZL19] Benito van der Zander and Maciej Liśkiewicz. "Finding Minimal d-separators in Linear Time and Applications". In: *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2019

1.2 Structure of this Thesis

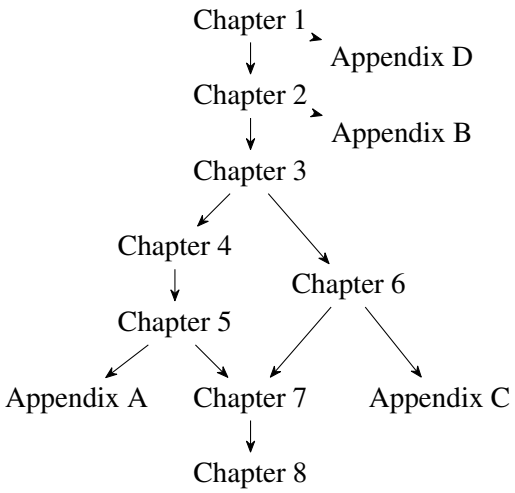


Figure 1.2: This thesis as a DAG representing the non-transitive dependencies between chapters.

Figure 1.2 shows the order in which the chapters of this thesis should be read.

The next chapter, Chapter 2, introduces basic notations for probability distributions and graphs, and formally defines paths, separation, various graphical models, and causal effects.

The following three chapters examine non-parametric models. Chapter 3 presents efficient algorithms for separation, especially for testing or finding separators, minimal separators, and minimum separators.

Chapter 4 presents efficient algorithms for covariate adjustment, especially for testing or finding adjustment sets, minimal adjustment sets, and minimum adjustment sets. For this goal, the algorithms of Chapter 3 are combined with new adjustment criteria that decide whether a separating set is an adjustment set.

Chapter 5 empirically evaluates some of the algorithms of Chapter 4 on random graphs and compares them with other identification methods.

Chapter 6 is the second part of this thesis and studies structural equation models. It introduces several definitions of instrumental variables and instrumental sets, and algorithms to test and find them, as well as NP-completeness results.

The thesis ends with a discussion in Chapter 7, the bibliography in Chapter 8, and the appendices.

2 Preliminaries: Causal Graphical Models

In this chapter, we recall the language of graphical models and present our notations. We also introduce the class of restricted chain graphs.

2.1 General Background and Notation

Probability distributions. We write random variables with upper-case letters like X, Y, Z, and their values with lower-case letters x, y, z. Sets of variables and their (ordered associated) values are written in boldface, e.g., $\mathbf{V} = \{X, Y, Z\}$ and $\mathbf{v} = (x, y, z)$.

We consider *discrete joint probability distributions* defined by their *probability mass function* $P(\mathbf{V} = \mathbf{v})$ and *continuous probability distributions* equivalently defined by their *probability density function* $P(\mathbf{V} = \mathbf{v})$. For $\mathbf{V} = \{V_1, \ldots, V_n\}$, we use the notation $P(\mathbf{v}) = P(\mathbf{V} = \mathbf{v}) = P(v_1, \ldots, v_n) = P(V_1 = v_1, \ldots, V_n = v_n)$. Most of our results will be written in the notation of discrete distributions, but they are also valid for continuous distributions.

probability density function

There are two ways to obtain a probability distribution on a subset of variables $\mathbf{V}' = \mathbf{V} \setminus \mathbf{W}$. First, *conditioning*, which fixes the value of the variables $\mathbf{W}$, i.e., $P(\mathbf{v}' \mid \mathbf{w})$. $P(\mathbf{v}' \mid \mathbf{w})$ is called the *conditional probability (distribution)* of $\mathbf{v}'$ given $\mathbf{w}$ and calculated with *Bayes' rule* as $P(\mathbf{v}' \mid \mathbf{w}) = \frac{P(\mathbf{v}', \mathbf{w})}{P(\mathbf{w})} = \frac{P(\mathbf{w} \mid \mathbf{v}')P(\mathbf{v}')}{P(\mathbf{w})}$. Secondly, *marginalization*, which sums over all possible values of $\mathbf{W}$, i.e., $P(\mathbf{v}') = \sum_{\mathbf{w}} P(\mathbf{v}, \mathbf{w}) = \sum_{\mathbf{w}} P(\mathbf{v} \mid \mathbf{w})P(\mathbf{w})$.

conditioning
conditional probability
Bayes' rule
marginalization

Two sets of variables $\mathbf{U}$ and $\mathbf{W}$ are *conditionally independent* if $P(\mathbf{u} \mid \mathbf{w}) = P(\mathbf{u})$, or equivalently, if $P(\mathbf{u}, \mathbf{w}) = P(\mathbf{u})P(\mathbf{w})$, for all values $\mathbf{u}$ and $\mathbf{w}$.

conditionally independent

Mixed graphs and paths. We consider *mixed graphs* $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ with nodes (vertices) $\mathbf{V}$ and directed ($A \to B$), undirected ($A - B$), and bidirected ($A \leftrightarrow B$) edges $\mathbf{E}$. The number of nodes is $n = |\mathbf{V}|$ and the number of edges is $m = |\mathbf{E}|$. The nodes generally correspond to random variables and the edges represent certain relationships between the variables.

mixed graphs

We also consider graphs $\mathcal{G} = (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E})$ where the nodes are partitioned into two sets, $\mathbf{O}$ of *observed* nodes (or *measured*, *endogenous* nodes) and $\mathbf{V} \setminus \mathbf{O}$ of *unobserved* (or *latent*, *exogenous* nodes). We may omit "$\mathbf{O} \subseteq$" if $\mathbf{O}$ is clear from the context.

observed
unobserved
latent

We only consider graphs that have at most one edge between any pair of nodes, although, in some examples and proofs, we abbreviate a node with two children $\leftarrow U \rightarrow$ as one bidirected edge, in which case there can be a directed and a bidirected edge between the same pair of nodes. Nodes linked by an edge are *adjacent* to (*neighbors* of) each other. The *degree* $\deg(\mathcal{G})$ of a graph is the number of neighbors the node with the highest number of neighbors has.

adjacent
neighbors
degree

A *walk* of length k is a node sequence $V_1, \ldots, V_{k+1}$ such that there exists an edge sequence

walk

start node *end node* *internal node* *path* *cycle* *subwalk* *subpath*

$E_1, E_2, \ldots, E_k$ for which every edge E_i connects V_i, V_{i+1}. Then V_1 is called the *start node* and V_{k+1} the *end node* of the walk. Every other node $V_2, \ldots, V_k$ is an *internal node*. A *path* is a walk in which no node occurs more than once. A *cycle* is a walk whose start and end node are the same node. A subsequence $V_i, \ldots, V_j$ of a walk (path) is a *subwalk* (*subpath*). By $V_1 \overset{*}{\sim} V_{k+1}$ we denote a walk between V_1 and V_{k+1} with $k \geq 0$ and by $V_1 \overset{+}{\sim} V_{k+1}$ a walk with $k > 0$.

Depending on the context, we will represent a walk as a sequence of nodes or a sequence of incident edges. Neither the nodes nor edges need to exist in the graph unless we explicitly talk about existing walks or say something like "there is a path". For example, when a node V is given and we say "there exists no path $\rightarrow V \leftarrow$", it means that there exist no distinct nodes V_1, V_2 such that there are directed edges $V_1 \rightarrow V$ and $V_2 \rightarrow V$ in a graph that you need to infer from the context, usually the graph containing V. If we say "there exists no walk $\rightarrow V \leftarrow$", nodes V_1 and V_2 do not need to be distinct.

For a path π, we denote by $\pi[V_i \overset{*}{\sim} V_{i'}]$ the subpath of π consisting of the nodes between V_i and $V_{i'}$. For walks, a subwalk, which does not need to be unique. We concatenate node and edge sequences by writing them behind each other, e.g., $V_1\pi[V_2 \overset{*}{\sim} V_{k+1}]$ or $\pi[V_1 \overset{*}{\sim} V_i]\pi[V_{i+1} \overset{*}{\sim} V_{k+1}]$ form again the path π. For the sake of simplicity, we write often the last concatenation as $\pi[V_1 \overset{*}{\sim} V_i]\pi[V_i \overset{*}{\sim} V_{k+1}]$ assuming a duplicated end node V_i in the resulting path is implicitly removed. The concatenation of two paths might not be a path, but it is always a walk (see Lemma 3.11).

$\mathbf{X}$*-proper* *proper*

Given a node set $\mathbf{X}$ and a node set $\mathbf{Y}$, a walk from $X \in \mathbf{X}$ to $Y \in \mathbf{Y}$ is called $\mathbf{X}$*-proper* if only its start node is in $\mathbf{X}$. If $\mathbf{X}$ is clear from the context, it is omitted and we just say the walk is *proper*.

We will apply set operators to graphs $\mathcal{G}$ and walks w as if they were sets, and the result can be either a set, a graph, or a walk. For example, depending on context $\mathcal{G} \cap \mathbf{X}$ (or $w \cap \mathbf{X}$) can mean the set of all nodes in $\mathcal{G}$ (or w) intersected by $\mathbf{X}$ or $\mathcal{G}$ (or w) with all nodes outside of $\mathbf{X}$ removed. Or $\mathcal{G} \setminus (X \rightarrow Y)$ means a graph without the edge $X \rightarrow Y$.

directed *causal* *non-causal* *ancestor* *descendant*

Ancestry. A walk of the form $V_1 \rightarrow \ldots \rightarrow V_k$ is *directed* (*causal*). A walk that is not causal is *non-causal* or *biasing*. If there is a directed walk from U to V, then U is called an *ancestor* of V and V a *descendant* of U. We write directed walks from U to V as $U \overset{*}{\rightarrow} V$ or in reverse $V \overset{*}{\leftarrow} U$. If U and V need to be distinct, we also use $U \overset{+}{\rightarrow} V$ or $V \overset{+}{\leftarrow} U$.

possibly directed

A walk is *possibly directed* if it were directed after replacing all edges — by $\rightarrow$. We write possibly undirected walks from U to V as $U \overset{*}{\rightarrow} V$, $U \overset{+}{\rightarrow} V$, $V \overset{*}{\leftarrow} U$, or $V \overset{+}{\leftarrow} U$. The plus sign marks again distinct nodes. A possibly directed path can consist of only undirected — edges. If a possibly directed path contains at least one directed edge, it is called *semi-directed*. If there is a possibly directed path from U to V, then U is called a *possible ancestor* of V and V is a *possible descendant* of U. Some authors call possible ancestors *anteriors*.

semi-directed *possible ancestor* *possible descendant*

All ancestors of V are possible ancestors of V and all descendants are possible descendants. Every node is its own ancestor, descendant, possible ancestor, and possible descendant, connected to itself by a walk of length 0. For a node set $\mathbf{X}$, the set of all of its ancestors is written as $An(\mathbf{X})$. The descendant, possible ancestor, and possible descendant sets $De(\mathbf{X})$, $pAn(\mathbf{X})$, and $pDe(\mathbf{X})$ are analogously defined.

parent *child*

If there is an edge $A \rightarrow B$, A is a *parent* of B and B a *child* of A. Also, we denote by $Pa(\mathbf{X})$, ($Ch(\mathbf{X})$, $Ne(\mathbf{X})$), the set of parents (children, neighbors) of $\mathbf{X}$.

Subgraphs. Give a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, a graph $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$ is a *subgraph* (or *configuration*) of $\mathcal{G}$ if $\mathbf{V}' \subseteq \mathbf{V}$ and $\mathbf{E}' \subseteq \mathbf{E}$. The *induced subgraph* for a given node set $\mathbf{V}'$ is the graph $\mathcal{G}_{\mathbf{V}'} = (\mathbf{V}', \mathbf{E}')$ that only contains the edges $\mathbf{E}' = \mathbf{E} \cap (\mathbf{V}' \times \mathbf{V}')$ that are adjacent to nodes in $\mathbf{V}'$. *subgraph* *induced subgraph*

The *skeleton* of $\mathcal{G}$ is a graph with the same nodes in which every edge is replaced by an undirected edge. A *connected component* is a subgraph in which every pair of nodes is connected by a path. A subgraph containing only a single node is also a connected component. A connected component is a *bidirectionally connected component* if for every pair of nodes there exists a connecting path that contains only bidirected edges. *skeleton* *connected component* *bidirectionally connected component*

For any subset $\mathbf{A}$ and $\mathbf{B}$ of $\mathbf{V}$, by $\mathbf{A} \to \mathbf{B}$ we denote the set of all edges $A \to B$ in $\mathbf{E}$, such that $A \in \mathbf{A}$ and $B \in \mathbf{B}$; the sets $\mathbf{A} \leftarrow \mathbf{B}$, $\mathbf{A} \leftrightarrow \mathbf{B}$, and $\mathbf{A} - \mathbf{B}$ are defined analogously. Using this notation, the graph obtained from a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ by removing all edges entering a certain node set $\mathbf{X}$ is written as $\mathcal{G}_{\overline{\mathbf{X}}} = (\mathbf{V}, \mathbf{E} \setminus ((\mathbf{V} \to \mathbf{X}) \cup (\mathbf{V} \leftrightarrow \mathbf{X})))$. The removal of all edges leaving $\mathbf{X}$ is written as $\mathcal{G}_{\underline{\mathbf{X}}} = (\mathbf{V}, \mathbf{E} \setminus ((\mathbf{X} \to \mathbf{V}) \cup (\mathbf{X} - \mathbf{V})))$. The application of both these operations $(\mathcal{G}_{\overline{\mathbf{X}}})_{\underline{\mathbf{X}'}}$ is abbreviated as $\mathcal{G}_{\overline{\mathbf{X}}\underline{\mathbf{X}'}}$. Descendants and ancestors in these graphs are written as set subscripts at the corresponding function without specifying the graph, e.g., $De_{\overline{\mathbf{X}}}$ or $An_{\underline{\mathbf{X}}}$.

A *v-structure* are three nodes A, B, and C that induce a subgraph $A \to B \leftarrow C$, i.e., A and C are unconnected parents of B. Some authors call v-structures immoralities. *v-structure*

A graph is *complete*[1] if there is an edge between any pair of nodes. A graph is *chordal* if every induced subgraph with more than three nodes is not a cycle, i.e., every cycle with more than three nodes contains two nodes that are connected by an edge that does not belong to the cycle. *complete* *chordal*

Separation. A node V on a walk w is called a *collider* if two arrowheads of w meet at V, e.g., $U \to V \leftarrow W$ or $U \leftrightarrow V \leftarrow W$. Walks containing less than two edges cannot contain a collider. An internal node that is not a collider is called a *non-collider*. If a node occurs multiple times on a walk, it can occur as a collider and as a non-collider on the same walk. A *fork* is a node with adjacent edges $\leftarrow\!\rightarrow$, the opposite of a collider. Two nodes are *collider connected* if there is a path between them on which all internal nodes are colliders. Adjacent vertices are collider connected. *collider* *non-collider* *fork* *collider connected*

A walk is *active* given $\mathbf{Z}$ if every non-collider is not in $\mathbf{Z}$ and every collider is in $An(\mathbf{Z})$. A walk is *blocked* by $\mathbf{Z}$ if is not active given $\mathbf{Z}$. For a walk active given (blocked by) the empty set, we just say the walk is active (blocked). Active walks are also called d-connecting, m-connecting, or just connecting. This is an unusual definition[2], but in Section 3.1.2 we will see that standard definitions are equivalent. *active* *blocked*

Two nodes X and Y are *d-connected* by a set $\mathbf{Z}$ if there is a path $X \overset{*}{\sim} Y$ active given $\mathbf{Z}$. Two node sets $\mathbf{X}, \mathbf{Y}$ are *d-connected* by a set $\mathbf{Z}$ if there is any pair $X \in \mathbf{X}$, $Y \in \mathbf{Y}$ such that X and Y are d-connected by $\mathbf{Z}$. Two node sets $\mathbf{X}, \mathbf{Y}$ not d-connected by $\mathbf{Z}$ are *d-separated* by $\mathbf{Z}$ and then $\mathbf{Z}$ *d-separates* them. Two node sets $\mathbf{X}, \mathbf{Y}$ are *d-separable* if there exists any set $\mathbf{Z}$ that d-separates them. We will only consider disjoint sets $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. For nodes d-connected (d-separated) by the empty set, we will just say they are d-connected (d-separated). We will use *m-connected* or just *connected* as a synonym for d-connected, and *m-separated* or just *separated* as a synonym for d-separated, likewise for d-separates and d-separable. A set $\mathbf{Z}$ that *d-connected* *d-separated* *m-connected* *connected* *m-separated* *separated*

[1] Not to be confused with a *complete criterion*, which is a criterion formulating a necessary condition.

[2] Most authors use different terminology for different classes of graphs, and only use colliders "in $An(\mathbf{Z})$" for

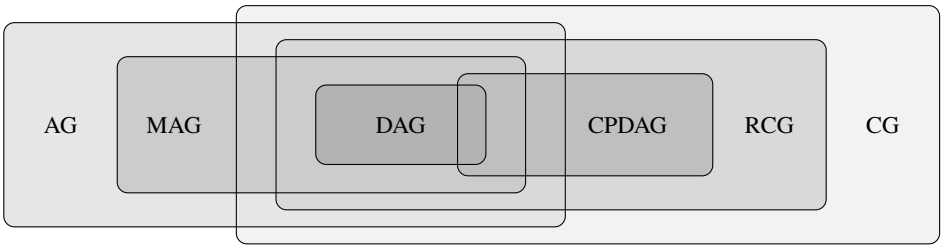


Figure 2.1: The inclusion relationships between the relevant classes of causal graphs. The inclusions are proper, i.e., DAG $\subsetneq$ RCG $\subsetneq$ CG, CPDAG $\subsetneq$ RCG $\subsetneq$ CG, and DAG $\subsetneq$ MAG $\subsetneq$ AG, which means for example every DAG is an RCG but not necessarily a CPDAG.

separator separates $\mathbf{X}$ and $\mathbf{Y}$ is a *separator* (relative to $(\mathbf{X}, \mathbf{Y})$).

$(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})_{\mathcal{G}}$ shall denote that $\mathbf{X}$ and $\mathbf{Y}$ are d-separated given $\mathbf{Z}$ in the graph $\mathcal{G}$. $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})_{\mathcal{G}}$ means $\mathbf{X}$ and $\mathbf{Y}$ are d-connected given $\mathbf{Z}$ in the graph $\mathcal{G}$. The graph subscript is omitted if the graph is clear from the context.

2.2 Classes of Graphical Models

In this section, we introduce the most commonly used classes of graphical models. The models differ in the kind of allowed edges, their definition of separation, and their statistical interpretation. For each model, we give a quick motivation and definition. Figure 2.1 shows a summary of the inclusion relationships between important classes. We only define one novel class – the RCGs – all other classes are well known.

Directed acyclic graphs (DAGs), Bayesian network. A graph that only contains directed *directed graph* edges is a *directed graph*, and a *DAG* is a directed graph that contains no directed cycle. A *DAG* *Bayesian network* is a DAG $\mathcal{G}$ representing a set of random variables $\mathbf{V} = \{V_1, \ldots, V_n\}$ as nodes and the dependencies between the variables as edges, such that each variable depends (only) on its parents. The random variables can either be discrete or continuous variables since all definitions are valid for both cases.

compatible A joint probability distribution P over the variables $\mathbf{V}$ is *compatible* with $\mathcal{G}$ if P can be factorized as

$$P(\mathbf{v}) = \prod_{i=1}^{n} P(v_i \mid pa_i), \tag{2.1}$$

where $\mathbf{v}$ (resp. v_i and pa_i) denotes a particular realization of variables $\mathbf{V}$ (resp. V_i and $Pa(V_i)$), and $P(v_i \mid pa_i)$ is a conditional probability distribution of V_i given $Pa(V_i)$.

Let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be three disjoint sets of random variables. If $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in the DAG $\mathcal{G}$, $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ in every probability distribution compatible with $\mathcal{G}$. If $\mathbf{X}$ and $\mathbf{Y}$ are not d-separated by $\mathbf{Z}$ in the DAG $\mathcal{G}$, they are conditionally dependent in at least one probability distribution compatible with $\mathcal{G}$ [3]. Thus, by studying the properties of the DAG one can make statements that hold for all compatible probability distributions without ever studying the probability distributions themselves.

paths, and "in $\mathbf{Z}$" for walks.

[3] one could also say: conditionally dependent in almost all probability distributions compatible with $\mathcal{G}$ [Pea09].

Although the edges in the Bayesian network are directed, the direction of each edge carries no deeper meaning besides conditional independence. In a *causal DAG*, an edge $V_i \to V_j$ is taken to represent a direct causal effect of V_i on V_j and an intervention on variable V_i yields different results than an intervention on variable V_j (see Subsection 2.3)[Pea09]. *causal DAG*

Completed partially directed acyclic graphs (CPDAGs). A given probability distribution can be compatible with multiple Bayesian networks. A common example is the three DAGs $\mathcal{G}_{\to} : A \to B \to C$, $\mathcal{G}_{\leftrightarrow} : A \leftarrow B \to C$, and $\mathcal{G}_{\leftarrow} : A \leftarrow B \leftarrow C$. Any probability distribution $P(A, B, C)$ that is compatible with one of these graphs is compatible with the two other graphs as $P(A)P(B|A)P(C|B) = P(A|B)P(B)P(C|B) = P(A|B)P(B|C)P(C)$ follows from Bayes' rule. Nevertheless, P is generally not compatible with a DAG $\mathcal{G}_{\times} : A \to B \leftarrow C$.

Two graphs that encode the same set of probability distributions are called *Markov equivalent*. Two DAGs are Markov equivalent iff they imply the same conditional independences; and they imply the same conditional independences iff they have the same skeleton and the same v-structures[VP90]. For a DAG $\mathcal{D}$, the class of Markov equivalent graphs to $\mathcal{D}$, denoted as $[\mathcal{D}]$, is defined as $[\mathcal{D}] = \{\mathcal{D}' \mid \mathcal{D}' \text{ is Markov equivalent to } \mathcal{D}\}$. For the above example graphs, we have $[\mathcal{G}_{\to}] = \{\mathcal{G}_{\to}, \mathcal{G}_{\leftrightarrow}, \mathcal{G}_{\leftarrow}\}$ and $[\mathcal{G}_{\times}] = \{\mathcal{G}_{\times}\}$. $\mathcal{G}_{\to}$ and $\mathcal{G}_{\times}$ are not Markov equivalent since $\mathcal{G}_{\to}$ has no v-structures and $\mathcal{G}_{\times}$ has one v-structure. *Markov equivalent*

A *completed partially directed acyclic graph* (*CPDAG* or *essential graph*) $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a mixed graph with directed and undirected edges that represents an entire Markov equivalence class $[\mathcal{D}]$. The edge $A \to B$ is in $\mathbf{E}$ if $A \to B$ belongs to every $\mathcal{D}' \in [\mathcal{D}]$ and $A - B$ is in $\mathbf{E}$ if there exist $D', D'' \in [\mathcal{D}]$ such that $A \to B$ is an edge of $\mathcal{D}'$ and $A \leftarrow B$ an edge of $\mathcal{D}''$ [Chi95; AMP+97]. Any replacement of all undirected edges in a CPDAG by directed edges which does not introduce new v-structures yields a DAG in the represented Markov equivalence class, and any DAG in the equivalence class can be obtained this way. *completed partially directed acyclic graph* *CPDAG*

Chain graphs (CGs), partial directed acyclic graphs. *Chain graphs* are mixed graphs that contain no semi-directed cycle [LW89; Fry90; AMP97]. *chain graph* *CG*

They generalize CPDAGs by representing a subset of the Markov equivalence class. For example, a chain graph $A \leftarrow B - C$ would represent DAGs $A \leftarrow B \to C$ and $A \leftarrow B \leftarrow C$. Another example is given in Figure 2.2. Formally, given a CG $\mathcal{G}$ a DAG $\mathcal{D}$ is a *consistent DAG extension* of $\mathcal{G}$ if and only if (1) $\mathcal{G}$ and $\mathcal{D}$ have the same skeletons, (2) if $A \to B$ is in $\mathcal{G}$, then $A \to B$ is in $\mathcal{D}$, and (3) $\mathcal{G}$ and $\mathcal{D}$ have the same v-structures. We refer to all consistent DAG extensions of a mixed graph $\mathcal{G}$ as $CE(\mathcal{G})$. Notice that if a CG $\mathcal{G}$ is a CPDAG then $CE(\mathcal{G}) = [\mathcal{D}]$ for some DAG $\mathcal{D}$, and if $\mathcal{G}$ is a DAG then $CE(\mathcal{G}) = \{\mathcal{G}\}$. *consistent DAG extension*

A chain graph can be partitioned into a set of *chain components*, each chain component consisting of all nodes that are connected to each other by undirected paths. A node not adjacent to any undirected edge would be a chain component by itself. *chain components*

[AMP+97] show that a CG $\mathcal{G}$ is a CPDAG if and only if (1) every chain component of $\mathcal{G}$ is chordal, (2) no induced subgraph of $\mathcal{G}$ is $A \to B - C$, and (3) every directed edge $A \to B \in \mathcal{G}$ is strongly protected. An edge is *strongly protected* if it occurs in one of the four induced subgraphs shown in Figure 2.3. *strongly protected*

D-separation in a chain graph $\mathcal{G}$ is defined through d-separation of the represented DAGs $\mathcal{D} \in CE(\mathcal{G})$. A path π in $\mathcal{G}$ corresponds to a path $\pi_{\mathcal{D}}$ that has the same nodes in $\mathcal{D}$. Every collider on π is a collider on $\pi_{\mathcal{D}}$, but a non-collider on π can either be a non-collider or a collider on $\pi_{\mathcal{D}}$ when there are undirected edges. Using the notation of [Zha08], a non-collider on π is

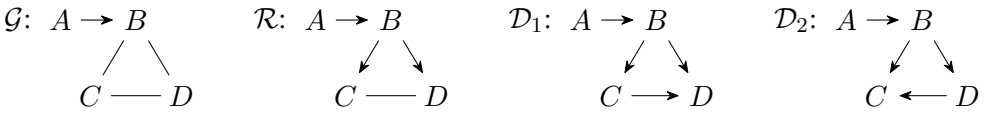


Figure 2.2: An example chain graph $\mathcal{G}$ that is neither a CPDAG nor an RCG. It represents the DAGs $CE(\mathcal{G}) = \{\mathcal{D}_1, \mathcal{D}_2\}$, so $\mathcal{R}$ is the RCG equivalent to $\mathcal{G}$.

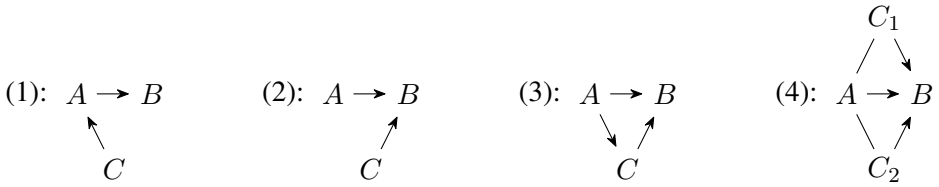


Figure 2.3: All induced subgraphs whose occurrence makes an edge $A \to B$ strongly protected [AMP+97]. Arbitrary nodes can be chosen as C, C_1 and C_2, but all nodes need to be distinct.

definite non-collider called a ***definite non-collider*** if it is also a non-collider on $\pi_{\mathcal{D}}$ for every DAG $\mathcal{D} \in CE(\mathcal{G})$. π is
definite status a ***definite status*** path if every internal node is either a collider or a definite non-collider. Two nodes X and Y are d-connected by a set $\mathbf{Z}$ in the CG if there is a definite status path $X \overset{*}{\sim} Y$ active given $\mathbf{Z}$. Definite status paths can be characterized without involving the represented DAGs (see Section 3.1.1).

restricted chain graph **Restricted chain graph (RCGs).** A chain graph $\mathcal{G}$ is a *restricted chain graph* (*RCG*) if
RCG and only if (1) every chain component of $\mathcal{G}$ is chordal and (2) no induced subgraph of $\mathcal{G}$ is $A \to B - C$. Hence, RCGs are between CPDAGs and CGs in terms of generality. Any CPDAG, DAG, or undirected chordal graph is an RCG. We will later show that any CG $\mathcal{G}$ that represents at least one DAG can be transformed to an equivalent RCG $\mathcal{R}$ that satisfies $CE(\mathcal{R}) = CE(\mathcal{G})$ (see Proposition 3.39).

RCGs serve three purposes. First, they allow us to properly reason about some subgraphs of CPDAGs which are not CPDAGs themselves. Secondly, they capture the algorithmically relevant properties of CPDAGs because, for most algorithms, it does not matter if the edges are strongly protected or not. Thirdly, such graphs arise naturally when learning a causal structure from statistical data with background knowledge. Structure learning algorithms proposed by [VP90; VP90; VP92] construct from a given list of conditional independence statements a CPDAG. The learning algorithm can be extended to include required and exclude forbidden directed edges given as background knowledge [Mee95], in which case the resulting graph might not be a CPDAG anymore.

Ancestral graphs (AGs). By marginalizing or conditioning a probability distribution compatible to a DAG, one can obtain a probability distribution on fewer variables that is not compatible to any DAG [RS02].

Ancestral graphs were designed to represent such marginalized and conditioned distributions. They generalize DAGs by allowing bidirected and undirected edges, whereby a bidirected edge

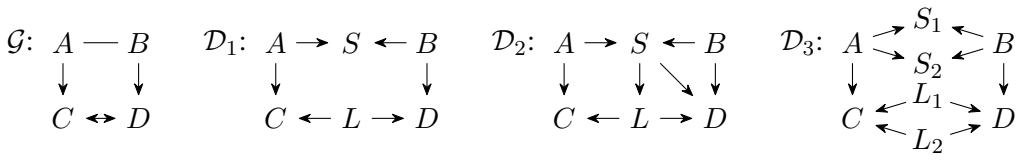


Figure 2.4: An exemplary ancestral graph $\mathcal{G}$. DAGs $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ are represented by $\mathcal{G}$ via $\mathcal{G} = \mathcal{D}_1[^{S}_{L} = \mathcal{D}_2[^{S}_{L} = \mathcal{D}_3[^{\{S_1,S_2\}}_{\{L_1,L_2\}}$. $\mathcal{D}_1$ is also the canonical DAG $\mathcal{C}(\mathcal{G})$ of $\mathcal{G}$. $\mathcal{G}$ is a maximal, ancestral graph, but we will not refer to it as MAG, since it contains an undirected edge.

$\leftrightarrow$ stands for a latent node $\leftarrow L \rightarrow$ and an undirected edge — stands for a biased node $\rightarrow S \leftarrow$. AGs are closed under marginalization and conditioning, so that marginalizing and conditioning the distribution of an AG always leads to a distribution that can be represented by another AG [RS02].

AGs can be characterized as follows: A mixed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is an *ancestral graph* (*AG*) if the following two conditions hold [RS02]: (1) For each edge $A \leftarrow B$ or $A \leftrightarrow B$, A is not a possible ancestor of B. (2) For each edge A — B, there are no edges $A \leftarrow C$, $A \leftrightarrow C$, $B \leftarrow C$, or $B \leftrightarrow C$. From these conditions, it follows that there can be at most one edge between two nodes in an AG [RS02]. Syntactically, all AGs containing only directed edges are DAGs and all DAGs are AGs [Zha08]. *ancestral graph* *AG*

Separation in ancestral graphs is usually called m-separation rather than d-separation [RS02], however, for DAGs m-separation and d-separation are identical, so we have defined d-separation as m-separation in Section 2.1 and we will use the term d-separation or even just separation for all graph classes when giving statements that are valid for different classes of graphs.

Maximal ancestral graph (MAGs). An AG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a *maximal ancestral graph* (*MAG*) if every non-adjacent pair of nodes U, V can be m-separated by some $\mathbf{Z} \subseteq \mathbf{V} \setminus \{U, V\}$. Every AG $\mathcal{G}$ can be made maximal by adding bidirected edges between node pairs that cannot be m-separated, which preserves all m-separation relationships in the graph [RS02]. *maximal ancestral graph* *MAG*

A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is represented by a MAG $\mathcal{M} = \mathcal{G}[^{\mathbf{S}}_{\mathbf{L}}$ with nodes $\mathbf{V} \setminus (\mathbf{S} \cup \mathbf{L})$ for sets $\mathbf{S}, \mathbf{L} \subseteq \mathbf{V}$, whereby $\mathcal{M}$ has an edge between a pair of nodes U, V if U, V cannot be d-separated in $\mathcal{G}$ by any $\mathbf{Z}$ with $\mathbf{S} \subseteq \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{L}$. That edge has an arrowhead at node V if $V \notin An(U \cup \mathbf{S})$. Hence, every MAG represents an infinite set of underlying DAGs that all share the same ancestral relationships. Thus, one can use MAGs to identify causal effects without measuring or knowing all occurring variables.

The *canonical DAG* $\mathcal{C}(\mathcal{M})$ of a MAG $\mathcal{M}$ is the DAG obtained from $\mathcal{M}$ by replacing every $\leftrightarrow$ edge with $\leftarrow L \rightarrow$ and every $-$ edge with $\rightarrow S \leftarrow$ with new nodes L or S which form sets $\mathbf{L}$, $\mathbf{S}$. Clearly $\mathcal{C}(\mathcal{M})[^{\mathbf{S}}_{\mathbf{L}} = \mathcal{M}$ (see [RS02]). An example is shown in Figure 2.4. *canonical DAG*

Like [Zha08] we will only consider MAGs that contain no undirected edges, i.e., $\mathbf{S} = \emptyset$ whenever we construct a MAG $\mathcal{G}^{\emptyset}_{\mathbf{L}} = \mathcal{G}^{\mathbf{S}}_{\mathbf{L}}$ from a DAG $\mathcal{G}$. Ancestral graphs without undirected edges can also be characterized as the class of the mixed graphs that do not contain cycles of the form $V_1 \overset{+}{\to} V_k \to V_1$ or $V_1 \leftrightarrow V_2 \overset{*}{\to} V_k \to V_1$.

Semi-Markovian models. A *semi-Markovian model* is a DAG with additional bidirected edges that represent latent variables [Pea09]. Bidirectional edges can occur between nodes also connected by a directed edge, and each bidirected edge represents exactly one latent *semi-Markovian model*

variable. The statistical interpretation of a semi-Markovian model is the same as the statistical interpretation of the DAG one obtains by replacing each bidirected edge $V_i \leftrightarrow V_j$ with a subgraph $V_i \leftarrow L_{ij} \rightarrow V_j$. Thus, semi-Markovian models can be considered as another notation of writing DAGs with unobserved nodes rather than as a separate class[4].

structural equation model *SEM*

Structural equation models (SEMs). A *structural equation model* (*SEM*) is a parametric semi-Markovian model which is more specific than the graphical models defined above. Each random variable V_j with domain $\mathbb{R}$ follows a linear equation $v_j = \sum_i \lambda_{ji} v_i + \epsilon_j$, where λ_{ji} is a parameter associated with edge $V_i \rightarrow V_j$ and ϵ_j is a Gaussian random variable. Thus, the term $P_j(v_j \mid pa_j)$ in the factorization of $P(\mathbf{v})$ of a DAG becomes the probability that V_j takes some value v_j given by that linear equation.

This class is properly defined and discussed further in Chapter 6.

Related Classes. There exist many other classes of graphical models that we do not investigate in this thesis, like undirected graphs, cyclic graphs, partial ancestral graphs (PAGs), or maximal PDAGs. Appendix B gives an overview about these classes.

2.3 Do-operator and Causal Effects

intervention

Causal models provide a way to formally model causal effects by describing how a probability distribution $P(\mathbf{v})$ is changed by an intervention on variables $\mathbf{X}$. Such an *intervention* forces the variables $\mathbf{X}$ to have values $\mathbf{x}$, regardless of all other influences which might affect the variables $\mathbf{X}$, just like one could change the values of $\mathbf{X}$ in a real-life experiment to see how the change will affect all other variables. The *post-intervention distribution* is denoted as $P(\mathbf{V} = \mathbf{v} \mid do(\mathbf{X} = \mathbf{x}))$ or $P(\mathbf{v} \mid do(\mathbf{x}))$.

Given a pre-intervention distribution $P(\mathbf{v})$ that is compatible to a DAG $\mathcal{G}$, the post-intervention distribution $P(\mathbf{v} \mid do(\mathbf{x}))$ is compatible to the DAG $\mathcal{G}_{\overline{\mathbf{X}}}$, in which all edges into $\mathbf{X}$ have been removed and $\mathbf{X}$ is disconnected from its parents.

$P(\mathbf{v})$ can be factorized according to equation 2.1, so $P(\mathbf{v} \mid do(\mathbf{x}))$ can be calculating by removing all factors for $\mathbf{X}$:

$$P(\mathbf{v} \mid do(\mathbf{x})) = \begin{cases} \prod_{V_i \in \mathbf{V} \setminus \mathbf{X}} P(v_i \mid pa_i) & \text{for } \mathbf{v} \text{ consistent with } \mathbf{x}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

where $\mathbf{v}$ is consistent with $\mathbf{x}$ if the values for $\mathbf{X}$ among $\mathbf{v}$ are the same as $\mathbf{x}$. For an example of a not consistent $\mathbf{v}$, consider $P(\mathbf{v} \mid do(x = 1))$, which is necessarily zero if $\mathbf{v}$ contains $x = 2$.

total causal effect *direct causal effect*

The *(total) causal effect* of $\mathbf{X}$ on $\mathbf{Y}$ is $P(\mathbf{y} \mid do(\mathbf{x}))$, i.e., the effect of an intervention on $\mathbf{X}$ on some variables $\mathbf{Y}$. If $\mathbf{X}$ and $\mathbf{Y}$ are singletons and $\mathbf{Y}$ is a child of $\mathbf{X}$, then $P(\mathbf{y} \mid do(\mathbf{x}))$ is also a *direct causal effect*. When all variables in the graph are observed, the causal effect $P(\mathbf{y} \mid do(\mathbf{x}))$ can be calculated using the above factorization.

identifiable

However, when some variables are unobserved, the question of whether $P(\mathbf{y} \mid do(\mathbf{x}))$ is *identifiable*, i.e., whether it can be computed from a pre-intervention distribution $P(\mathbf{O} = \mathbf{o})$ independently of the unknown quantities involving the unobserved variables $\mathbf{V} \setminus \mathbf{O}$, becomes

[4] [Pea09] also calls all DAGs that have unobserved variables semi-Markovian models regardless if they contain bidirected edges or not. He calls DAGs that have neither bidirected edges nor unobserved variables Markovian models.

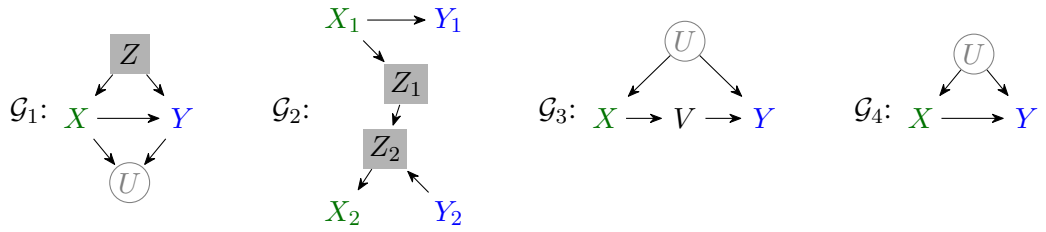


Figure 2.5: In $\mathcal{G}_1$, the causal effect of $\mathbf{X} = \{X\}$ on $\mathbf{Y} = \{Y\}$ can be identified with the parents as an adjustment set $\mathbf{Z} = \{Z\}$. In $\mathcal{G}_2$, the causal effect of $\mathbf{X} = \{X_1, X_2\}$ on $\mathbf{Y} = \{Y_1, Y_2\}$ can be identified with the adjustment set $\mathbf{Z} = \{Z_1, Z_2\}$, which does not satisfy the back-door criterion. In $\mathcal{G}_3$ and $\mathcal{G}_4$, the causal effect of $\mathbf{X} = \{X\}$ on $\mathbf{Y} = \{Y\}$ can not be identified by any (observed) adjustment set due to the unobserved variable U. In $\mathcal{G}_3$, the causal effect can be identified using the do-calculus [Pea09, Section 3.3], while in $\mathcal{G}_4$ the causal effect cannot be identified at all.

much more complex. For a formal definition of identifiability, see [Pea09, Section 3.2]. Figure 2.5 shows three DAGs $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$ for which the causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable, and a DAG $\mathcal{G}_4$ for which the effect is not identifiable.

If at least the parents $\mathbf{Z} = Pa(X)$ of a variable X are observed besides $X \cup \mathbf{Y}$ and are disjoint from $\mathbf{Y}$, the causal effect of X on variables $\mathbf{Y} \subseteq \mathbf{V} \setminus (X \cup \mathbf{Z})$ can be calculated by *adjusting* for the parents [Pea09, Section 3.2]: *adjusting*

$$P(\mathbf{y} \mid do(x)) = \sum_{\mathbf{z}} P(\mathbf{y} \mid x, \mathbf{z}) P(\mathbf{z}). \tag{2.3}$$

In this expression, the parents $\mathbf{Z}$ are used as a so-called *adjustment set*. If the parents are not observed, they cannot be used as an adjustment set. But there are other sets $\mathbf{Z}$ besides the parents that can be used as adjustment sets. The most popular criterion that characterizes possible adjustment sets is Pearl's back-door criterion: *adjustment set*

Definition 2.1 (Back-door criterion (BC) [Pea93; Pea09])**.** *A set of variables $\mathbf{Z}$ satisfies the* back-door criterion *relative to an ordered pair of variables (X, Y) in a DAG $\mathcal{G}$ if:* *back-door criterion*

(a) $\mathbf{Z} \subseteq \mathbf{V} \setminus De(X)$*, and*

(b) $\mathbf{Z}$ *blocks every path between X and Y that contains an arrow into X.*

Similarly, if $\mathbf{X}$ and $\mathbf{Y}$ are two disjoint subsets of nodes in $\mathcal{G}$, then $\mathbf{Z}$ is said to satisfy the back-door criterion relative to $(\mathbf{X}, \mathbf{Y})$ if it satisfies the back-door criterion relative to any pair (X, Y) such that $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$.

In order to use the adjustment set for identification, all variables in $\mathbf{Z}$ need to be observed. Condition (b) can also be understood as "$\mathbf{X}$ and $\mathbf{Y}$ need to be d-separated by $\mathbf{Z}$ in $\mathcal{G}_{\underline{\mathbf{X}}}$".

However, the back-door criterion is not complete, i.e., there are sets that are adjustment sets, but do not satisfy the back-door criterion. For example, $\mathbf{Z} = \{Z_1, Z_2\}$ in $\mathcal{G}_2$ of Figure 2.5 is an adjustment set that does not satisfy condition (a), and no other adjustment set exist in $\mathcal{G}_3$ for $\mathbf{X}$ and $\mathbf{Y}$. We will discuss this further in Chapter 4.

There also exist causal effects that cannot be identified by any adjustment set but can be identified with other approaches. Pearl has discovered the three rules of the *do-calculus*, which state equivalences between probability distributions:

do-calculus

Theorem 2.2 (Do-calculus [Pea09])**.** *Given a DAG $\mathcal{G}$ and disjoint sets $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, and $\mathbf{W}$, the following rules are valid for all probability distributions P compatible with $\mathcal{G}$:*

Rule 1. Insertion/deletion of observations

$$P(\mathbf{y} \mid do(\mathbf{x}), \mathbf{z}, \mathbf{w}) = P(\mathbf{y} \mid do(\mathbf{x}), \mathbf{w}) \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z} \mid \mathbf{X}, \mathbf{W}) \text{ in } \mathcal{G}_{\overline{\mathbf{X}}}.$$

Rule 2. Exchange of actions/observations

$$P(\mathbf{y} \mid do(\mathbf{x}), do(\mathbf{z}), \mathbf{w}) = P(\mathbf{y} \mid do(\mathbf{x}), \mathbf{z}, \mathbf{w}) \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z} \mid \mathbf{X}, \mathbf{W}) \text{ in } \mathcal{G}_{\overline{\mathbf{X}}\underline{\mathbf{Z}}}.$$

Rule 3. Insertion/deletion of actions

$$P(\mathbf{y} \mid do(\mathbf{x}), do(\mathbf{z}), \mathbf{w}) = P(\mathbf{y} \mid do(\mathbf{x}), \mathbf{w}) \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z} \mid \mathbf{X}, \mathbf{W}) \text{ in } \mathcal{G}_{\overline{\mathbf{X}}\,\overline{\mathbf{Z(W)}}},$$
$$\text{where } \mathbf{Z}(\mathbf{W}) \text{ is short for } \mathbf{Z} \setminus An_{\overline{\mathbf{X}}}(\mathbf{W}).$$

The do-calculus is sound and complete, so if a causal effect $P(\mathbf{y} \mid do(\mathbf{x}))$ is identifiable, one can always derive a valid, equal expression involving only terms of $P(\mathbf{o})$ by applying the rules of the do-calculus [HV06]. The IDC-algorithm can calculate such a derivation in polynomial time [SP06b; SP06a]. However, the IDC-algorithm can be slow (see Chapter 5). In practice, adjustment is, however, often preferred to such alternatives[5] because its statistical properties are well understood, giving access to useful methodologies like robust estimators and confidence intervals. In contrast, knowledge about the statistical properties of, e.g., front-door estimation is still lacking [Van09; GK17].

Identification in graphical models beyond DAGs can be much harder. These models represent a (possibly infinite) set of DAGs, and in order to identify the causal effect, one needs to find one expression involving only $P(\mathbf{o})$ that can identify the causal effect $P(\mathbf{y} \mid do(\mathbf{x}))$ in every represented DAG.

More specific graphical models like SEMs assume that the compatible probability distributions have a certain structure and allow the identification of causal effects that cannot be identified in more general models. For example, in linear SEMs variables are assumed to be normally distributed and only linearly affected by their parents. We will discuss identification in linear SEMs further in Chapter 6.

[5] Quoting [Van09], "Time will perhaps tell whether results like Pearl's front-door path adjustment theorem and its generalizations are actually useful for epidemiologic research or whether the results are simply of theoretical interest."

3 Separation: An Algorithmic Framework

In this chapter, we compile an algorithmic framework to solve m-separation problems in ancestral graphs and d-separation problems in restricted chain graphs. Tasks involving separating sets are one of the most fundamental primitives in the theory of causality, and they are necessary building blocks to solve complex causal problems. In the next chapter, we will apply the algorithms of this chapter to solve problems involving adjustment sets. Thus, it is important to provide efficient algorithms to solve separation problems in various causal models.

For given disjoint node sets $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ in a mixed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, we consider three classes of problems. *Testing* problems, where a set $\mathbf{Z}$ is given and the task is to decide whether $\mathbf{Z}$ separates $\mathbf{X}$ and $\mathbf{Y}$, *finding* problems, where the task is to find such a set $\mathbf{Z}$, and *enumeration* problems where we want to find all such sets $\mathbf{Z}$.

The separating set $\mathbf{Z}$ should always be bounded between given sets $\mathbf{I} \subseteq \mathbf{R} \subseteq \mathbf{V}$ as $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. The variables in $\mathbf{I}$ will always be included in the separating set even if the set remains a separator without these variables. The constraint $\mathbf{Z} \subseteq \mathbf{R}$ is necessary to handle unobserved variables, which cannot be used for separation, i.e., rather than distinguishing observed and unobserved variables as a property of a given graph we choose the more flexible approach of passing the observed variables as a parameter to the algorithms. It will also be important in the next chapter when we use the algorithms of this chapter to find adjustment sets. We will show that certain variables cannot be used in adjustment sets, so these forbidden variables are excluded from $\mathbf{R}$. In practice, including additional variables in an adjustment set can improve the precision of the causal effect estimation, even if these variables are not strictly necessary, so researchers might want to force inclusion of these variables using the constraining set $\mathbf{I}$. Both constraints $\mathbf{I}$ and $\mathbf{R}$ are also necessary to build our enumeration algorithm.

Another possible constraint is that $\mathbf{Z}$ should be minimal or minimum. A separating set is *minimal* if no strict subset of it is separating. It is *minimum* if no smaller separating set exists, whereby "smaller" can refer to the number of nodes in the set or a sum of individual node costs over all nodes in the set. We further distinguish between strong minimality and weak minimality ($\mathbf{I}$-minimality), depending whether the subset is required to contain $\mathbf{I}$. Smaller sets are preferable in practice since actually measuring the variables can be very expensive.

minimal
minimum

Our problems are defined in Table 3.1, which gives a summary of the main algorithmic results presented in this chapter, including the asymptotic runtimes of our algorithms. We use the same name for the problem and the algorithm that solves it.

The rest of this chapter presents our algorithms to solve the problems of Table 3.1. Section 3.1 proves fundamental facts about paths in causal graphs. Section 3.2 gives the basic reachability algorithm REACHABLE and algorithms TESTSEP, FINDSEP, and LISTSEP based on it. Section 3.3 describes the case of minimal and minimum separators, i.e., the precise difference

		Runtime	Proposition
Verification: For given $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and constraint $\mathbf{I}$, decide whether ...			
TESTSEP	$\mathbf{Z}$ separates $\mathbf{X}, \mathbf{Y}$	$\mathcal{O}(n+m)$	3.17
TESTMINSEP	$\mathbf{Z} \supseteq \mathbf{I}$ separates $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ is ...		
	$\mathbf{I}$-minimal	$\mathcal{O}(n+m)$	3.28
	strongly-minimal	$\mathcal{O}(n+m)$	3.28
Construction: For given $\mathbf{X}, \mathbf{Y}$ and constraints $\mathbf{I}, \mathbf{R}$, output a ...			
FINDSEP	separator $\mathbf{Z}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$\mathcal{O}(n+m)$	3.19
FINDMINSEP	separator $\mathbf{Z}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...		
	$\mathbf{I}$-minimal	$\mathcal{O}(n+m)$	3.29
	strongly-minimal	NP-complete	3.31
FINDMINCOSTSEP	separator $\mathbf{Z}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...		
	$\mathbf{I}$-minimum	$\mathcal{O}(n^3)$	3.34
	strongly-minimum	$\mathcal{O}(n^3)$	3.35
Enumeration: For given $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$, enumerate all ...		Delay	
LISTSEP	separators $\mathbf{Z}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$\mathcal{O}(n(n+m))$	3.20
LISTMINSEP	$\mathbf{I}$-minimal separators $\mathbf{Z}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$\mathcal{O}(n^3)$	3.33

Table 3.1: Definitions of algorithmic tasks related to m-separation in ancestral graphs and to d-separation in restricted chain graphs as well as the time complexities of our algorithms solving the associated problems. Throughout, $\mathbf{X}, \mathbf{Y}, \mathbf{R}$ are pairwise disjoint node sets, the set $\mathbf{Z}$ is disjoint with the non-empty sets $\mathbf{X}, \mathbf{Y}$, and each of the sets $\mathbf{I}, \mathbf{R}, \mathbf{Z}$ can be empty. n denotes the number of nodes and m the number of edges of the graph. A minimum separator minimizes the sum $\sum_{Z \in \mathbf{Z}} w(Z)$ for a cost function w respecting the given constraints, i.e., $w(V) = \infty$ for $V \notin \mathbf{R}$. The construction algorithms output $\perp$ if no set fulfilling the listed condition exists. Delay complexity, for e.g. LISTMINSEP, refers to the time needed to output one solution when there can be exponentially many solutions (see [Tak10]).

Problem	Work	Runtime	Constraints
TESTSEP	Shachter [Sha98]	$\mathcal{O}(n+m)$	
	our work	$\mathcal{O}(n+m)$	$\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$
FINDSEP	Tian et. al. [TPP98]	$\mathcal{O}(n+m)$	$\mathbf{Z} \subseteq \mathbf{R}$
	our work	$\mathcal{O}(n+m)$	$\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$
TESTMINSEP and FINDMINSEP			
	Tian et. al. [TPP98]	$\mathcal{O}(km)$	$\mathbf{Z} \subseteq \mathbf{R}$
	Tian et. al. [TPP98]	$\mathcal{O}(n^2)$	$\mathbf{Z} \subseteq \mathbf{R}$
	our work	$\mathcal{O}(n+m)$	$\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$
FINDMINCOSTSEP	Acid and De Campos [ADC96]	$\mathcal{O}(n^3)$	$\mathbf{I} \subseteq \mathbf{Z}$
	Tian et. al. [TPP98]	$\mathcal{O}(n^3)$	$\mathbf{Z} \subseteq \mathbf{R}$
	our work	$\mathcal{O}(n^3)$	$\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$
LISTMINSEP	Textor and Liśkiewicz [TL11]	delay $\mathcal{O}(n^3)$	$\mathbf{Z} \subseteq \mathbf{R}$
	our work	delay $\mathcal{O}(n^3)$	$\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$

Table 3.2: Comparison of our algorithms to the previously known best algorithms on separating sets in DAGs. We have either improved the runtime or the supported constraints. Here, $k = \mathcal{O}(n)$ is the (maximal) size of the separating set. We are not aware of any work studying these problems for AGs or CGs before our work, besides the trivial problems TESTSEP and FINDSEP.

between weakly-minimal and strongly-minimal as well as algorithms TESTMINSEP, FINDMINSEP, FINDMINCOSTSEP and LISTMINSEP. Section 3.4 shows how to use the algorithms for restricted chain graphs on (unrestricted) chain graphs.

Scientific Contribution. Prior to this work various algorithms to test and find d-separating sets were known [ADC96; Sha98; TPP98; TL11]. We have generalized them to ancestral graphs [ZLT14; ZLT19] and (restricted) chain graphs [ZL16b]. Later we have improved the algorithms for weakly minimal sets to run in linear time [ZL19]. Table 3.2 shows our improvements. The algorithms have been implemented in DAGitty [Tex+16].

3.1 Properties of Walks and Paths

Before we can describe the separation algorithms for ancestral graphs and restricted chain graphs, we need to study the paths that can occur in these graphs in great detail.

The key difference between general chain graphs and restricted chain graphs is that we can remove the last undirected edge from any possibly directed path that contains a configuration $\rightarrow\!\!-$ in restricted chain graphs. In other words, any possibly directed path that starts with a directed edge can be converted to a directed path.

Lemma 3.1. *Let $\mathcal{G}$ be an RCG. If there exists a possibly directed walk $w : X \overset{*}{\rightarrow} Y$ in $\mathcal{G}$ that contains a subwalk $U \rightarrow V - W$, there also exists the walk $w' = w_{XU} \rightarrow w_{WY}$ in $\mathcal{G}$ where w_{XU} (w_{WY}) is the subwalk of w from X to U (from W to Y).*

Proof. Since $\mathcal{G}$ is restricted, there exists an edge between U and W. This edge cannot be $U - W$ or $U \leftarrow W$ because it would create a semi-directed cycle. So there is an edge $U \rightarrow W$,

which we can use to form walk w'. □

The lemma uses the notation $w' = w_{XU} \to w_{WY}$ rather than $w' = w[X \overset{*}{\sim} U] \to w[W \overset{*}{\sim} Y]$, since latter notation might remove more than one node if the walk visits U or W multiple times.

In ancestral graphs, we cannot and do not need to remove $\to\!\!-$ from paths because this configuration never occurs in ancestral graphs:

Lemma 3.2. *Let $\mathcal{G}$ be an AG. No possibly directed walk $w : X \overset{*}{\rightarrow} Y$ in $\mathcal{G}$ contains a subwalk $U \to V - W$.*

Proof. The configuration $\to\!\!-$ is not allowed in an ancestral graph. □

So we can analyze paths in both graph classes as if they never contain $\to\!\!-$. Particularly, for any possibly directed path between X and Y, we can assume it is an undirected subpath followed by a directed subpath.

Corollary 3.3. *Given two nodes X, Y in an AG or RCG, it holds: $X \in pAn(Y)$ if and only if there exists a path $X \overset{*}{-}\!\!\overset{*}{\to} Y$.*

Proof. In an AG, this is Corollary 3.3 of [RS02]. In an RCG, Lemma 3.1 can be applied iteratively to remove all occurrences of $\to\!\!-$ until all undirected edges occur at the beginning of the path before directed edges. □

Another interesting corollary holds in ancestral graphs, which has also been shown by [RS02] for paths:

Corollary 3.4. *Let w be a walk in an AG. A collider C on w is an ancestor of* **Z** *if and only if C is a possible ancestor of* **Z**.

This corollary is not true for restricted chain graphs. For example, on the path $U \to V \leftarrow W$ in the RCG $\mathcal{G}_1$ of Figure 3.1, the node V is a possible ancestor of Z, but not an ancestor of Z.

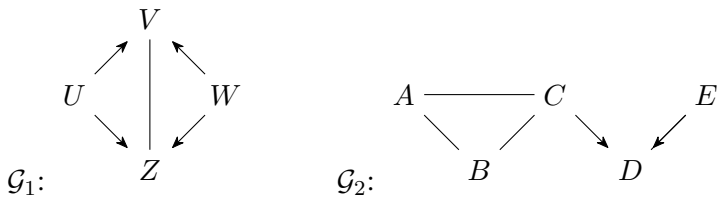


Figure 3.1: Two examples of restricted chain graphs (RCGs).

3.1.1 Almost Definite Status

Usually, d-separation in chain graphs is defined using the concept of definite status paths – paths on which every node has the same status as a collider or non-collider in the CG as in every represented DAG. Any collider on a path in a CG is a collider on that path in any represented DAG because directed edges of CGs remain directed edges in the represented DAGs. A non-collider might stay a non-collider or become a collider, so definite non-colliders are defined as non-colliders that do not become a collider in any represented DAG:

Definition 3.5 (**Definite Status** [Zha08])**.** *Let w be a walk in a mixed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$. An internal node V on w is called a* definite non-collider *if it occurs on w as $A \leftarrow V$, $V \rightarrow B$, or as an induced subgraph $A - V - B$, where A and B are the nodes preceding/succeeding V on w.* *definite non-collider*

An internal node V on w is said to be of definite status if it is either a collider or a definite non-collider on w.

A walk w is said to be of definite status *if all internal nodes on the walk are of definite status.* *definite status*

Lemma 3.6. *Let $\mathcal{G}$ be an RCG, $\mathcal{D} \in CE(\mathcal{G})$ a represented DAG, X, Y nodes, and $\mathbf{Z}$ a set. There exists a definite status path $\pi : X \overset{+}{\sim} Y$ active given $\mathbf{Z}$ in $\mathcal{G}$ iff there exists a path π' active given $\mathbf{Z}$ in $\mathcal{D}$.*

Proof. [Zha08] proves this for PAGs, and it is not hard to see that their proof also works for RCGs. □

"Induced subgraph $A - V - B$" in Definition 3.5 means V is only of definite status if A and B are not connected by an edge. This complicates the usage of definite status paths in algorithms because one cannot know if a given path is of definite status without checking the adjacencies in the underlying graph. For example, $A - V - B$ is of definite status in some graphs and not of definite status in other graphs.

Hence, we define a new, simpler kind of paths by dropping the condition of the unconnectedness of A and B.

Definition 3.7 (**Almost Definite Status**)**.** *Let w be a walk in a mixed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$. An internal node V on w is called an* almost definite non-collider *if it occurs on w as $A \leftarrow V$, $V \rightarrow B$, or as $A - V - B$, where A and B are the nodes preceding/succeeding V on w.* *almost definite non-collider*

An internal node V on w is said to be of almost definite status if it is either a collider or an almost definite non-collider on w.

A walk w is said to be of almost definite status *if all internal nodes on the walk are of almost definite status.* *almost definite status*

This property of almost definite status only depends on the edges in the paths, not on those outside the path. For example, in the RCG $\mathcal{G}_2$ of Figure 3.1 the path $A - B - C \rightarrow D \leftarrow E$ is of almost definite status. It is not of definite status, because A and C are connected, and thus there exists a consistent DAG extension of $\mathcal{G}_2$ that contains a collider $A \rightarrow B \leftarrow C$.

In chain graphs, the only configurations not of almost definite status are $\rightarrow -$ and $- \leftarrow$. In general mixed graphs containing bidirected edges, two additional configurations could occur that are also not of almost definite status, $\leftrightarrow -$ and $- \leftrightarrow$.

Since ancestral graphs are mixed graphs, we can also apply Definition 3.7 to walks in ancestral graphs. However, the definition is not meaningful there:

Lemma 3.8. *Every path and walk in an ancestral graph is of almost definite status.*

Proof. The configurations $\rightarrow -$, $- \leftarrow$, $\leftrightarrow -$, and $- \leftrightarrow$ do not occur in ancestral graphs. □

3.1.2 Equivalences

So far, we have introduced various notions of connectedness between nodes – paths, walks, almost definite status path, definite status paths, etc. We will now show that all these notions

are equivalent. First, we continue with almost definite status paths and show the equivalence between almost definite status and definite status in AGs and RCGs:

Lemma 3.9. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an AG or RCG, and X, Y nodes. There exists a definite status walk (path) $w_{ds} : X \overset{+}{\sim} Y$ if and only if there exists an almost definite status walk (path) $w_{ads} : X \overset{+}{\sim} Y$. Moreover, both walks (paths) have the same directed edges and every non-collider or collider of w_{ds} occurs on w_{ads} .*

Proof. Let w_{ads} be the shortest almost definite status walk between X and Y for which no such w_{ds} exists. Then w_{ads} contains a subwalk $U - V - W$ such that U and W are connected by an edge. This edge cannot be (bi-)directed (in an AG) or there would be a semi-directed cycle (in an RCG). Thus, we can replace $U - V - W$ by $U - W$. Since U and W still have the same kind of adjacent edges, this replacement yields a shorter almost definite status walk, which is either of definite status and satisfies the conditions of the lemma or contradicts w_{ads} being the shortest walk without a w_{ds}. The proof of the opposite implication is trivial since every definite status walk is an almost definite status walk. The same argument holds for paths. □

Corollary 3.10. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an AG or RCG, X, Y nodes, and $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ a node set. There exists a definite status walk (path) $w_{ds} : X \overset{+}{\sim} Y$ active given $\mathbf{Z}$ if and only if there exists an almost definite status walk (path) $w_{ads} : X \overset{+}{\sim} Y$ active given $\mathbf{Z}$.*

This corollary implies that Lemma 3.6 also holds for almost definite status paths in RCGs. However, it is no longer true that the corresponding paths in the represented DAGs have exactly the same nodes as the path in the RCG.

The equivalence of paths and walks is well-known as deleting all nodes with multiple occurrences from a walk yields a path, e.g., [RS02, Section 3.4.2] for ancestral graphs. For later proofs, it is also important to know that the colliders and non-colliders on the walk and the resulting path are closely related:

Lemma 3.11. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an AG or RCG, X, Y nodes, and $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ a node set.*

If there exists an almost definite status walk $w : X \overset{*}{\sim} Y$ active given $\mathbf{Z}$, there exists an almost definite status path $\pi' : X \overset{*}{\sim} Y$ active given $\mathbf{Z}$, such that

1. *every collider on π' is an ancestor of a collider on w,*
2. *π' has at most as many colliders as w, and*
3. *every non-collider on π' occurs as a non-collider on w.*

Proof. Let w be the shortest walk for which this lemma does not hold. If every node on w occurs only once, w is a path with the same colliders and non-colliders as the walk.

Otherwise, there is a first duplicate node D. Let w'' be the walk with the nodes between the first and last occurrence of D removed. All nodes of w'' except D have the same state as non-colliders/colliders on both w and w'', so we only need to show the properties for D.

If D is a non-collider on w'', one of its adjacent edges has no arrow pointing towards D, and this edge also exists on w, so D is a non-collider on w and not in $\mathbf{Z}$. If D is not of almost definite status, $\mathcal{G}$ is an RCG and there is a subwalk $C \to D \overset{*}{-} E$ or $C \overset{*}{-} D \leftarrow E$ of w''. Due to Lemma 3.1, we can replace this subwalk with $C \to E$ or $C \leftarrow E$ without introducing a new collider because no edge can point to a node adjacent to an undirected edge in w. The resulting

walk is of almost definite status and d-connected as C and E are non-colliders on w and not in $\mathbf{Z}$.

If D is a collider on w'' and not on w, D occurs in a subwalk $\rightleftarrows\ D \rightarrow^* \leftarrow D\ \leftrightarrows$ in w. Hence, D is of definite status and an ancestor of a collider.

Because w'' is shorter than w, there is a path π' that has the required properties relative to w'' and thus also relative to w. □

Now we are ready for the big lemma that shows the equivalence of all possible definitions of separation in AGs and RCGs, including several possible variations of the condition that the colliders in a path need to satisfy. We abbreviate definite status as ds- and almost definite status as ads- for the sake of brevity.

Lemma 3.12. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an AG or RCG, and $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ disjoint sets of nodes. The following statements are equivalent:*

(def)	$\mathbf{X}$ *and* $\mathbf{Y}$ *are not* d*-separated (*m*-separated) by* $\mathbf{Z}$.	
(*ds-p-An*)	*There exists a ds-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in An*$(\mathbf{Z})$.
(*ds-p-pAn*)	*There exists a ds-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in pAn*$(\mathbf{Z})$.
(*ds-p-AnXY*)	*There exists a ds-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in An*$(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.
(*ds-p-pAnXY*)	*There exists a ds-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in pAn*$(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.
(*ads-p-An*)	*There exists an ads-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in An*$(\mathbf{Z})$.
(*ads-p-pAn*)	*There exists an ads-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in pAn*$(\mathbf{Z})$.
(*ads-p-AnXY*)	*There exists an ads-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in An*$(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.
(*ads-p-pAnXY*)	*There exists an ads-path* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in pAn*$(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.
(*ds-w-Z*)	*There exists a ds-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in* $\mathbf{Z}$.
(*ds-w-An*)	*There exists a ds-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in An*$(\mathbf{Z})$.
(*ds-w-pAn*)	*There exists a ds-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in pAn*$(\mathbf{Z})$.
(*ds-w-AnXY*)	*There exists a ds-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in An*$(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.
(*ds-w-pAnXY*)	*There exists a ds-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in pAn*$(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.
(*ads-w-Z*)	*There exists an ads-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in* $\mathbf{Z}$.
(*ads-w-An*)	*There exists an ads-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in An*$(\mathbf{Z})$.
(*ads-w-pAn*)	*There exists an ads-walk* $\mathbf{X} \stackrel{+}{\sim} \mathbf{Y}$	*containing no non-collider in* $\mathbf{Z}$ *and no collider not in pAn*$(\mathbf{Z})$.

(*ads-w-AnXY*) *There exists an ads-walk* $\mathbf{X} \overset{+}{\sim} \mathbf{Y}$ *containing no non-collider in* $\mathbf{Z}$ *and no collider not in* $An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.

(*ads-w-pAnXY*) *There exists an ads-walk* $\mathbf{X} \overset{+}{\sim} \mathbf{Y}$ *containing no non-collider in* $\mathbf{Z}$ *and no collider not in* $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.

Proof. (*def*) $\Leftrightarrow$ (*ds-p-An*): This is the definition of d-separation in chain graphs resp. m-separation in ancestral graphs.

(*ds-w-pAnXY*) $\Rightarrow$ (*ds-w-Z*): (*ads-w-pAnXY*) $\Rightarrow$ (*ads-w-Z*): Every subwalk $B \to C \leftarrow D$ with a collider C can be replaced by $B \to C \overset{*}{\to} V \overset{*}{\leftarrow} C \leftarrow D$ with $V \in \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$ such that all non-colliders between C and V are not in $\mathbf{Z}$. Due to Lemma 3.1 (in RCGs) or Lemma 3.2 (in AGs), the subwalk can again be replaced by $B \overset{+}{\to} V \overset{+}{\leftarrow} D$. This replacement is of (almost) definite status since only directed edges pointing away from the path are added. After truncating the walk to start at its last node in $\mathbf{X}$ and end at the next node in $\mathbf{Y}$, all colliders are in $\mathbf{Z}$.

(*w-AntXY*) $\Rightarrow$ (*w-Z*): Every collider C on the walk can be replaced by a walk $C \overset{*}{\to} V \overset{*}{\leftarrow} C$ with $V \in \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$ such that all non-colliders between C and V are not in $\mathbf{Z}$. After truncating the walk to start at its last node in $\mathbf{X}$ and end at the next node in $\mathbf{Y}$, all colliders are in $\mathbf{Z}$. (for ancestral graphs this is also Lemma 3.14 in [RS02].)

(*ads-w-Z*) $\Rightarrow$ (*ads-p-An*): This is Lemma 3.11.

(*ads-p-An*) $\Rightarrow$ (*ds-p-An*): This is Lemma 3.9.

This completes the proof together with the trivial directions that definite status implies almost definite status, paths are walks, and $\mathbf{Z} \subseteq An(\mathbf{Z}) \subseteq An(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$. Every case trivially implies case (*ads-w-pAnXY*) and is trivially implied by either (*ds-p-An*) or (*ds-w-Z*), and we have just shown (*ads-w-pAnXY*) $\Rightarrow$ (*ads-w-Z*) $\Rightarrow$ (*ads-p-An*) $\Rightarrow$ (*ds-p-An*) $\Rightarrow$ (*ds-w-pAnXY*) $\Rightarrow$ (*ds-w-Z*). □

3.1.3 Augmentation and Moralization

D-separation and m-separation are not monotonic, that is, adding a node to a separating set can unblock a path and result in a non-separating set, while removing it again can turn a non-separating set into a separating set. Often it is helpful to have a monotonic separation criterion, e.g., for finding minimal separating sets by removing all nodes until no further node can be removed without opening a path. In DAGs, a commonly used criterion is separation in the moral graph whereby the DAG is converted to an undirected graph – the moral graph – in which d-separation becomes equivalent to an st-vertex cut [LS88; Lau+90]. This generalizes well to AGs and RCGs. We will introduce the definition of the moral graph step by step:

Lemma 3.13. *If* $w : \mathbf{X} \overset{*}{\sim} \mathbf{Y}$ *is an almost definite status walk active given* $\mathbf{Z}$ *in an RCG or AG, all nodes of* w *are in* $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$.

Proof. For AGs, this is Lemma 3.13 in [RS02]. For RCGs, let w be a walk with a node $V \notin pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$. Assume V is the first such node and U the preceding node. Then V is a child of U. The walk does not contain $U \to V$ — as it is of almost definite status, nor $U \to V \leftarrow$, since V is not an ancestor of $\mathbf{Z}$. For the same reasons, the walk does not contain $U \to V \overset{+}{\to}$— or $U \to V \overset{+}{\to} \leftarrow$. Hence, it ends with $U \to V \overset{+}{\to} Y$, and V is an ancestor of Y. □

So Lemma 3.13 says all nodes outside of $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$ can be ignored, and Lemma 3.12 says all colliders within $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$ cannot block the path, which implies the only relevant

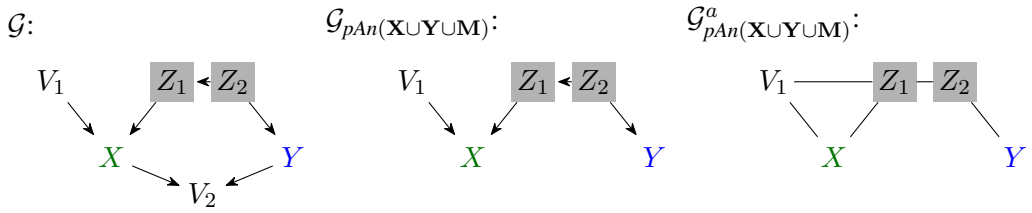


Figure 3.2: The transformation of a graph $\mathcal{G}$ to $\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{M})}$ to $(\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{M})})^a$ with $\mathbf{M} = \emptyset$. The separating set $\{Z_1, Z_2\}$ is not minimal as no node is reachable from both $\mathbf{X}$ and $\mathbf{Y}$, but each node alone, $\{Z_1\}$ or $\{Z_2\}$, is a minimal separator.

nodes are non-colliders in $pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{Z})$. Hence, in the subgraph $\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{Z})}$, the separation criterion can be simplified to "A path is active if and only if no non-collider is in $\mathbf{Z}$".

It can be simplified even further to "A path is active if and only if it contains no node in $\mathbf{Z}$" if we ensure that the path contains no colliders. For this, we insert an undirected edge between every two nodes that can be connected by a path on which every node is a collider as shown in Figure 3.2. Then all colliders on any path can be replaced by this undirected edge, so, for any path, there exists an equivalent path without colliders.

Definition 3.14. *The* augmented graph $(\mathcal{G})^a$ *of a certain mixed graph $\mathcal{G}$ is an undirected graph with the same nodes as $\mathcal{G}$ whose edges are all pairs of nodes that are collider connected in $\mathcal{G}$.*

augmented graph

In DAGs or RCGs, every collider on a path is a child of two parents that are both adjacent to the collider on the path. Thus, constructing the augmented graph inserts an edge between those parents, and one could also say the parents are getting married because they have a child. Hence, the augmented graph is also called the *moral graph* and its construction *moralization*.

moral graph
moralization

Lemma 3.15. *Given an AG or RCG $\mathcal{G}$ and three disjoint sets $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$, set $\mathbf{Z}$ separates $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$ if and only if $\mathbf{Z}$ intersects every path between $\mathbf{X}$ and $\mathbf{Y}$ in $(\mathcal{G}_{pAn(\mathbf{X},\mathbf{Y},\mathbf{Z})})^a$.*

Because the augmented graph is an undirected graph, vertex cuts in the augmented graph are monotonic.

Since separation in the moral graph is equivalent to d/m-separation, both approaches can be used as the canonical separation criterion in causal graphs. [RS02] has shown that moralization works in ancestral graphs. Some authors [Fry90; BS95] define separation of chain graphs only in terms of moralization. The fact that we can moralize restricted chain graphs, shows that the definition of separation with (almost) definite status paths is equivalent to their definitions.

3.2 Algorithms for Separators

In this section, we present algorithms that test, find, and enumerate sets $\mathbf{Z}$ that separate sets $\mathbf{X}$ and $\mathbf{Y}$ under a constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ given disjoint node sets $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ in an AG or RCG.

In later chapters, when using these algorithms for adjustment sets, the constraint given by the set $\mathbf{R}$ corresponds to the nodes forbidden by the condition (a) of the criterion (Definition 4.33). The constraint given by $\mathbf{I}$ helps to enumerate all such sets.

3.2.1 Reachability Algorithm

The reachability problem might the most fundamental graph problem and is an important building block for more complex algorithms. We will use a breadth-first search (BFS) algorithm throughout this thesis to find all nodes reachable from a node set $\mathbf{X}$ in a graph $\mathcal{G}$. Here we give the pseudocode for BFS as a higher-order function, so we do not have to specify in every later algorithm how to keep track of visited nodes and edges:

function REACHABLE($\mathcal{G}, \mathbf{X}, accept$)
  $\mathbf{Q} \leftarrow \{(\leftarrow, X) \mid X \in \mathbf{X}\}$ ▷ queue of pairs (type of edge, node) to visit
  $\mathbf{P} \leftarrow \mathbf{Q}$ ▷ all processed pairs
  while $\mathbf{Q}$ not empty **do**
    Let (e, T) be a (type of edge, node) pair of $\mathbf{Q}$
    Remove (e, T) from $\mathbf{Q}$
    for all neighbors N of T **do**
      Let T and N be connected by edge f
      if $(f, N) \notin \mathbf{P}$ **and** $accept(e, T, f, N)$ **then**
        Add (f, N) to $\mathbf{Q}$ and $\mathbf{P}$
  return $\{V \mid (e, V) \in \mathbf{P}$ for any edge type $e\}$

Analysis of the Algorithm. Algorithm REACHABLE returns all nodes $V \in \mathbf{V}$ that are reachable from a node $X \in \mathbf{X}$ by a walk $w = X \overset{*}{\sim} V$ such that function $accept$ returns TRUE for all subwalks of the form $\sim T \sim N$ on the walk $\leftarrow w$. Each node is only visited once from each adjacent edge, so the runtime is $\mathcal{O}(n + m)$. □

We will use the notation **function**$(\cdot, \cdot, \cdot, \cdot)\{\ldots$ **return** $\ldots\}$ for the function passed to REACHABLE as an actual $accept$ parameter. For example, we can define the trivial functions *An*, *De*, *pAn*, and *pDe* using function REACHABLE like this:

function $An_{\mathcal{G}}(\mathbf{X})$
  return REACHABLE($\mathcal{G}, \mathbf{X},$ **function**$(e, U, f, V)\{$**return** $f = \leftarrow\}$)

function $De_{\mathcal{G}}(\mathbf{X})$
  return REACHABLE($\mathcal{G}, \mathbf{X},$ **function**$(e, U, f, V)\{$**return** $f = \rightarrow\}$)

function $pAn_{\mathcal{G}}(\mathbf{X})$
  return REACHABLE($\mathcal{G}, \mathbf{X},$ **function**$(e, U, f, V)\{$**return** $f \in \{\leftarrow, —\}\}$)

function $pDe_{\mathcal{G}}(\mathbf{X})$
  return REACHABLE($\mathcal{G}, \mathbf{X},$ **function**$(e, U, f, V)\{$**return** $f \in \{\rightarrow, —\}\}$)

The decision variant of the reachability problem, i.e., given $V \in \mathbf{V}$, deciding whether $V \in \text{REACHABLE}(\mathcal{G}, \mathbf{X}, accept)$ with a constant time $accept$ function, is NL-complete for DAGs, where NL denotes the class of problems which can be solved by nondeterministic Turing machines in logarithmic space [Pap93]. There reachability can be solved by guessing the next node N nondeterministically. This will also work for AG and RCGs, so all our functions

only calling REACHABLE are likely also in NL or even NL-complete when formulated as a decision problem. However, we will not investigate this further, since linear-time algorithms are more useful than nondeterministic logspace algorithms in practice.

A property of reachability is that if one removes edges from a graph, REACHABLE and functions defined through REACHABLE like *An* return fewer nodes.

Claim 3.16. *Given a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, a subgraph $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$, sets $\mathbf{V}' \subseteq \mathbf{V}$, $\mathbf{E}' \subseteq \mathbf{E}$, $\mathbf{X}' \subseteq \mathbf{X} \subseteq \mathbf{V}'$, and functions $accept$ and $accept'$ s.t. $accept(e, U, f, V)$ always returns* TRUE *when $accept'(e, U, f, V)$ returns* TRUE *on all visited walks $eUfV$, then* REACHABLE$(\mathcal{G}', \mathbf{X}', accept') \subseteq$ REACHABLE$(\mathcal{G}, \mathbf{X}, accept)$.

Proof. If a node is not visited, it can only lead to further nodes that are not visited, and never to more visited nodes. □

3.2.2 Testing Separators

Problem TESTSEP just requires us to verify the d/m-separation definition for given sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, so a simple reachability search implementing the separation rules for almost definite status walks suffices. Figure 3.3 shows all edge configurations that might occur on a walk between one node and the next and whether this next node is reachable through the edges. In DAGs, the commonly used algorithm to test d-separation in this way is the Bayes-Ball algorithm [Sha98], which is visualized as sending balls through the DAG along the edges. If the next node is reachable, the ball passes through to it, otherwise the ball is blocked. If the ball moves back to the previously visited node, it is said to have bounced back.

We obtain the following algorithm for testing whether a given $\mathbf{Z}$ separates $\mathbf{X}$ and $\mathbf{Y}$:

function TESTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$)
  $accept :=$ **function**$(e, U, f, V)\{$
    return (eUf is of almost definite status)
      $\wedge$ ((U is a collider on $eUf \wedge U \in \mathbf{Z}$) $\vee$ (U is a non-collider on $eUf \wedge U \notin \mathbf{Z}$))
  $\}$
  return REACHABLE$(\mathcal{G}, \mathbf{X}, accept) \cap \mathbf{Y} = \emptyset$

Analysis of the Algorithm. From the rules in Figure 3.3, it is obvious that the following statement about algorithm TESTSEP holds: The ball only passes through the walk segment of two consecutive edges if the segment is an almost definite active walk. Due to Lemma 3.12, testing whether a collider on a walk is in $\mathbf{W}$ is equivalent to testing whether it is in $An(\mathbf{W})$. The correctness follows from the fact that the algorithm searches over all walks starting in $\mathbf{X}$. The runtime is linear as it only calls REACHABLE. □

Proposition 3.17. *Using the algorithm above, the task* TESTSEP *can be solved in time $\mathcal{O}(n+m)$ in AGs and RCGs.*

3.2.3 Finding Separators

The next problem FINDSEP asks to find a set $\mathbf{Z}$ separating certain sets $\mathbf{X}$ and $\mathbf{Y}$ under the constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. From the moral graph, we know that only the ancestors of $\mathbf{X}$ and $\mathbf{Y}$ are relevant, which implies a canonical, closed-form solution for this problem:

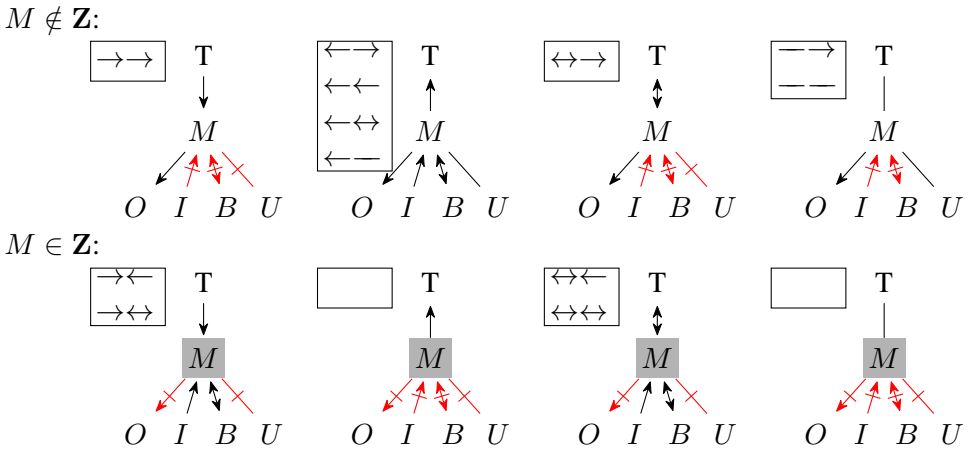


Figure 3.3: Expanded rules for Bayes-Ball in AGs and RCGs, listing (in boxes) all combinations of edge pairs through which the ball is allowed to pass. The Bayes ball starts at the top node T and passes through the middle node M to one of the bottom nodes $\{O, I, B, U\}$. Forbidden passes are marked in red and crossed out. Here, by a pair of edges, we mean an edge between T (Top node) and M (Middle node) and $M \to O$ (Out-node), resp. $M \leftarrow I$ (In-node), $M \leftrightarrow B$ (Bidirected edge), and $M - U$ (Undirected edge). The figure above shows all possible types of edges between T and M. We consider two cases: $M \notin \mathbf{Z}$ and $M \in \mathbf{Z}$ (gray). The leaving edge can correspond to the entering edge, i.e., T can belong to $\{O, I, B, U\}$, in which case the ball might return to T, which is called a bouncing ball in the Bayes-Ball algorithm.

Lemma 3.18. *Let* $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ *be sets of nodes with* $\mathbf{I} \subseteq \mathbf{R}$, $\mathbf{R} \cap (\mathbf{X} \cup \mathbf{Y}) = \emptyset$. *If there exists a separator* $\mathbf{Z}_0$ *relative to* $(\mathbf{X}, \mathbf{Y})$ *with* $\mathbf{I} \subseteq \mathbf{Z}_0 \subseteq \mathbf{R}$, *then* $\mathbf{Z} = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \cap \mathbf{R}$ *is a separator relative to* $(\mathbf{X}, \mathbf{Y})$.

Proof. Let us consider a proper walk $w : \mathbf{X} \overset{+}{\sim} \mathbf{Y}$ not blocked by $\mathbf{Z}$. Due to Lemma 3.13, $w \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$. If there is just one non-collider on w in $\mathbf{R}$, $\mathbf{Z}$ would block w, so all non-colliders are not in $\mathbf{R}$ and thus not in $\mathbf{Z}_0$. All colliders are in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}_0)$, so according to Lemma 3.12 $\mathbf{Z}_0$ is not a separator. □

This lemma generalizes well-known results [SGS01] and adds the constraint $\mathbf{I}$. It yields the following linear-time algorithm to find a separator relative to $(\mathbf{X}, \mathbf{Y})$:

function FINDSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$)
  $\mathbf{R}' \leftarrow \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y})$
  $\mathbf{Z} \leftarrow pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \cap \mathbf{R}'$
  if TESTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$) **then return** $\mathbf{Z}$
  else return $\perp$

Analysis of the Algorithm. The algorithm finds a separator due to Lemma 3.18 and runs in linear time, since the set $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \cap \mathbf{R}$ can be calculated in linear time and the algorithm TESTSEP runs in linear time as well. □

Proposition 3.19. *Using the algorithm above, the task* FINDSEP *can be solved in time* $\mathcal{O}(n+m)$ *in AGs and RCGs.*

Set $\mathbf{R}$ is required to be disjoint with $\mathbf{X}$ and $\mathbf{Y}$ because separation of a set $\mathbf{Z}$ relative to $\mathbf{X}, \mathbf{Y}$ is not defined when $\mathbf{Z}$ contains a node of $\mathbf{X}$ or $\mathbf{Y}$, so $\mathbf{Z}$ is always a subset of $\mathbf{R} \subseteq \mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Y})$. However, our algorithms still remove $\mathbf{X} \cup \mathbf{Y}$ from $\mathbf{R}$ as it might prevent bugs in practical implementations and it is reasonable to find a separator that does not contain the variables to be separated.

3.2.4 Enumerating All Separators

After finding a single separator, it is also interesting to know which other separators exist and enumerate all or some of them.

Since there might be an exponential number of separators, it is not possible to list all of them in polynomial time. For example, a directed path of length k from X to Y can be blocked by any separator that contains at least one node of the path, so there are $2^k - 1$ many separators. Therefore, we aim for polynomial delay time, which means that only polynomial time passes between the start or the output of a separator and the output of the next separator or the end of the algorithm. This approach is similar to the st-cut enumeration algorithm by [Tak10].

function LISTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$)
  if FINDSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$) $\neq \perp$ **then**
    if $\mathbf{I} = \mathbf{R}$ **then** Output $\mathbf{I}$
    else
      $V \leftarrow$ an arbitrary node of $\mathbf{R} \setminus \mathbf{I}$
      LISTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I} \cup \{V\}, \mathbf{R}$)
      LISTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R} \setminus \{V\}$)

Analysis of the Algorithm. Algorithm LISTSEP performs backtracking to enumerate all $\mathbf{Z}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$, aborting branches that will not find a valid separator. Since every leaf of the implicit search tree will output a separator, the tree height is at most n. The existence check via the call of FINDSEP takes $\mathcal{O}(n + m)$ time, so the delay time of LISTSEP is $\mathcal{O}(n(n + m))$. The algorithm generates every separator exactly once: if initially $\mathbf{I} \subsetneq \mathbf{R}$, with $V \in \mathbf{R} \setminus \mathbf{I}$, then the first recursive call returns all separators $\mathbf{Z}$ with $V \in \mathbf{Z}$ and the second call returns all $\mathbf{Z}'$ with $V \notin \mathbf{Z}'$. Thus, the generated separators are pairwise disjoint. □

Proposition 3.20. *Using the algorithm above, the task* LISTSEP *can be solved with polynomial delay $\mathcal{O}(n(n + m))$ in AGs and RCGs.*

3.3 Algorithms for Minimal and Minimum Separators

Observing variables can be very expensive, so it is desirable to work with separators that contain no unnecessary nodes. Hence, in this section, we will modify the algorithms of the previous section to search minimal or minimum sets $\mathbf{Z}$ that separate sets $\mathbf{X}$ and $\mathbf{Y}$ under a constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ given disjoint node sets $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ in an AG or RCG.

3.3.1 Weak Minimality versus Strong Minimality

First we need to define the meaning of the terms *minimal* and *minimum* precisely. A naive definition could be, a separator $\mathbf{Z}$ is minimal if there exists no separator $\mathbf{Z}' \subsetneq \mathbf{Z}$. However, this ignores the constraint $\mathbf{I}$. In some situations, removing the nodes of $\mathbf{I}$ from $\mathbf{Z}$ might lead

minimal
minimum

to a smaller separator, but if someone has included these nodes in $\mathbf{I}$, they probably want those nodes to be in every smaller separator as well. Hence, it is reasonable to require that $\mathbf{Z}'$ should satisfy the constraint $\mathbf{I} \subseteq \mathbf{Z}' \subseteq \mathbf{R}$. In order to analyze both ideas, we introduce the concept of $\mathbf{M}$-minimality:

$\mathbf{M}$-*minimal*

Definition 3.21. *For pairwise disjoint* $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$*, and a subset* $\mathbf{M}$ *of* $\mathbf{V}$*, a separator* $\mathbf{Z}$ *relative to* $(\mathbf{X}, \mathbf{Y})$ *is* $\mathbf{M}$-minimal *if* $\mathbf{M} \subseteq \mathbf{Z}$ *and no set* $\mathbf{Z}' \subsetneq \mathbf{Z}$ *with* $\mathbf{M} \subseteq \mathbf{Z}'$ *is a separator relative to* $(\mathbf{X}, \mathbf{Y})$*.*

For $\mathbf{M} = \emptyset$, i.e., $\emptyset$-minimal separators, this definition corresponds to the former idea, while $\mathbf{M} = \mathbf{I}$, i.e., $\mathbf{I}$-minimal separators, matches the latter idea.

$\mathbf{M}$-*minimum*

Definition 3.22. *For pairwise disjoint* $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$*, and a subset* $\mathbf{M}$ *of* $\mathbf{V}$*, a separator* $\mathbf{Z}$ *relative to* $(\mathbf{X}, \mathbf{Y})$ *is* $\mathbf{M}$-minimum *according to a cost function* $w : \mathbf{V} \to \mathbb{R}^+$*, if* $\mathbf{M} \subseteq \mathbf{Z}$ *and no set* $\mathbf{Z}'$ *with* $\mathbf{M} \subseteq \mathbf{Z}'$ *and* $\sum_{Z \in \mathbf{Z}'} w(Z) < \sum_{Z \in \mathbf{Z}} w(Z)$ *is a separator relative to* $(\mathbf{X}, \mathbf{Y})$*.*

If no cost function is given, we mean by a minimum separator minimum according to the cost function $w(v) = 1$, i.e. according to the cardinality.

strongly-minimal *strongly-minimum* *weakly-minimal* *weakly-minimum*

We will also call $\emptyset$-minimal ($\emptyset$-minimum) sets *strongly-minimal* (*strongly-minimum*), and $\mathbf{I}$-minimal ($\mathbf{I}$-minimum) sets *weakly-minimal* (*weakly-minimum*).

Note a subtle, but important difference between weak and strong minimality, the existence of a weakly-minimal separator does not necessarily imply that a strongly-minimal separator exists. For instance, in the DAG $X \to I \leftarrow V \to Y$, set $\mathbf{Z} = \{I, V\}$ is an I-minimal separator relative to (X, Y), but there exists no strongly-minimal separator $\mathbf{Z}'$, with $I \subseteq \mathbf{Z}'$. On the other hand, it is easy to see that every strongly-minimal separator $\mathbf{Z}$, with $\mathbf{I} \subseteq \mathbf{Z}$, is also an $\mathbf{I}$-minimal separator and the same holds for the minimum sets.

The difference between $\mathbf{I}$-minimal and strongly-minimal separators $\mathbf{Z} \supseteq \mathbf{I}$ is also illustrated in the earlier Figure 3.2. There are exactly two strongly-minimal separating sets: $\{Z_1\}$ and $\{Z_2\}$. No other separator will be strongly-minimal regardless of which nodes are added to a constraining set $\mathbf{I}$. Therefore, for $\mathbf{I} = \emptyset$, both separators satisfy the constraint, for $\mathbf{I} = \{Z_1\}$ or $\mathbf{I} = \{Z_2\}$, only one of them does and, for $\mathbf{I} = \{Z_1, Z_2\}$ or $V_1 \in \mathbf{I}$, no strongly-minimal separator satisfies it. The constraint $\mathbf{I}$ just chooses some separators of a fixed set of strongly-minimal separators.

On the other hand, when computing an $\mathbf{I}$-minimal separator, we treat the nodes of $\mathbf{I}$ as fixed and search for a minimal separator among all supersets of $\mathbf{I}$. In Figure 3.2, if $\mathbf{I}$ is either $\{Z_1\}$, $\{Z_2\}$ or $\{Z_1, Z_2\}$, then $\mathbf{I}$ itself is an $\mathbf{I}$-minimal separator and no other $\mathbf{I}$-minimal separator exists. If $\mathbf{I} = \{V_1\}$, then $\{Z_1, V_1\}$ and $\{Z_2, V_1\}$ are $\mathbf{I}$-minimal. This is an easier and in some sense more natural concept, since it can be modeled by removing the nodes of $\mathbf{I}$ from the graph and searching a minimal separator for the remaining nodes.

From a covariate adjustment perspective, the definition of $\mathbf{M}$-minimality is most meaningful in the case $\mathbf{M} = \emptyset$ or $\mathbf{M} = \mathbf{I}$. However, our algorithms technically also allow $\mathbf{M}$ to lie "between" $\emptyset$ and $\mathbf{I}$ or even partly outside $\mathbf{M}$, even though this is less relevant for our application. For example, if $\emptyset \neq \mathbf{M} \subset \mathbf{I}$, the nodes of $\mathbf{M}$ can be ignored for the minimality, while the nodes of $\mathbf{I} \setminus \mathbf{M}$ must be unremovable like all nodes in the case of $\emptyset$-minimality. In an example, in Figure 3.2, for $\mathbf{M} = \{V_1\}, \mathbf{I} = \{Z_1, V_1\}$ would only accept $\{Z_1, V_1\}$ as a separator. $\mathbf{M} = \{Z_1\}, \mathbf{I} = \{Z_1, Z_2\}$ would not allow any. Every separator $\mathbf{Z}$ is $\mathbf{Z}$-minimal. Without the condition that an $\mathbf{M}$-minimal separator needs to contain $\mathbf{M}$, every separator would also be $\mathbf{R}$-minimal and $\mathbf{V}$-minimal.

The only one of these four variations that appears to be turned into a hard problem by the restriction $\mathbf{Z} \supseteq \mathbf{I}$ is computing strongly-minimal separators – we discuss its complexity in detail in Section 3.3.5. This is a very surprising result since finding objects of minimum costs or sizes is typically harder than constructing minimal objects.

Thus, some of our efficient algorithms will only work with $\mathbf{I}$-minimal sets. $\mathbf{I}$-minimality can always be handled by removing the nodes of $\mathbf{I}$ from the graph and connecting their former neighbors before applying the algorithms, but we will show more efficient ways.

3.3.2 Properties of Minimal Separators

Lemma 3.18 gave us a closed-form solution to find a separator, which yields a constraint for minimal separators:

Corollary 3.23 (Ancestry of minimal separators)**.** *Given an AG or RCG $\mathcal{G}$ and three sets $\mathbf{X}, \mathbf{Y}, \mathbf{M}$, every $\mathbf{M}$-minimal separator $\mathbf{Z}$ is a subset of $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$.*

Proof. Assume there is an $\mathbf{M}$-minimal separator $\mathbf{Z}$ with $\mathbf{Z} \not\subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$. Setting $\mathbf{I} = \mathbf{M}$ and $\mathbf{R} = \mathbf{Z}$, Lemma 3.18 shows that $\mathbf{Z}' = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) \cap \mathbf{Z}$ is a separator with $\mathbf{M} \subseteq \mathbf{Z}'$. But $\mathbf{Z}' \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$ and $\mathbf{Z}' \subseteq \mathbf{Z}$, so $\mathbf{Z} \neq \mathbf{Z}'$ and $\mathbf{Z}$ is not an $\mathbf{M}$-minimal separator. □

This implies, if $\mathbf{I} \subsetneq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$, no $\mathbf{M}$-minimal separator containing $\mathbf{I}$ exists.

Corollary 3.24 (Ancestry of minimal separators)**.** *Given an AG or RCG $\mathcal{G}$ and three sets $\mathbf{X}, \mathbf{Y}, \mathbf{I}$, every $\mathbf{I}$-minimal or $\emptyset$-minimal separator is a subset of $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$.*

Proof. This follows from Corollary 3.23 with $\mathbf{M} = \mathbf{I}$ or $\mathbf{M} = \emptyset$. In both cases, we have $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$. □

This means that $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$ for every minimal separator $\mathbf{Z} \supseteq \mathbf{I}$. So the moral graph for every minimal separator $\mathbf{Z}$ is $(\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{Z})})^a = (\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})})^a$ and can be constructed without knowing $\mathbf{Z}$. Thus, all problems involving minimal separators can be solved with corresponding standard algorithms in the undirected, moral graph. However, the moral graph can have $O(n^2)$ many edges, e.g., if there is one node that is a child of all other nodes. Hence, there are no linear time algorithms using the moral graph, and we will use another approach to solve the problems.

[TPP98] show that a separator $\mathbf{Z}$ in a DAG is minimal if and only if no node $Z \in \mathbf{Z}$ can be removed from $\mathbf{Z}$, i.e., iff $\mathbf{Z} \setminus Z$ is not a separator for all $Z \in \mathbf{Z}$. This statement appears trivial, however, it is not, since there might be a separator from which removing any single element does not result in a separator while removing two elements together still yields a separator. But there is no such separator. Their result can easily be generalized to AGs or RCGs and then holds for any separator $\mathbf{Z}$, but the proof is lengthy, so we will only state it for separators $\mathbf{Z}$ with $\mathbf{M} \subseteq \mathbf{Z} \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$, which only requires a rather short proof.

Lemma 3.25. *A separator $\mathbf{Z}$ with $\mathbf{M} \subseteq \mathbf{Z} \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$ is $\mathbf{M}$-minimal if and only if, for every $Z \in \mathbf{Z} \setminus \mathbf{M}$, the set $\mathbf{Z} \setminus Z$ is not a separator.*

Proof. If there is an $Z \in \mathbf{Z} \setminus \mathbf{M}$ such that $\mathbf{Z} \setminus Z$ is a separator, $\mathbf{Z}$ is clearly not $\mathbf{M}$-minimal.

In the other direction, we have $\mathbf{Z} \setminus Z$ is not a separator for every $Z \in \mathbf{Z} \setminus \mathbf{M}$, so, for each $Z \in \mathbf{Z} \setminus \mathbf{M}$, there is a definite status path $\pi_Z : \mathbf{X} \overset{*}{\nsim} \mathbf{Y}$ that is not blocked by $\mathbf{Z} \setminus Z$, i.e., every non-collider is not in $\mathbf{Z} \setminus Z$ and every collider is in $An(\mathbf{Z} \setminus Z)$.

Assume $\mathbf{Z}$ is not $\mathbf{M}$-minimal, so there is a separator $\mathbf{Z}' \subset \mathbf{Z}$ with $\mathbf{M} \subseteq \mathbf{Z}'$. Let $Z \in \mathbf{Z} \setminus \mathbf{Z}'$. No non-collider of π_Z is in $\mathbf{Z}' \subseteq \mathbf{Z} \setminus Z$ and every collider is in $An(\mathbf{Z} \setminus Z) \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup (\mathbf{Z} \setminus Z)) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}')$, so π_Z is not blocked by $\mathbf{Z}'$ due to Lemma 3.12 and $\mathbf{Z}'$ is not a separator. Hence, $\mathbf{Z}$ is $\mathbf{M}$-minimal. □

Now we show that a separator relative to $(\mathbf{X}, \mathbf{Y})$ is minimal iff all its nodes are reachable from both $\mathbf{X}$ and $\mathbf{Y}$. It is easy to see that this is true in an undirected graph, and was used by [TPP98] and [ZLT14] to find minimal separators in the moral graph. But it also holds in an AG or RCG:

Lemma 3.26. *A separator $\mathbf{Z}$ with $\mathbf{M} \subseteq \mathbf{Z}$ is an $\mathbf{M}$-minimal separator iff, for every $Z \in \mathbf{Z} \setminus \mathbf{M}$, there exists an almost definite status path $\pi : \mathbf{X} \overset{+}{\sim} Z \overset{+}{\sim} \mathbf{Y}$ such that every node on π is in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$ and every non-collider is not in $\mathbf{Z} \setminus Z$.*

Proof. ($\Rightarrow$): Let $\mathbf{Z}$ be an $\mathbf{M}$-minimal separator. For every $Z \in \mathbf{Z} \setminus \mathbf{M}$, the set $\mathbf{Z} \setminus Z$ is not a separator due to Lemma 3.25, so there is an almost definite status path $\pi : \mathbf{X} \overset{+}{\sim} \mathbf{Y}$ open given $\mathbf{Z} \setminus Z$ and blocked by $\mathbf{Z}$, i.e., blocked by Z, so Z is a non-collider on π. Every other non-collider is not in $\mathbf{Z} \setminus Z$ or π was already blocked. From Corollary 3.23, we know $\mathbf{Z} \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$, so all nodes of π are in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$ due to Lemma 3.13.

($\Leftarrow$): Every node in $\mathbf{Z} \setminus \mathbf{M}$ is on an almost definite status path in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$, so $\mathbf{Z} \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$. Let $Z \in \mathbf{Z} \setminus \mathbf{M}$ and π be such a path. Every collider on π is in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup (\mathbf{Z} \setminus Z))$, so due to Lemma 3.12, $\mathbf{Z} \setminus Z$ does not separate $\mathbf{X}$ and $\mathbf{Y}$, and $\mathbf{Z}$ is $\mathbf{M}$-minimal. □

Corollary 3.27. *A separator $\mathbf{Z}$ with $\mathbf{M} \subseteq \mathbf{Z}$ is an $\mathbf{M}$-minimal separator iff, for every $Z \in \mathbf{Z} \setminus \mathbf{M}$, there exists an almost definite status walk $w : \mathbf{X} \overset{+}{\sim} Z \overset{+}{\sim} \mathbf{Y}$ such that every node on w is in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$ and every non-collider is not in $\mathbf{Z} \setminus Z$.*

Proof. Let $Z \in \mathbf{Z} \setminus \mathbf{M}$. If there exists such an almost definite status walk $w : \mathbf{X} \overset{+}{\sim} Z \overset{+}{\sim} \mathbf{Y}$, there exists a corresponding almost definite status path $\pi' : \mathbf{X} \overset{+}{\sim} \mathbf{Y}$ due to Lemma 3.11. Every non-collider on π' is a non-collider on w (see Lemma 3.11) and is not in $\mathbf{Z} \setminus Z$. Every collider is in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$, so due to Lemma 3.12, $\mathbf{Z} \setminus Z$ does not block π' and $\mathbf{Z}$ is only a separator if Z is a non-collider on π'. So Lemma 3.26 applies. □

Hence, the problems TESTMINSEP and FINDMINSEP can be solved efficiently by a reachability search that finds walks from $\mathbf{X}$ and $\mathbf{Y}$ to each node of $\mathbf{Z}$ containing only nodes of $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$ and no non-colliders in $\mathbf{Z}$. The resulting reachability rules are shown in Figure 3.4.

3.3.3 Testing Minimal Separators

Here we provide the function TESTMINSEP which tests if $\mathbf{Z}$ is an $\mathbf{M}$-minimal separator relative to $(\mathbf{X}, \mathbf{Y})$ in an AG or RCG $\mathcal{G}$ under the constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$:

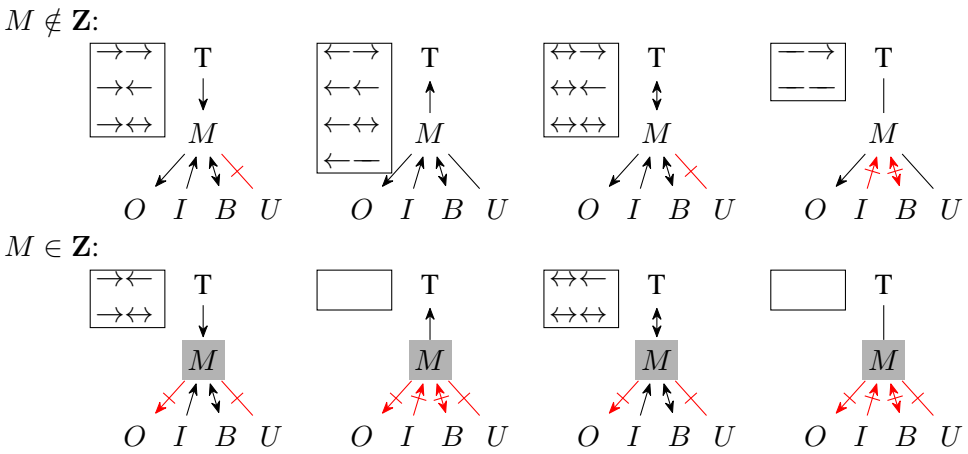


Figure 3.4: Expanded rules for Bayes-Ball in AGs modified to test or find $\mathbf{M}$-minimal separators. The figure is structured as Figure 3.3, but all nodes are assumed to be in $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$.

function TESTMINSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{M}, \mathbf{I}, \mathbf{R}$)
  if $\mathbf{I} \not\subseteq \mathbf{Z} \vee \mathbf{Z} \not\subseteq \mathbf{R}$ **then return** FALSE
  $\mathbf{A} := pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$
  if $\mathbf{Z} \not\subseteq \mathbf{A}$ **then return** FALSE
  $accept :=$ **function**$(e, U, f, V)\{$
    return $V \in \mathbf{A} \wedge$
      eUf is of almost definite status $\wedge$
      (U is a collider on $eUf \vee U \notin \mathbf{Z}$)
  $\}$
  $\mathbf{W}_X :=$ REACHABLE$(\mathcal{G}, \mathbf{X}, accept)$
  if $\mathbf{W}_X \cap \mathbf{Y} \neq \emptyset$ **then return** FALSE
  if $\mathbf{Z} \setminus \mathbf{M} \not\subseteq \mathbf{W}_X$ **then return** FALSE
  $\mathbf{W}_Y :=$ REACHABLE$(\mathcal{G}, \mathbf{Y}, accept)$
  if $\mathbf{Z} \setminus \mathbf{M} \not\subseteq \mathbf{W}_Y$ **then return** FALSE
  return TRUE

Analysis of the Algorithm. TESTMINSEP first checks the basic constraints $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ and the constraint $\mathbf{Z} \subseteq pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$ from Corollary 3.23. REACHABLE finds all nodes that are reachable from $\mathbf{X}$ ($\mathbf{Y}$) by almost definite status walks that only contain nodes of $\mathbf{A} = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$ and on which all non-colliders are not in $\mathbf{Z}$. From Lemma 3.12 and Lemma 3.13, it follows $\mathbf{Z}$ is a separator if and only if $\mathbf{W}_X$ contains no node of $\mathbf{Y}$ (alternatively and equivalently: $\mathbf{W}_Y$ contains no node of $\mathbf{X}$).

Then the algorithm returns TRUE iff $\mathbf{Z} \setminus \mathbf{M} \subseteq \mathbf{W}_X \cap \mathbf{W}_Y$, which means, for each $Z \in \mathbf{Z} \setminus \mathbf{M}$, there exist almost definite status walks $\mathbf{X} \overset{+}{\sim} Z$ and $Z \overset{+}{\sim} \mathbf{Y}$ through $\mathbf{A}$ on which each non-collider is not in $\mathbf{Z}$, i.e., all non-colliders of the combined walk $\mathbf{X} \overset{+}{\sim} Z \overset{+}{\sim} \mathbf{Y}$ are not in $\mathbf{Z} \setminus Z$ (under the assumption that $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are disjoint sets), and, according to Corollary 3.27, the set $\mathbf{Z}$ is $\mathbf{M}$-minimal.

The algorithm runs in $\mathcal{O}(n + m)$ as it only uses elementary set operations and calls to REACHABLE.

By setting the parameter $\mathbf{M} = \mathbf{I}$, the algorithm tests $\mathbf{I}$-minimality of $\mathbf{Z}$. By setting $\mathbf{M} = \emptyset$, the algorithm tests strong-minimality. □

Proposition 3.28. *Using the algorithm above, the task* TESTMINSEP*, both for testing* $\mathbf{I}$*-minimality and* $\emptyset$*-minimality, can be solved in time* $\mathcal{O}(n + m)$ *in AGs and RCGs.*

3.3.4 Finding Weakly-Minimal Separators

Next we give an algorithm to *find* an $\mathbf{I}$-minimal separator:

function FINDMINSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$)
  $\mathbf{A} := pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$
  $\mathbf{Z}' := \mathbf{R} \cap (\mathbf{A} \setminus (\mathbf{X} \cup \mathbf{Y}))$
  $accept :=$ **function**$(e, U, f, V)\{$
    return $V \in \mathbf{A} \wedge$
      eUf is of almost definite status $\wedge$
      (U is a collider on $eUf \vee U \notin \mathbf{Z}'$)
  $\}$
  $\mathbf{Z}'' := \mathbf{Z}' \cap$ REACHABLE$(\mathcal{G}, \mathbf{X}, accept) \cup \mathbf{I}$
  $accept :=$ **function**$(e, U, f, V)\{$
    return $V \in \mathbf{A} \wedge$
      eUf is of almost definite status $\wedge$
      (U is a collider on $eUf \vee U \notin \mathbf{Z}''$)
  $\}$
  $\mathbf{Z} := \mathbf{Z}'' \cap$ REACHABLE$(\mathcal{G}, \mathbf{Y}, accept) \cup \mathbf{I}$
  if not TESTSEP$(\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ **then return** $\perp$
  return $\mathbf{Z}$

Analysis of the Algorithm. FINDMINSEP always returns a set if $\mathbf{X}$ and $\mathbf{Y}$ are separable. $\mathbf{Z}'$ is a separator according to Lemma 3.18. $\mathbf{Z}'$ contains $\mathbf{I}$ since $\mathbf{I} \subseteq \mathbf{R}$ should be disjoint from $\mathbf{X} \cup \mathbf{Y}$. All nodes removed during the construction of $\mathbf{Z}''$ ($\mathbf{Z}$) are not reachable from $\mathbf{X}$ ($\mathbf{Y}$) by connected walks, so they cannot block a walk and the set remains a separator after their removal.

The set $\mathbf{Z}$ returned by the algorithm is a separator with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. All nodes in $\mathbf{Z} \setminus \mathbf{I}$ are reachable from $\mathbf{X}$ and $\mathbf{Y}$ by a walk through $pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})$ on which every non-collider is not in $\mathbf{Z}$, since all non-reachable nodes are removed and the reachable nodes are not affected by removal of non-reachable nodes. Thus, $\mathbf{Z}$ is $\mathbf{I}$-minimal according to Corollary 3.27.

The algorithm runs in $\mathcal{O}(n + m)$ as it only uses elementary set operations and calls to REACHABLE or TESTSEP. □

Proposition 3.29. *The algorithm above finds an* $\mathbf{I}$*-minimal separator* $\mathbf{Z}$ *with* $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ *in time* $\mathcal{O}(n + m)$ *in AGs and RCGs.*

Algorithm FINDMINSEP finds an $\mathbf{I}$-minimal separator. Note that setting the argument $\mathbf{I}$ of the algorithm to $\emptyset$ does not lead to a strongly-minimal separator $\mathbf{Z}$ that satisfies the constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ for a given non-empty set $\mathbf{I}$.

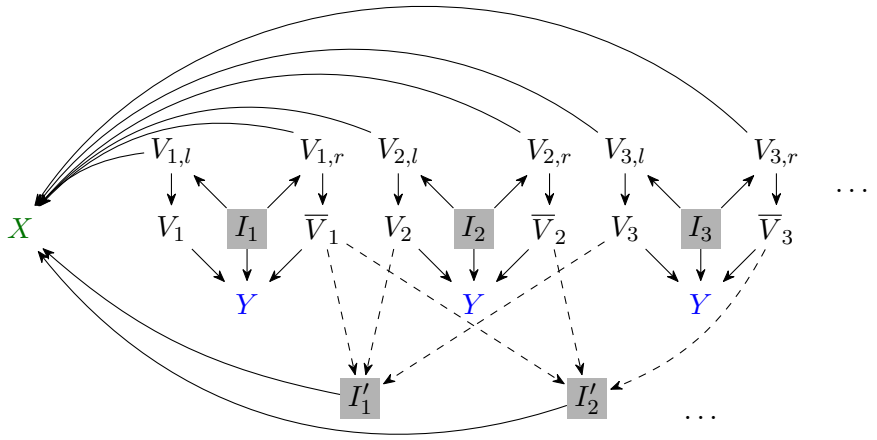


Figure 3.5: The graph used in the proof of Proposition 3.31, which represents the first three variables V_1, V_2, V_3, and two clauses $(\overline{V}_1 \vee V_2 \vee V_3)$ and $(\overline{V}_1 \vee \overline{V}_2 \vee \overline{V}_3)$. All shown Y nodes can be considered to be a single node Y, but we display them separately to reduce the number of overlapping edges in the figure.

Claim 3.30. *Given a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, a subgraph $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$, sets $\mathbf{V}' \subseteq \mathbf{V}$, $\mathbf{E}' \subseteq \mathbf{E}$, and $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}'$, then* FINDMINSEP$(\mathcal{G}', \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}) \subseteq$ FINDMINSEP$(\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R})$.

Proof. This follows from Claim 3.16. But note that when edges are removed from $\mathcal{G}$ to form $\mathcal{G}'$, the sets $\mathbf{A}$ and $\mathbf{Z}'$ calculated using $\mathcal{G}'$ can contain fewer nodes than the sets calculated using $\mathcal{G}$. Then function *accept* might return TRUE on some node $U \notin \mathbf{Z}'$ in $\mathcal{G}'$ on which it returns FALSE in $\mathcal{G}$, which would prevent the application of Claim 3.16. But since this only happens on nodes also removed from $\mathbf{A}$, these nodes are not visited anyways and do not affect the returned set $\mathbf{Z}$. □

3.3.5 The Hardness of Strong-Minimality

For most problems, it is easier to find a minimal solution than a minimum one, but for separators the opposite is true. If a strongly-minimum separator exists, it is also strongly-minimal. However, there is a gap where no strongly-minimum separator exists and the $\mathbf{I}$-minimum or $\mathbf{I}$-minimal separators are not strongly-minimal.

We now show that it is hard to find a strongly-minimal separator even for singletons X and Y in DAGs. Due to the equivalence between d-separation and vertex separators in the moral graph, this implies that it is also NP-hard to find strongly-minimal vertex separators in undirected graphs. Together with the characterizations of adjustment sets in the coming sections, it will follow that it is also NP-hard to find strongly-minimal adjustments in DAGs or ancestral graphs.

Proposition 3.31. *It is an NP-complete problem to decide whether, in a given DAG, there exists a strongly-minimal d-separating set $\mathbf{Z}$ containing $\mathbf{I}$.*

Proof. The problem is in NP since verifying $\mathbf{I} \subseteq \mathbf{Z}$ is trivial for a given $\mathbf{Z}$ and the strong-minimality can be efficiently tested by algorithm TESTMINSEP with parameter $\mathbf{I} = \emptyset$.

To show the NP-hardness, we take an instance of 3-SAT, a canonical NP-complete problem [GJ79a], and construct a DAG $\mathcal{G}$ and a set $\mathbf{I}$ such that a strongly-minimal d-separating set containing $\mathbf{I}$ exists in $\mathcal{G}$ if and only if the 3-SAT formula is satisfiable.

The 3-SAT instance consists of k variables $V_1, \ldots, V_k$, and ℓ clauses $C_i = (W_{i,1} \vee W_{i,2} \vee W_{i,3})$ of literals $W_{i,j} \in \{V_1, \ldots, V_k, \overline{V}_1, \ldots, \overline{V}_k\}$. It is NP-hard to decide whether there exists a Boolean assignment $\phi : \{V_1, \ldots V_k\} \to \{\text{TRUE}, \text{FALSE}\}$ that satisfies the formula $C_1 \wedge \ldots \wedge C_\ell$.

The basic idea of the construction is to ensure that any separator must contain some nodes to block a path between X and Y, while no separator can contain all those nodes, because all nodes together would block all paths from X to nodes in $\mathbf{I}$ making the separator not $\mathbf{I}$-minimal. Choosing a node for inclusion in the separator will correspond to choosing an assignment to a variable of the 3-SAT problem.

Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be defined as:

$$
\begin{array}{rcl}
\mathbf{V} & = & \{X, Y\} \cup \{V_{i,l}, V_{i,r}, V_i, \overline{V}_i, I_i \mid i \in \{1, \ldots, k\}\} \cup \{I'_i \mid i \in \{1, \ldots, \ell\}\}. \\
\mathbf{E} & = & \{X \leftarrow V_{i,l} \rightarrow V_i \rightarrow Y \mid i \in \{1, \ldots, k\}\} \\
 & \cup & \{X \leftarrow V_{i,r} \rightarrow \overline{V}_i \rightarrow Y \mid i \in \{1, \ldots, k\}\} \\
 & \cup & \{I_i \rightarrow V_{i,l},\ I_i \rightarrow V_{i,r},\ I_i \rightarrow Y \mid i \in \{1, \ldots, k\}\} \\
 & \cup & \{I'_i \rightarrow X \mid i \in \{1, \ldots, \ell\}\} \\
 & \cup & \{W_{i,j} \rightarrow I'_i \mid i \in \{1, \ldots, \ell\}, W_{i,j} \in C_i\}. \\
\mathbf{I} & = & \{I_i \mid i \in \{1, \ldots, k\}\} \cup \{I'_i \mid i \in \{1, \ldots, \ell\}\}.
\end{array}
$$

The resulting graph is shown in Figure 3.5. We identify the literal V_i ($\overline{V}_i$) in the formula with the node V_i ($\overline{V}_i$) in the graph. $V_{i,l}$ ($V_{i,r}$) is a left (right) node, , l in the index should not be confused with the number of clauses ℓ.

($\Leftarrow$): Let ϕ be a Boolean assignment that satisfies the formula. Let

$$
\begin{aligned}
\mathbf{Z} = \mathbf{I} \cup & \\
& \{V_i, V_{i,r} \mid i \in \{1, \ldots, k\},\ \phi(V_i) = \text{FALSE}\} \cup \\
& \{\overline{V}_i, V_{i,l} \mid i \in \{1, \ldots, k\},\ \phi(V_i) = \text{TRUE}\}.
\end{aligned}
$$

To show that $\mathbf{Z}$ d-separates X and Y, we begin a breadth first search at X and enumerate all reachable nodes. Immediately reachable are nodes I'_i, but they are in $\mathbf{I} \subseteq \mathbf{Z}$, so the path stops there. Each parent $V_{i,l}$ is reachable, but we have either $V_{i,l} \in \mathbf{Z}$ or $V_i \in \mathbf{Z}$, so the path stops either at the parent $V_{i,l}$, the next node V_i, or at $I_i \in \mathbf{Z}$.

$\mathbf{Z}$ is also strongly-minimal: If $V_i \in \mathbf{Z}$, then $V_{i,l} \notin \mathbf{Z}$ and $\mathbf{Z} \setminus \{V_i\}$ would not block the path $X \leftarrow V_{i,l} \rightarrow V_i \rightarrow Y$. A similar path would be opened by removing $\overline{V}_i$, $V_{i,l}$ or $V_{i,r}$. $\mathbf{Z} \setminus \{I_i\}$ is not a separator as it would not block either the path $X \leftarrow V_{i,l} \rightarrow I_i \rightarrow Y$ or $X \leftarrow V_{i,r} \rightarrow I_i \rightarrow Y$. If a clause C_i is satisfied by a literal V_j, $\mathbf{Z} \setminus \{I'_i\}$ is not a separator, as it would not block the path $X \leftarrow I'_i \leftarrow V_j \rightarrow Y$. Likewise, $X \leftarrow I'_i \leftarrow \overline{V}_j \rightarrow Y$ would be opened by removing I'_i if the clause C_i is satisfied by a literal $\overline{V}_j$.

Therefore, $\mathbf{Z}$ is a strongly-minimal separator.

($\Rightarrow$): Now we show that a strongly-minimal separator $\mathbf{Z}$ yields a satisfying assignment ϕ. For every i, the two paths $X \leftarrow V_{i,l} \rightarrow V_i \rightarrow Y$ and $X \leftarrow V_{i,r} \rightarrow \overline{V}_i \rightarrow Y$ need to be blocked by a node of $\{V_{i,l}, V_i\}$ and a node of $\{V_{i,r}, \overline{V}_i\}$. If neither V_i nor $\overline{V}_i$ are in $\mathbf{Z}$, both $V_{i,l}$ and $V_{i,r}$ must be in $\mathbf{Z}$, so I_i is not reachable from X, $\mathbf{Z} \setminus \{I_i\}$ is a separator, and $\mathbf{Z}$ is not strongly-minimal. Therefore, V_i or $\overline{V}_i$ is in $\mathbf{Z}$, and the following Boolean assignment ϕ to the variables is well-defined:

$$\phi(V_i) = \begin{cases} \text{TRUE} & V_i \notin \mathbf{Z}, \\ \text{FALSE} & \overline{V}_i \notin \mathbf{Z}, \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

Since $I'_i \in \mathbf{I}$ for all i, I'_i has to be reachable from Y, so there is an open path $I'_i \leftarrow V_j \rightarrow Y$ (or $I'_i \leftarrow \overline{V}_j \rightarrow Y$) and V_j (or $\overline{V}_j$) is not in $\mathbf{Z}$ for some j. This V_j (or $\overline{V}_j$) satisfies clause C_i according to the definition of ϕ. Hence, every clause and the formula are satisfiable. □

3.3.6 Augmentation and Moralization

As mentioned after Corollary 3.24, the moral graph of any minimal separator $\mathbf{Z}$ containing $\mathbf{I}$ is $(\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})})^a$, so it can be constructed without knowing $\mathbf{Z}$. This also holds for minimum separators since every minimum separator is minimal.

However, constructing the augmented graph according to its definition and searching a collider connected path for all pairs of nodes would have a suboptimal runtime of $\mathcal{O}(n^2m)$ in ancestral graphs. Therefore, we will describe an asymptotically optimal (linear time in output size) moralization algorithm for AGs (and RCGs). Although it was deemed too slow compared to the linear-time algorithms for testing and finding minimal separators, there might be other problems involving minimal separators which might have an easier solution in undirected graphs than in an RCG or AG, e.g., problems involving minimum separators.

Lemma 3.32 (Efficient AG moralization)**.** *Given an AG or RCG $\mathcal{G}$, the augmented graph $(\mathcal{G})^a$ can be computed in time $\mathcal{O}(n^2)$.*

Proof. The algorithm proceeds in four steps.

1. Start by setting $(\mathcal{G})^a$ to the skeleton of $\mathcal{G}$.
2. Partition $\mathcal{G}$ in all its maximal bidirectionally connected components.
3. For each pair (U, V) of nodes from the same component, add the edge $U \multimap V$ to $(\mathcal{G})^a$ if it did not exist already.
4. For each component, identify all its parents and link them all by undirected edges in $(\mathcal{G})^a$.

Now two nodes are adjacent in $(\mathcal{G})^a$ if and only if they are collider connected in $\mathcal{G}$. All four steps can be performed in time $\mathcal{O}(n^2)$. □

The constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ can be handled by removing the nodes of $\mathbf{I}$ and outside $\mathbf{R}$ from the graph as shown in Figure 3.6, similarly to the approach in [ADC96]. After the removal of nodes outside $\mathbf{R}$, their former neighbors need to be connected, which requires time $\mathcal{O}(|\mathbf{V} \setminus \mathbf{R}| \cdot |Ne(\mathbf{V} \setminus \mathbf{R})|^2) = \mathcal{O}(n^3)$, so this removal of nodes outside $\mathbf{R}$ only makes sense for slow algorithm like algorithm LISTMINSEP in Section 3.2.4.

3.3.7 Enumerating Weakly-Minimal Separators

Algorithm LISTSEP for non-minimal separators can enumerate a huge amount of uninteresting separators. For example, when it lists l separators in a graph $\mathcal{G}$ and we add k independent nodes without any edges to create a graph $\mathcal{G}' = \mathcal{G} \cup \{V'_1, \ldots, V'_k\}$, LISTSEP will list $2^k l$ separators

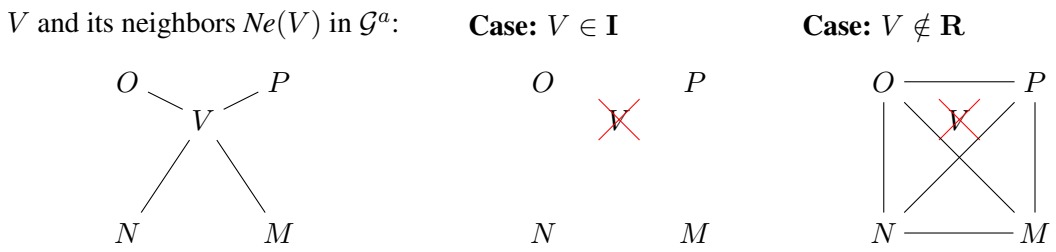


Figure 3.6: This figure explains the removal of nodes in $\mathbf{I}$ and outside of $\mathbf{R}$ from the augmented graph $(\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})})^a$. Shown is an example node V with all its neighbors in the augmented graph. In the case $V \in \mathbf{I}$, the node V blocks all paths through V, so the second graph obtained by removing V has no remaining edges. In the case $V \notin \mathbf{R}$, no path is blocked by V, so after removing the node, all its neighbors need to be linked to preserve the connectivities as shown in the third graph.

for $\mathcal{G}'$, each original separators combined with all 2^k subsets of $\{V'_1, \ldots, V'_k\}$ despite the nodes $\{V'_1, \ldots, V'_k\}$ being completely irrelevant for separation. Hence, it is usually more useful to enumerate only minimal separators in practice even though there are still exponential many minimal separators. For example, a minimal separator must contain either V or V' on every path $X \leftarrow V \leftarrow V' \leftarrow Y$ between X and Y, so a graph containing k such paths will have at least 2^k different minimal separators.

One could assume naively that replacing FINDSEP with FINDMINSEP in LISTSEP will enumerate minimal separators. However, this replacement does not change the output of the algorithm and it will list the very same separators, since a weakly-minimal separator exists if and if only any separator exists. Such a replacement might work with a function that finds a strongly-minimal separator, but since finding a strongly-minimal separator is NP-complete, this is not feasible.

So we will use Takata's algorithm [Tak10] that enumerates all separators of an undirected graph in $O(nm)$ delay time and $O(n)$ space after converting the AG or RCG to the moral graph. However, since the number of edges in the moral graph can be quadratic in the number of nodes, this leads to a delay time complexity of $O(n^3)$.

function LISTMINSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$)
  $\mathcal{G}' := \mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})}$
  $\mathcal{G}'^a := (\mathcal{G}_{pAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})})^a$
  Add a node X^m connected to all $\mathbf{X}$ nodes.
  Add a node Y^m connected to all $\mathbf{Y}$ nodes.
  Remove nodes of $\mathbf{I}$.
  Remove nodes of $\mathbf{V} \setminus \mathbf{R}$ connecting the neighbors of each removed node.
  Use the algorithm in [Tak10] to list all sets separating X^m and Y^m.

Analysis of the Algorithm. The correctness is shown by [TL11] for adjustment sets and generalizes directly to separators, because after moralization, both problems are equivalent to enumerating vertex cuts of an undirected graph. The handling of $\mathbf{I}$ is shown by [ADC96]. □

Proposition 3.33. *The task* LISTMINSEP *can be solved with polynomial delay $\mathcal{O}(n^3)$ in AGs and RCGs.*

3.3.8 Finding Minimum Separators

Finally, we consider the problem FINDMINCOSTSEP to find a minimum separator, for situations where even a minimal separator contains too many nodes. Thereby each node V is associated with a cost $w(V)$ given by a cost function $w : \mathbf{V} \to \mathbb{R}^+$ and the task is to find a set $\mathbf{Z}$ separating $\mathbf{X}$ and $\mathbf{Y}$ which minimizes the total cost $\sum_{Z \in \mathbf{Z}} w(Z)$ under the constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. In order to find a separator of minimum size, we can use a function $w(V) = 1 \; \forall V \in \mathbf{R}$ that assigns unit cost to each node. Alternatively, we might want to find a separator that minimizes the cost of measuring the variables in the separator or that minimizes the number of combinations that the values of these variables can take. When each node V corresponds to a random variable that can take k_V different values, there are $\prod_{V \in \mathbf{V}} k_V$ combinations, which can be minimized by a logarithmic cost function $w(V) = \log k_V \; \forall V \in \mathbf{R}$.

We again construct the augmented graph and can afterward solve the problem with any weighted min-cut algorithm.

function FINDMINCOSTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}, w$)
  $\mathcal{G}' := \mathcal{G}_{pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})}$
  $\mathcal{G}'^a := (\mathcal{G}_{pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^a$
  Add a node X^m connected to all nodes in $\mathbf{X}$, and a node Y^m connected to all nodes in $\mathbf{Y}$.
  Assign infinite cost to all nodes in $\mathbf{X} \cup \mathbf{Y} \cup (\mathbf{V} \setminus \mathbf{R})$ and cost $w(Z)$ to every other node Z.
  Remove all nodes of $\mathbf{I}$ from $\mathcal{G}'^a$.
  return a minimum vertex cut $\mathbf{Z}$ separating X^m and Y^m in the undirected graph.

Analysis of the Algorithm. The correctness follows from the fact that a minimum set is a minimal set and the minimum cut found in the ancestor moral graph is therefore the minimum m-separating set. The minimum cut can be found using a maximum flow algorithm in $\mathcal{O}(n^3)$ due to the well-known min-cut-max-flow theorem [Eve79, Chapter 6]. □

Proposition 3.34. *The algorithm above solves in time $\mathcal{O}(n^3)$ the task* FINDMINCOSTSEP *in case of* $\mathbf{I}$*-minimality in AGs and RCGs.*

The runtime primarily depends on the used min-cut/max-flow algorithm. Using a state-of-the-art max-flow algorithm by Orlin improves the runtime to $\mathcal{O}(nm_a)$ [Orl13], although this is not necessarily an improvement in our setting, because the augmented graph can have $m_a = \mathcal{O}(n^2)$ edges and then $\mathcal{O}(nm_a)$ and $\mathcal{O}(n^3)$ are the same. Faster max-flow algorithms are also known for specific graph classes or specific cost functions. In the special case of a unit cost function $w(V) = 1$ on undirected graphs, an $\mathcal{O}(m_a\sqrt{n})$ algorithm is known [Eve79], which is not directly applicable, since algorithm FINDMINCOSTSEP changes the nodes $\mathbf{X} \cup \mathbf{Y} \cup (\mathbf{V} \setminus \mathbf{R})$ to have infinite costs. However, we can apply the max-flow algorithm to a new graph containing only nodes in $\mathbf{R}' = (\mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y})) \cap pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$ by removing the nodes of $\mathbf{V} \setminus \mathbf{R}$ iteratively in $\mathcal{O}((n - |\mathbf{R}|)n^2)$ as shown in Figure 3.6 or by creating a new graph only containing those nodes in $\mathcal{O}(|\mathbf{R}'|m_a)$ as described in [TPP98], resulting in a total runtime of $\mathcal{O}(\min((n - |\mathbf{R}|)n^2, |\mathbf{R}'|m_a) + m_a\sqrt{|\mathbf{R}'|}) = \mathcal{O}(|\mathbf{R}'|m_a)$ for a unit cost function.

The set $\mathbf{Z}$ returned by algorithm FINDMINCOSTSEP is both weakly-minimum and strongly-minimum unless no strongly-minimum set $\mathbf{Z}$ exists under the given constraints $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. To see this assume $\mathbf{Z}$ is not strongly-minimum and there exists a strongly-minimum set $\mathbf{Z}'$ satisfying the constraint. Then $\sum_{Z \in \mathbf{Z}'} w(Z) < \sum_{Z \in \mathbf{Z}} w(Z)$. But $\mathbf{Z}'$ satisfies $\mathbf{I} \subseteq \mathbf{Z}'$, so $\mathbf{Z}$ was not $\mathbf{I}$-minimum, a contradiction.

All strongly-minimum sets for a graph have the same minimum sum $\sum_{Z \in \mathbf{Z}} w(Z)$, regardless of the constraint $\mathbf{I}$, so we can test if the set returned by FINDMINCOSTSEP is strongly-minimum by finding one strongly-minimum set $\mathbf{Z}'$ and testing whether $\sum_{Z \in \mathbf{Z}} w(Z) = \sum_{Z \in \mathbf{Z}'} w(Z)$. Such a strongly-minimum set $\mathbf{Z}'$ can be obtained by calling FINDMINCOSTSEP again with parameter $\mathbf{I} = \emptyset$. Although $\mathbf{Z}'$ might not fulfill the constraint $\mathbf{I} \subseteq \mathbf{Z}'$, it has the same required cost. Thus, we get the following:

Proposition 3.35. *Finding a separator $\mathbf{Z}$, with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$, that is strongly-minimum can be done in time $\mathcal{O}(n^3)$ in AGs and RCGs.*

The same arguments as above show that $\mathbf{Z}$ is $\mathbf{M}$-minimum for any $\mathbf{M} \subset \mathbf{I}$ if an $\mathbf{M}$-minimum separator satisfying the constraints exists.

This is a surprising result since finding objects of minimum costs or sizes is, typically, harder than constructing minimal objects because a minimum object is also minimal. The explanation is that in the instances where it is hard to decide whether a strongly-minimal separator exists or not, no strongly-minimum separator exists.

3.4 Relating Chain Graphs and Restricted Chain Graphs

All the above algorithms were specified to solve problems in restricted chain graphs, which covers the common classes of DAGs and CPDAGs, but these algorithms cannot be applied to general chain graphs where Lemma 3.1 does not hold. So when an arbitrary chain graph $\mathcal{G}$ is given, we first need to verify that $\mathcal{G}$ is either an RCG or can be converted to one.

Subsection 3.4.1 explains how to test whether $\mathcal{G}$ is an RCG, and Subsection 3.4.2 gives an algorithm to perform the conversion for CGs that are not RCGs.

3.4.1 Recognizing Restricted Chain Graphs

Let us first mention a clever way to recognized CPDAGs. Because every class of Markov equivalent DAGs is represented by a unique CPDAG, it is possible to test whether a given chain graph $\mathcal{G}$ is a CPDAG by finding one consistent DAG extension $\mathcal{D}$ of $\mathcal{G}$, generating the CPDAG $\mathcal{G}'$ for $\mathcal{D}$ and comparing the resulting graph $\mathcal{G}'$ with $\mathcal{G}$. Graph $\mathcal{G}$ is a CPDAG if and only if $\mathcal{G} = \mathcal{G}'$. Using the CG-to-DAG conversion algorithm of [AMP97] and the DAG-to-CPDAG conversion of [Chi95], this can be done in time $\mathcal{O}(m \log n)$. Alternatively, if the degree of the graph is bounded by a constant k, the algorithm of [Chi02] decreases the running time to $\mathcal{O}((n+m)k^2)$.

However, to recognize RCGs such an approach does not work and one needs to test the conditions of an RCG directly. The first property – the chordality of components – can be tested with a lexicographic breadth-first search in linear time [RTL76]. A naive test of the second condition, that $A \to B \multimap C$ does not exist as an induced subgraph, is possible in time $\mathcal{O}(nm)$. But there is also a more sophisticated method:

Let D, U, and M be three adjacency matrices corresponding to directed, undirected, resp. missing edges:

$$D[i,j] = \begin{cases} 1 & \text{if } i \to j \in \mathbf{E}, \\ 0 & \text{otherwise.} \end{cases}$$

$$U[i,j] = \begin{cases} 1 & \text{if } i \multimap j \in \mathbf{E}, \\ 0 & \text{otherwise.} \end{cases}$$

$$M[i,j] = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are unconnected,} \\ 0 & \text{otherwise.} \end{cases}$$

The trace of the product $Tr[D \cdot U \cdot M]$ is zero if and only if the second condition above is satisfied by the graph, since it corresponds to cycles $i \to j \text{ — } k$ -no-edge- i. Thus, this second condition can be verified in time $\mathcal{O}(n^\alpha)$, with $\alpha < 2.373$ [LG14], using a fast matrix multiplication algorithm. This dominates the time complexity of the whole recognition algorithm.

3.4.2 Reducing a Chain Graph to a Restricted Chain Graph

If a given chain graph $\mathcal{G}$ is not a restricted chain graph, our algorithms can only be used after $\mathcal{G}$ has been converted to an RCG. The idea is simple: when $\mathcal{G}$ contains an induced subgraph $A \to B \text{ — } C$, no represented DAG can contain $A \to B \leftarrow C$, since that would be an additional v-structure. Hence, replacing every induced subgraph $A \to B \text{ — } C$ with $A \to B \to C$ either yields a restricted chain graph or $\mathcal{G}$ represents no causal model, that is, $CE(\mathcal{G}) = \emptyset$. The proof requires some lemmas:

Lemma 3.36. *Let $\mathcal{G}$ be a chain graph and let $\mathcal{G}_r$ be obtained from $\mathcal{G}$ after a single application of the rule $A \to B \text{ — } C \;\Rightarrow\; A \to B \to C$. If $\mathcal{G}$ and $\mathcal{G}_r$ have the same v-structures, then $CE(\mathcal{G}) = CE(\mathcal{G}_r)$; otherwise, if $\mathcal{G}$ and $\mathcal{G}_r$ do not have the same v-structures, then $CE(\mathcal{G}) = \emptyset$.*

Proof. Every DAG $\mathcal{D} \in CE(\mathcal{G})$ has the same v-structures as $\mathcal{G}$, so if $A \to B \text{ — } C$ occurs in $\mathcal{G}$, $\mathcal{D}$ cannot contain $A \to B \leftarrow C$. Thus, $\mathcal{D}$ must contain $A \to B \to C$. If the replacement has changed the v-structures in $\mathcal{G}_r$, $\mathcal{D}$ cannot have the same v-structures as $\mathcal{G}$, and $CE(\mathcal{G}) = \emptyset$. If the replacement has not changed them and $CE(\mathcal{G}) \neq \emptyset$, then $CE(\mathcal{G}) = CE(\mathcal{G}_r)$ because every $\mathcal{D} \in CE(\mathcal{G})$ contains $B \to C$.

If $CE(\mathcal{G}) = \emptyset$ then, unless a new v-structure is created, $CE(\mathcal{G}_r) = \emptyset$ since replacing an undirected edge with a directed one cannot add a DAG to $CE(\mathcal{G})$, . □

Lemma 3.37. *Let $\mathcal{G}$ be a chain graph and let $\mathcal{G}_r^*$ be the closure of $\mathcal{G}$ under the rule $A \to B \text{ — } C \;\Rightarrow\; A \to B \to C$. Then $\mathcal{G}_r^*$ is a chain graph.*

Proof. If $\mathcal{G}_r^*$ is not a chain graph, the change of an edge $B \text{ — } C$ has introduced a semi-directed cycle. This means there is an undirected cycle containing $B \text{ — } C$ in $\mathcal{G}$. Let $w : V_1 \text{ — } V_2 \stackrel{+}{-} V_k \text{ — } V_1$ with $V_1 = B$, $V_2 = C$ be this cycle. The node A cannot be adjacent to any V_i with an undirected edge or an edge pointing towards A, otherwise $A \to V_1 \stackrel{+}{-} V_i \to A$ or $A \to V_1 \stackrel{+}{-} V_i \text{ — } A$ would have been a semi-directed cycle in $\mathcal{G}$. If V_k is not adjacent to A, there is the induced subgraph $A \to V_1 \text{ — } V_k$ and w would not become a semi-directed cycle. Thus, there is an arrow $A \to V_k$ in $\mathcal{G}$, and if A is not adjacent to V_{k-1}, the induced subgraph $A \to V_k \text{ — } V_{k-1}$ would have been replaced by $A \to V_k \to V_{k-1}$ in $\mathcal{G}$, preventing the semi-directed cycle. Inductively, it follows: if there is a V_i not adjacent to A, w will not become a semi-directed cycle. We know that A is not adjacent to $B = V_2$, and thus $\mathcal{G}_r^*$ is a chain graph. □

Lemma 3.38. *Every chain component of a chain graph $\mathcal{G}$ with $CE(\mathcal{G}) \neq \emptyset$ is chordal.*

Proof. This follows from Proposition 4.2 with Remark 4.2 in [AMP97]. □

Now we give the algorithm:

function CONVERT-CG-TO-RCG($\mathcal{G}$)
  while there exists an induced subgraph $\mathcal{H} : A \to B - C$ in $\mathcal{G}$ **do**
    if $Pa(C) \setminus Ne(B) \neq \emptyset$ **then**
      return $\perp$
    Replace $\mathcal{H}$ with $A \to B \to C$ in $\mathcal{G}$
  for every chain component $\mathcal{C}$ of $\mathcal{G}$ **do**
    if $\mathcal{C}$ is not chordal **then**
      return $\perp$
  return the resulting graph $\mathcal{G}$

Analysis of the Algorithm. It follows from Lemma 3.36, Lemma 3.37, and Lemma 3.38 that the algorithm does not abort with $\perp$, if $CE(\mathcal{G}) \neq \emptyset$, and that $CE(\mathcal{R}) = CE(\mathcal{G})$. It is also guaranteed that $\mathcal{R}$ is an RCG.

The runtime $\mathcal{O}(\deg(\mathcal{G})^2 m) \leq \mathcal{O}(n^4)$ follows from the straightforward implementation of the algorithm. Chordality can be tested in linear time by lexicographic breadth-first search [RTL76]. □

Proposition 3.39. *Let $\mathcal{G}$ be a chain graph. If $CE(\mathcal{G}) \neq \emptyset$, then algorithm* CONVERT-CG-TO-RCG *generates an RCG $\mathcal{R}$ with $CE(\mathcal{G}) = CE(\mathcal{R})$. Otherwise, the algorithm returns $\perp$. The runtime is $\mathcal{O}(\deg(\mathcal{G})^2 m) \leq \mathcal{O}(n^4)$.*

There is no need to consider specific DAG-to-RCG or RCG-to-DAG conversion algorithms since every DAG is already an RCG and every RCG is a chain graph, so the algorithms cited above can be used for latter task. For the task RCG-to-CPDAG, the usual DAG-to-CPDAG algorithms can be used because they always generate and continue on RCGs in intermediate steps.

3.5 Discussion

We have solved the problems of Table 3.1 for AGs and RCGs, providing efficient algorithms to find separators as well as minimal and minimum separators. We have generalized existing algorithms and improved their performance. Many tasks can only be solved by our algorithms, and on tasks that can be solved by both existing algorithms and our algorithms, our algorithms have the same or better runtime, so our algorithms are preferable in every situation.

The algorithms are implemented in the open-source software DAGitty [Tex+16].

Like [ADC96; TPP98], we have described an algorithm that finds a minimum separator using a network flow in the moral graph in runtime $\mathcal{O}(n^3)$. It is probably possible to improve this runtime to $\mathcal{O}(nm)$ by searching the paths of the network flow directly in the causal graph using the same technique we have developed to improve the runtime of the algorithms for minimal separators. The delay complexity of enumerating minimal separators might be improved similarly.

A problem that we have ignored is to enumerate all minimum separators. One approach to solve it could be to enumerate all minimal separators and only output those that are minimum, although this might lead to a high complexity when many minimal sets need to be discarded as non-minimum. One could prune the search tree by aborting branches where the cost of the smallest possible minimal set on that branch is already higher than the cost of an earlier found minimum separator.

4 Identification via Covariate Adjustment

Covariate adjustment is one of the most widely used techniques to estimate causal effects from observational data. In this chapter, we leverage the algorithmic framework of Chapter 3 together with constructive, sound, and complete criteria for covariate adjustment to solve all problems listed in Table 3.1 for adjustment sets rather than separating sets in the same asymptotic time (see Table 4.1).

Given nodes sets $\mathbf{X}$ and $\mathbf{Y}$ in a causal graph, we want to test, find, or enumerate adjustment sets $\mathbf{Z}$ that can identify the causal effect of $\mathbf{X}$ on $\mathbf{Y}$. Our constructive criteria reduce tasks involving adjustment sets to separating sets in a subgraph of the original causal graph, which allows one to solve all tasks by calling a separation algorithm on the parameters obtained by the reduction without further modifying the separation algorithm. The reduction removes the first edge of certain causal paths from $\mathbf{X}$ to $\mathbf{Y}$, so after the removal, all remaining paths between $\mathbf{X}$ and $\mathbf{Y}$ are biasing paths, which need to be blocked by every adjustment set $\mathbf{Z}$. Furthermore, we characterize a set of forbidden nodes that cannot be adjusted, so the constraint $\mathbf{R}$ is applied to only return sets $\mathbf{Z}$ not containing these forbidden nodes.

Thus, the algorithms can search for adjustment sets $\mathbf{Z}$ that are minimal, minimum, or are bounded by constraining sets $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$.

The constraints now have an actual real-life motivation because not all covariates are equally well suited for adjustment. Some variables might be more difficult to measure than other variables even if both are equally observed. Measuring certain variables could also be more expensive or be affected by substantial measurement error. Thus, researchers might prefer adjustment sets that do not contain certain variables, are minimal, or minimize a user-supplied cost function. On the other hand, it might be necessary to include some additional variables in $\mathbf{I}$ to increase the precision of the estimation even if the causal effect is theoretically identifiable without those variables.

For example, in a biological study where one can either adjust for the genes or the resulting phenotype of an organism, DNA sequencing would be more expensive but give more accurate results. Which variables the biologist will prefer, depends on the funding available for their study or if the DNA has already been sequenced for another study, circumstances not modeled in the graph and thus unknown to our algorithms.

The constraints choose a subclass of the class of all valid adjustment sets for a given graph. Some variables cannot be used for adjustment at all, because the variables would bias the estimation. These forbidden variables are not excluded by the constraints of Table 4.1, rather they are excluded by the requirement to return adjustment sets. Nevertheless, after reducing a problem involving an adjusting set to the corresponding problem involving separating sets, the constraints are modified to exclude the forbidden variables from the separating set.

		Runtime
Verification: For given $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and constraint $\mathbf{I}$, decide whether ...		
TESTADJ	$\mathbf{Z}$ is an adjustment for $(\mathbf{X}, \mathbf{Y})$	$\mathcal{O}(n+m)$
TESTMINADJ	$\mathbf{Z} \supseteq \mathbf{I}$ is an adjustment for $(\mathbf{X}, \mathbf{Y})$ and $\mathbf{Z}$ is ...	
	$\mathbf{I}$-minimal	$\mathcal{O}(n+m)$
	strongly-minimal	$\mathcal{O}(n+m)$
Construction: For given $\mathbf{X}, \mathbf{Y}$ and constraints $\mathbf{I}, \mathbf{R}$, output an ...		
FINDADJ	adjustment $\mathbf{Z}$ for $(\mathbf{X}, \mathbf{Y})$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$\mathcal{O}(n+m)$
FINDMINADJ	adjustment $\mathbf{Z}$ for $(\mathbf{X}, \mathbf{Y})$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...	
	$\mathbf{I}$-minimal	$\mathcal{O}(n+m)$
	strongly-minimal	NP-complete
FINDMINCOSTADJ	adjustment $\mathbf{Z}$ for $(\mathbf{X}, \mathbf{Y})$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...	
	$\mathbf{I}$-minimum	$\mathcal{O}(n^3)$
	strongly-minimum	$\mathcal{O}(n^3)$
Enumeration: For given $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$, enumerate all ...		Delay
LISTADJ	adjustments $\mathbf{Z}$ for $(\mathbf{X}, \mathbf{Y})$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$\mathcal{O}(n(n+m))$
LISTMINADJ	$\mathbf{I}$-minimal adjustments $\mathbf{Z}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$\mathcal{O}(n^3)$

Table 4.1: Algorithmic tasks related to adjustment, similar to the definitions of Table 3.1. The runtimes are shown for DAGs, RCGs, and adjustment amenable MAGs. Other MAGs first need to be tested for adjustment amenability in $\mathcal{O}(n(n+m))$ time. Chain graphs require a conversion to an RCG in $\mathcal{O}(n^4)$ time.

Scientific Contribution. We have generalized the sound and complete adjustment criterion by [SVR10] from DAGs to MAGs [ZLT14; ZLT19] and to RCGs [ZL16b]. We then apply our criteria to continue the work of [TL11], and give the first efficient sound and complete algorithms for adjustment. We summarize our results in Table 4.1

4.1 Preliminaries

Let us recall the definition of adjustment, the full version of Equation 2.3:

Definition 4.1 (Adjustment [Pea09])**.** *Given a causal graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ and pairwise disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$, the set $\mathbf{Z}$ is called* covariate adjustment for estimating the causal effect of $\mathbf{X}$ on $\mathbf{Y}$*, or simply* adjustment *(set), if for every distribution P compatible with $\mathcal{G}$ we have*

adjustment set

$$P(\mathbf{y} \mid do(\mathbf{x})) = \begin{cases} P(\mathbf{y} \mid \mathbf{x}) & \text{if } \mathbf{Z} = \emptyset, \\ \sum_{\mathbf{z}} P(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) P(\mathbf{z}) & \text{otherwise.} \end{cases} \tag{4.1}$$

A probability distribution P is compatible with a DAG $\mathcal{D}$ if P factorizes according to $\mathcal{D}$. P is compatible with a CG $\mathcal{C}$ if it is compatible with a (every) DAG $\mathcal{D} \in [\mathcal{C}]$. P is compatible with a MAG $\mathcal{M}$ if it is the marginal of a distribution P' that is compatible with a DAG $\mathcal{D}'$ with $\mathcal{D}'[^{\mathbf{L}}_{\emptyset} = \mathcal{G}$ for any set of variables $\mathbf{L}$.

In practice, Definition 4.1 cannot be used to actually find an adjustment set as there are infinitely many probability distributions that are compatible to a certain graph.

4.2 Adjustment in DAGs

For DAGs, [SVR10] have discovered a characterization of adjustment sets in graphical terms:

Definition 4.2 (Adjustment criterion (AC) [SVR10])**.** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. The set $\mathbf{Z}$ satisfies the* adjustment criterion *relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$ if* *adjustment criterion*

(a) *no element in $\mathbf{Z}$ is a descendant in $\mathcal{G}_{\overline{\mathbf{X}}}$ of any $W \in \mathbf{V} \setminus \mathbf{X}$ which lies on a proper causal path from $\mathbf{X}$ to $\mathbf{Y}$ and*

(b) *all proper non-causal paths in $\mathcal{G}$ from $\mathbf{X}$ to $\mathbf{Y}$ are blocked by $\mathbf{Z}$.*

This criterion removes the need to consider infinitely many probability distributions, but the direct application of it still has to handle exponential many non-causal paths and check all of them to see if they are blocked.

We address this problem below by presenting a constructive adjustment criterion that reduces non-causal paths to separation, which allows us to directly apply all algorithms from Chapter 3 for adjustment.

But first we introduce a simpler notation for non-start nodes on a proper causal path from $\mathbf{X}$ to $\mathbf{Y}$:

$$\begin{aligned} \mathit{PCP}(\mathbf{X}, \mathbf{Y}) &= \{W \in \mathbf{V} \setminus \mathbf{X} \mid W \text{ lies on a proper causal path from } \mathbf{X} \text{ to } \mathbf{Y}\} \\ &= (\mathit{De}_{\overline{\mathbf{X}}}(\mathbf{X}) \setminus \mathbf{X}) \cap \mathit{An}_{\underline{\mathbf{X}}}(\mathbf{Y}). \end{aligned}$$

Then condition (a) in Definition 4.2 can be written as $\mathbf{Z} \subseteq \mathbf{V} \setminus \mathit{De}_{\overline{\mathbf{X}}}(\mathit{PCP}(\mathbf{X}, \mathbf{Y}))$.

We wonder, why did [SVR10] choose to use $\mathcal{G}_{\overline{\mathbf{X}}}$ in Definition 4.2? Is it necessary to construct this graph or can we use descendants in $\mathcal{G}$?

Lemma 4.3. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an ancestral graph, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables such that $\mathbf{Z}$ blocks all non-causal paths from $\mathbf{X}$ to $\mathbf{Y}$. Let $\mathbf{A} \subseteq \mathbf{X} \cup \mathbf{Y}$ and $\mathbf{B} \subseteq \mathbf{X}$.*

Then $\mathbf{Z}$ contains no descendant of $\mathit{PCP}(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$ if and only if $\mathbf{Z}$ contains no descendant of $\mathit{PCP}(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}_{\overline{\mathbf{A}}\underline{\mathbf{B}}}$.

Proof. ($\Leftarrow$): In $\mathcal{G}_{\overline{\mathbf{A}}\underline{\mathbf{B}}}$, there can only be fewer descendants since there are fewer edges.

($\Rightarrow$): Assume the statement is not true and there exist a $Z \in \mathbf{Z}$ and $W \in \mathit{PCP}(\mathbf{X}, \mathbf{Y})$ such that Z is only a descendant of W in $\mathcal{G}$. Then there are causal paths $\pi_{XY} = \mathbf{X} \dot{\to} W \overset{*}{\to} \mathbf{Y}$ and $\pi_{XZ} = \mathbf{X} \dot{\to} W \dot{\to} Z$. We have $W \neq Z$ since all nodes are their own descendants in any graph. We can assume the nodes were choosen such that the length of the subpath $\pi_{XZ}[W \overset{*}{\sim} Z]$ between W and Z is minimal. This implies that π_{XZ} is not blocked by $\mathbf{Z} \setminus Z$, because otherwise using the blocking node as Z would lead to shorter paths.

The subpath $\pi_{XZ}[W \overset{*}{\sim} Z]$ between W and Z intersects $\mathbf{A} \cup \mathbf{B}$ as it does not exist in $\mathcal{G}_{\overline{\mathbf{A}}\underline{\mathbf{B}}}$. Let $A \in \mathbf{A} \cup \mathbf{B} \subseteq \mathbf{X} \cup \mathbf{Y}$ be the node of this intersection closest to W; unless the closest node is $W = Y \in \mathbf{A}$, then let A be the second closest, because the case $Y \in \mathbf{A}$ only removes incoming edges and does not change a path $Y \overset{*}{\to} Z$.

If $A \in \mathbf{X}$, $A \neq W$ (or π_{XY} would not be proper) and the path $A \dot{\leftarrow} W \overset{*}{\to} Y$ would be a unblocked non-causal path. Otherwise, we have $A \in \mathbf{Y}$, so $A \in \mathit{PCP}(\mathbf{X}, \mathbf{Y})$ and the path from A to Z is shorter than the path from W to Z, so we could use A as W, which contradicts the choice of W. □

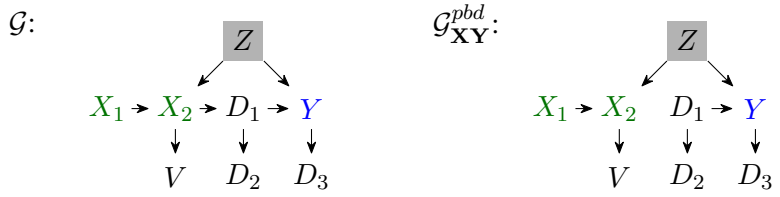


Figure 4.1: A DAG that permits exactly two adjustment sets for estimating the causal effect of $\mathbf{X} = \{X_1, X_2\}$ on $\mathbf{Y} = \{Y\}$: $\mathbf{Z} = \{Z\}$ and $\mathbf{Z}' = \{Z, V\}$. The set $\mathbf{V} \setminus De(PCP(\mathbf{X}, \mathbf{Y})) = \{X_1, X_2, Z, V\}$. So, every adjustment is a subset of $\{Z, V\}$. The nodes D_1, D_2, D_3 are not allowed in any adjustment as they are not in $\{Z, V\}$ – the set of descendants of a non-$\mathbf{X}$ node on the (only) proper causal path $X_2 \to D_1 \to Y$. Moreover, every adjustment must contain the variable Z to block the path between X_2 and Y in $\mathcal{G}^{pbd}_{\mathbf{XY}}$.

Hence, from now on we will use the condition $\mathbf{Z} \subseteq \mathbf{V} \setminus De(PCP(\mathbf{X}, \mathbf{Y}))$ rather than $\mathbf{Z} \subseteq \mathbf{V} \setminus De_{\overline{\mathbf{X}}}(PCP(\mathbf{X}, \mathbf{Y}))$ as condition (a).

To obtain a more efficient condition (b), we introduce the proper back-door graph:

proper back-door graph

Definition 4.4 (Proper back-door graph)**.** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ be disjoint subsets of variables. The* proper back-door graph*, denoted as $\mathcal{G}^{pbd}_{\mathbf{XY}}$, is obtained from $\mathcal{G}$ by removing the first edge of every proper causal path from $\mathbf{X}$ to $\mathbf{Y}$.*

Using the set $PCP(\mathbf{X}, \mathbf{Y})$ defined above the proper back-door graph can be specified as

$$\mathcal{G}^{pbd}_{\mathbf{XY}} = (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \to PCP(\mathbf{X}, \mathbf{Y}))),$$

i.e., all edges $X \to D$ with $D \in PCP(\mathbf{X}, \mathbf{Y})$ are removed. Thus, $\mathcal{G}^{pbd}_{\mathbf{XY}}$ can be constructed in linear time.

Note the difference between the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ and the famous back-door graph $\mathcal{G}_{\underline{\mathbf{X}}}$ of [Pea09]: in $\mathcal{G}_{\underline{\mathbf{X}}}$ *all* edges leaving $\mathbf{X}$ are removed while in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ only those that lie on a proper causal path (see Figure 4.8 for an example).

Now we propose the following adjustment criterion. For short, we will sometimes denote the set $De(PCP(\mathbf{X}, \mathbf{Y}))$ as $Dpcp(\mathbf{X}, \mathbf{Y})$.

constructive back-door criterion

Definition 4.5 (Constructive back-door criterion for DAGs (CBC))**.** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. The set $\mathbf{Z}$ satisfies the* constructive back-door criterion *relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$ if*

(a) $\mathbf{Z} \subseteq \mathbf{V} \setminus De(PCP(\mathbf{X}, \mathbf{Y}))$ *and*

(b) $\mathbf{Z}$ *d-separates* $\mathbf{X}$ *and* $\mathbf{Y}$ *in the proper back-door graph* $\mathcal{G}^{pbd}_{\mathbf{XY}}$.

Figure 4.1 shows how the constructive back-door criterion can be applied to find an adjustment set in an example DAG.

Theorem 4.6. *The constructive back-door criterion (CBC) is equivalent to the adjustment criterion (AC).*

Proof. Assume conditions (a) and (b) of the adjustment criterion AC hold. Due to Lemma 4.3, it is sufficient to show that condition (b) of the constructive back-door criterion is satisfied. Let π be any proper path from $\mathbf{X}$ to $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Because $\mathcal{G}^{pbd}_{\mathbf{XY}}$ does not contain causal paths from $\mathbf{X}$ to $\mathbf{Y}$, π is not causal and has to be blocked by $\mathbf{Z}$ in $\mathcal{G}$ by the assumption. Since removing edges cannot open paths, π is blocked by $\mathbf{Z}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ as well.

Now we show that (a) and (b) of the constructive back-door criterion CBC together imply (b) of the adjustment criterion AC. If that were not the case, then there could exist a proper non-causal path π from $\mathbf{X}$ to $\mathbf{Y}$ that is blocked in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ but open in $\mathcal{G}$. There can be two reasons why π is blocked in $\mathcal{G}^{pbd}_{\mathbf{XY}}$: (1) The path starts with an edge $X \to D$ that does not exist in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Then we have $D \in PCP(\mathbf{X}, \mathbf{Y})$. For π to be non-causal, it would have to contain a collider $C \in An(\mathbf{Z}) \cap De(D) \subseteq An(\mathbf{Z}) \cap Dpcp(\mathbf{X}, \mathbf{Y})$. But because of CBC (a), $An(\mathbf{Z}) \cap Dpcp(\mathbf{X}, \mathbf{Y})$ is empty. (2) A collider C on π is an ancestor of $\mathbf{Z}$ in $\mathcal{G}$, but not in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Then there must be a directed path from C to $\mathbf{Z}$ via an edge $X \to D$ with $D \in An(\mathbf{Z}) \cap PCP(\mathbf{X}, \mathbf{Y})$, contradicting CBC (a). So (b) of the AC holds, and so (a) holds as well due to Lemma 4.3. □

4.3 Adjustment in MAGs

Next we generalize our complete constructive criterion for identifying adjustment sets from DAGs to MAGs. Two examples may illustrate why this generalization is not trivial.

First, take $\mathcal{G}_1 = X \to Y$. If $\mathcal{G}_1$ is interpreted as a DAG, then the empty set is valid for adjustment. If $\mathcal{G}_1$ is, however, taken as a MAG, then there exists no valid adjustment set as $\mathcal{G}_1$ represents among others the DAG $U \to X \to Y$ where U is an unobserved confounder.

Second, take $\mathcal{G}_2 = A \to X \to Y$. In that case, the empty set is an adjustment set relative to (X, Y) regardless of whether $\mathcal{G}_2$ is interpreted as a DAG or a MAG. $\mathcal{G}_2$ does not represent a DAG like $\mathcal{D} = A \leftarrow U \to X \to Y$, because blocking the path $A \to X \to Y$ will open the path $A \to X \leftarrow U \to Y$. So A and Y cannot be d-separated in $\mathcal{D}$ and need to be adjacent in the representing MAG, i.e., the only MAG representing $\mathcal{D}$ is $U \to X \to Y$

The main topic of this subsection is to study the cases in which hidden confounders might invalidate an adjustment set and in which cases they might not.

Lemma 4.7 (Preservation of separating sets [RS02])**.** *Set* $\mathbf{Z}$ *m-separates* $\mathbf{X}, \mathbf{Y}$ *in* $\mathcal{G}[^{\mathbf{S}}_{\mathbf{L}}$ *if and only if* $\mathbf{Z} \cup \mathbf{S}$ *m-separates* $\mathbf{X}, \mathbf{Y}$ *in* $\mathcal{G}$*.*

Selection bias (i.e., $\mathbf{S} \neq \emptyset$) substantially complicates adjustment, and in fact nonparametric causal inference in general [Zha08][1]. Due to these limitations, we restrict ourselves to the case $\mathbf{S} = \emptyset$ in the rest of this section. Note, however, that recovery from selection bias is sometimes possible with additional population data, and graphical conditions exist to identify such cases [BTP14].

We now extend the concept of adjustment to MAGs in the usual way [MC15].

Definition 4.8 (Adjustment in MAGs)**.** *Given a MAG* $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ *and two variable sets* $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$*, the set* $\mathbf{Z} \subseteq \mathbf{V}$ *is an adjustment set for* $(\mathbf{X}, \mathbf{Y})$ *in* $\mathcal{M}$ *if for all DAGs* $\mathcal{G} = (\mathbf{V}', \mathbf{E}')$ *for which* $\mathcal{G}[^{\emptyset}_{\mathbf{L}} = \mathcal{M}$ *with* $\mathbf{L} = \mathbf{V}' \setminus \mathbf{V}$ *the set* $\mathbf{Z}$ *is an adjustment set for* $(\mathbf{X}, \mathbf{Y})$ *in* $\mathcal{G}$*.*

[1] A counterexample is the graph $A \leftarrow X \to Y$, where we can safely assume that A is the ancestor of a selection variable. A sufficient and necessary condition to recover a distribution $P(\mathbf{y} \mid \mathbf{x})$ from a distribution $P(\mathbf{y} \mid \mathbf{x}, \mathbf{s})$ under selection bias is $\mathbf{Y} \perp\!\!\!\perp \mathbf{S} \mid \mathbf{X}$ [BTP14], which is so restrictive that most statisticians would probably not even speak of "selection bias" anymore in such a case.

This definition is equivalent to requiring that $P(\mathbf{y} \mid do(\mathbf{x}))$ is equal to $\sum_{\mathbf{z}} P(\mathbf{y} \mid \mathbf{x}, \mathbf{z})P(\mathbf{z})$ for every probability distribution $P(\mathbf{v}')$ compatible with a DAG $\mathcal{G} = (\mathbf{V}', \mathbf{E}')$ for which $\mathcal{G}[^{\emptyset}_{\mathbf{L}} = \mathcal{M}$ with $\mathbf{L} = \mathbf{V}' \setminus \mathbf{V}$. If one was to extend the definition to include selection bias $\mathbf{S}$, one would need to give a requirement that holds for all DAGs $\mathcal{G} = (\mathbf{V}', \mathbf{E}')$ with $\mathcal{G}[^{\mathbf{S}}_{\mathbf{L}} = \mathcal{M}$ and $\mathbf{L} \cup \mathbf{S} = \mathbf{V}' \setminus \mathbf{V}$. Thereby one can define $P(\mathbf{y} \mid do(\mathbf{x}))$ as $\sum_{\mathbf{z}} P(\mathbf{y} \mid \mathbf{x}, \mathbf{z}, \mathbf{s})P(\mathbf{z} \mid \mathbf{s})$, $\sum_{\mathbf{z}} P(\mathbf{y} \mid \mathbf{x}, \mathbf{z}, \mathbf{s})P(\mathbf{z})$ or $\sum_{\mathbf{s}} \sum_{\mathbf{z}} P(\mathbf{y} \mid \mathbf{x}, \mathbf{z}, \mathbf{s})P(\mathbf{z}, \mathbf{s})$. The last definition is equivalent to $\mathbf{Z} \cup \mathbf{S}$ being an adjustment set in all these DAGs, but existing literature has used the second case[BTP14]. However, the first case captures the spirit of selection bias the most since in the presence of selection bias the probability distribution is only known given some selected bias $\mathbf{s}$.

4.3.1 Adjustment Amenability

In this section, we characterize a class of MAGs in which adjustment is impossible because of causal ambiguities – e.g., the simple MAG $X \to Y$ falls into this class, but the larger MAG $A \to X \to Y$ does not.

visible

Definition 4.9 (Visible edge [Zha08])**.** *Given a MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$, an edge $X \to D$ in $\mathbf{E}$ is called* visible *if in all DAGs $\mathcal{G} = (\mathbf{V}', \mathbf{E}')$ with $\mathcal{G}[^{\emptyset}_{\mathbf{L}} = \mathcal{M}$ for some $\mathbf{L} \subseteq \mathbf{V}'$, all d-connected walks between X and D in $\mathcal{G}$ that contain only nodes of $\mathbf{L} \cup X \cup D$ are directed paths. Otherwise,*

invisible

$X \to D$ is said to be invisible*.*

Intuitively, an invisible directed edge $X \to D$ means that there may exist hidden confounding factors between X and D, which is guaranteed not to be the case if the edge is visible.

Lemma 4.10 (Graphical conditions for edge visibility [Zha08])**.** *In a MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$, an edge $X \to D$ in $\mathbf{E}$ is visible if and only if there is a node A not adjacent to D where (1) $A \to X \in \mathbf{E}$ or $A \leftrightarrow X \in \mathbf{E}$, or (2) there is a collider path $A \leftrightarrow V_1 \leftrightarrow \ldots \leftrightarrow V_n \leftrightarrow X$ or $A \to V_1 \leftrightarrow \ldots \leftrightarrow V_n \leftrightarrow X$ where all V_i are parents of D.*

adjustment amenable

Definition 4.11. *We call a MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$* adjustment amenable *w.r.t. $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ if all proper causal paths from $\mathbf{X}$ to $\mathbf{Y}$ start with a visible directed edge.*

Lemma 4.12. *If a MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ is not adjustment amenable w.r.t. $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$, then there exists no valid adjustment set for $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{M}$.*

Proof. If the first edge $X \to D$ on some causal path to $\mathbf{Y}$ in $\mathcal{M}$ is not visible, then there exists a consistent DAG $\mathcal{G}$ where there is a non-causal path between X and $\mathbf{Y}$ via D that could only be blocked in $\mathcal{M}$ by conditioning on D or some of its descendants. But such conditioning would violate the adjustment criterion in $\mathcal{G}$. □

Note that adjustment amenability does not yet guarantee the existence of an adjustment set; the smallest example is the MAG $X \leftarrow Y$, which is adjustment amenable but admits no valid adjustment set.

4.3.2 Auxiliary Lemmas

In this subsection, we present the auxiliary lemmas used in the proof of Theorem 4.29, the main result of this section about MAGs. For Lemmas 4.21 and 4.25, we also give separate preparing claims and existing results before stating the lemmas themselves.

Lemma 3.11 has shown that m-separation can be defined equivalently by paths or by walks, and Lemma 3.12 that for walks m-separation given a set $\mathbf{Z}$ remains the same, regardless if colliders are defined to be opened by being in $\mathbf{Z}$ or being ancestors of $\mathbf{Z}$. Independently from these previous lemmas, we show now that such an equivalence also holds between proper non-causal paths and proper non-causal walks as long as $\mathbf{Z}$ contains no node of $Dpcp(\mathbf{X}, \mathbf{Y})$.

Lemma 4.13. *Given a DAG $\mathcal{G}$ and sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ satisfying $\mathbf{Z} \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$, the set $\mathbf{Z}$ m-connects a proper non-causal path π between $\mathbf{X}$ and $\mathbf{Y}$ if and only if it m-connects a proper non-causal walk w between $\mathbf{X}$ and $\mathbf{Y}$ such that the set $w \cap \mathbf{Z}$ is the set of all colliders on w.*

Proof. ($\Leftarrow$): Let w be the m-connected proper non-causal walk. It can be transformed to an m-connected path π by removing loops of nodes that are visited multiple times. Since no nodes have been added, π remains proper, and the first edges of π and w are the same. So if w does not start with a $\rightarrow$ edge, π is non-causal. If w starts with an edge $X \rightarrow D$, there exists a collider with a descendant in $\mathbf{Z}$ which is in $De(D)$. So π has to be non-causal, or it would contradict $\mathbf{Z} \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$.

($\Rightarrow$): Let π be an m-connected proper non-causal path. It can be changed to an m-connected walk w with $w \cap \mathbf{Z} = (\text{colliders of } w)$ by inserting $C_i \overset{*}{\rightarrow} Z_i \overset{*}{\leftarrow} C_i$ for every collider C_i on π and a corresponding $Z_i \in \mathbf{Z}$. Since no edges are removed from π, w is non-causal, but not necessarily proper, since the inserted walks might contain nodes of $\mathbf{X}$. However, in that case, w can be truncated to a proper walk w' starting at the last node of $\mathbf{X}$ on w. Then w' is non-causal since it contains the subpath $\mathbf{X} \overset{+}{\leftarrow} C_i$. □

Before proving Theorem 4.29, let us introduce the concepts of an *inducing path* and *inducing* $\mathbf{Z}$*-trail*, which we will use in our proof and the further analysis.

Definition 4.14 (Inducing path [RS02])**.** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and $\mathbf{Z}, \mathbf{L} \subseteq \mathbf{V}$ be disjoint. A path $\pi = V_1 \overset{+}{\sim} V_{k+1}$ in $\mathcal{G}$ is called* inducing *with respect to $\mathbf{Z}$ and $\mathbf{L}$ if all non-colliders on π are in $\mathbf{L}$ and all colliders on π are in $An(\{V_1, V_{k+1}\} \cup \mathbf{Z})$.* *inducing path*

Definition 4.15 (Inducing $\mathbf{Z}$-trail)**.** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and $\mathbf{Z}, \mathbf{L} \subseteq \mathbf{V}$ be disjoint. Let $\pi = V_1 \overset{+}{\sim} V_{k+1}$ be a path in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ such that $V_2, \ldots, V_k \in \mathbf{Z}$, $V_1, V_{k+1} \notin \mathbf{Z}$, for each $i \in \{1, \ldots, k\}$, there is an inducing path w.r.t. $\emptyset, \mathbf{L}$ linking V_i, V_{i+1}, and for each $i \in \{2, \ldots, k\}$, these inducing paths have arrowheads at V_i. Then π is called an* inducing $\mathbf{Z}$-trail. *inducing $\mathbf{Z}$-trail*

In all of the below, $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a DAG, $\mathbf{Z}, \mathbf{L} \subseteq \mathbf{V}$ are disjoint, and $\mathcal{M} = \mathcal{G}[^{\emptyset}_{\mathbf{L}}$. We notice first that every inducing path w.r.t. $\mathbf{Z}$ and $\mathbf{L}$ is m-connected by $\mathbf{Z}$.

Lemma 4.16 ([RS02])**.** *If there is an inducing path π from $U \in \mathbf{V}$ to $V \in \mathbf{V}$ with respect to $\mathbf{Z}, \mathbf{L}$, then there exists no set $\mathbf{Z}'$ with $\mathbf{Z} \subseteq \mathbf{Z}' \subseteq (\mathbf{V} \setminus \mathbf{L})$ such that $\mathbf{Z}'$ d-separates U and V in $\mathcal{G}$ or m-separates U and V in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$.*

Proof. This is Theorem 4.2, cases (v) and (vi), in [RS02]. □

Claim 4.17. *Two nodes U, V are adjacent in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ if and only if $\mathcal{G}$ contains an inducing path π between U and V with respect to $\emptyset, \mathbf{L}$. Moreover, the edge between U, V in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ can only have an arrowhead at U (V) if all such π have an arrowhead at U (V) in $\mathcal{G}$.*

Proof. The first part on adjacency is proved in [RS02]. For the second part on arrowheads, suppose π does not have an arrowhead at U, then π starts with an edge $U \to D$. Hence, $D \notin An(U)$, so $D \in An(V)$ because π is an inducing path, and therefore also $U \in An(V)$. Thus, the edge between U and V in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ must be $U \to V$. The argument for V is identical. □

Claim 4.18. *Suppose $Z_0 \sim Z_1 \sim Z_2$ is a path in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ on which Z_1 is a non-collider. Suppose an inducing path π_{01} from Z_0 to Z_1 w.r.t. $\emptyset, \mathbf{L}$ in $\mathcal{G}$ has an arrowhead at Z_1, and an inducing path π_{12} from Z_1 to Z_2 w.r.t. $\emptyset, \mathbf{L}$ has an arrowhead at Z_1. Then the walk $w_{012} = \pi_{01}\pi_{12}$ can be truncated to an inducing path from Z_0 to Z_2 w.r.t. $\emptyset, \mathbf{L}$ in $\mathcal{G}$.*

Proof. The walk w_{012} does not contain more non-colliders than those on π_{01} or π_{12}, so they must all be in $\mathbf{L}$. It remains to show that the colliders on w_{012} are in $An(Z_0 \cup Z_2)$. Because Z_1 is not a collider on $Z_0 \sim Z_1 \sim Z_2$, at least one of the edges $Z_0 \sim Z_1$ and $Z_1 \sim Z_2$ must be a directed edge pointing away from Z_1. Assume without loss of generality that $Z_0 \leftarrow Z_1$ is that edge. Then all colliders on π_{01} are in $An(Z_0 \cup Z_1) = An(Z_0) \subseteq An(Z_0 \cup Z_2)$, and all colliders on π_{12} are in $An(Z_1 \cup Z_2) \subseteq An(Z_0 \cup Z_2)$. Z_1 itself is a collider on w_{012} and is also in $An(Z_0)$. Hence, the walk w_{012} is d-connected, and can be truncated to an inducing path that starts with the first arrow of π_{01} and ends with the last arrow of π_{12}. □

Claim 4.19. *Let $\pi = V_1 \overset{+}{\sim} V_{k+1}$ be an inducing $\mathbf{Z}$-trail, and let π' be a subsequence of π formed by removing one non-collider V_i of π with $V_i \in \mathbf{Z}$. Then π' is an inducing $\mathbf{Z}$-trail.*

Proof. According to Claim 4.18, if V_i is a non-collider on π, then V_{i-1} and V_{i+1} are linked by an inducing path π that contains an arrowhead at V_{i-1} (V_{i+1}) if $V_{i-1} \in \mathbf{Z}$ ($V_{i+1} \in \mathbf{Z}$). Therefore, V_{i-1} and V_{i+1} are themselves adjacent, π' is a path, and is a $\mathbf{Z}$-trail. □

Corollary 4.20. *Every inducing $\mathbf{Z}$-trail $\pi = V_1 \overset{+}{\sim} V_{k+1}$ has a subpath π' that is m-connected by $\mathbf{Z}$.*

Proof. Transform π into π' by replacing non-collider nodes in $\mathbf{Z}$ by the direct edge linking their neighbors until no such node exists anymore. By inductively applying Claim 4.19, we see that π' is also an inducing $\mathbf{Z}$-trail, and every node in $\mathbf{Z}$ is a collider because otherwise we would have continued transforming. So π' must be m-connected by $\mathbf{Z}$. □

Lemma 4.21. *Let $w_{\mathcal{G}}$ be a walk from X to Y in $\mathcal{G}$, $X, Y \notin \mathbf{L}$, that is d-connected by $\mathbf{Z}$. Let $w_{\mathcal{M}} = V_1 \overset{+}{\sim} V_{k+1}$ be the subsequence of $w_{\mathcal{G}}$ consisting only of the nodes in $\mathcal{M} = \mathcal{G}[^{\emptyset}_{\mathbf{L}}$. Then $\mathbf{Z}$ m-connects X and Y in $\mathcal{M}$ via a path along a subsequence $w'_{\mathcal{M}}$ formed from $w_{\mathcal{M}}$ by removing some nodes in $\mathbf{Z}$ (possibly $w'_{\mathcal{M}} = w_{\mathcal{M}}$).*

Proof. First, truncate from $w_{\mathcal{M}}$ all subwalks between nodes in $\mathbf{Z}$ that occur more than once. Now consider all subsequences $V_1 \overset{+}{\sim} V_{l+1}$, $l > 1$, of $w_{\mathcal{M}}$ where $V_2, V_n \in \mathbf{Z}$, $V_1, V_{l+1} \notin \mathbf{Z}$, which now are all paths in $\mathcal{M}$. On those subsequences, every V_i must be adjacent in $\mathcal{G}$ to V_{i+1} via a path containing no colliders, and all internal nodes of that path must be in $\mathbf{L}$. So there are inducing paths w.r.t. $\emptyset, \mathbf{L}$ between all V_i, V_{i+1}, which have arrowheads at V_i (V_{i+1}) if $V_i \in \mathbf{Z}$ ($V_{i+1} \in \mathbf{Z}$). So $V_1 \overset{+}{\sim} V_{l+1}$ is an inducing $\mathbf{Z}$-trail, and has a subpath which m-connects V_1, V_{l+1} given $\mathbf{Z}$ due to Corollary 4.20. Transform $w_{\mathcal{M}}$ to $w'_{\mathcal{M}}$ by replacing all inducing $\mathbf{Z}$-trails by their m-connected subpaths. According to Claim 4.17, non-colliders on $w_{\mathcal{M}}$ cannot be colliders on $w'_{\mathcal{M}}$, as bypassing inducing paths can remove but not create arrowheads. Moreover, all nodes in $\mathbf{Z}$ on $w'_{\mathcal{M}}$ are colliders. Hence, $w'_{\mathcal{M}}$ is m-connected by $\mathbf{Z}$. □

Corollary 4.22. *Each edge on $w'_{\mathcal{M}}$ as defined above corresponds to an inducing path w.r.t. $\emptyset, \mathbf{L}$ in $\mathcal{G}$ along nodes on $w_{\mathcal{G}}$.*

Claim 4.23. *Suppose there exists an inducing path π_{01} from Z_0 to Z_1 w.r.t. $\mathbf{S}, \mathbf{L}$ with an arrowhead at Z_1 and an inducing path from Z_1 to Z_2 w.r.t. $\mathbf{S}', \mathbf{L}$ with an arrowhead at Z_1. Then the walk $w_{012} = \pi_{01}\pi_{12}$ can be truncated to an inducing path from Z_0 to Z_2 w.r.t. $\mathbf{S} \cup \mathbf{S}' \cup \{Z_1\}, \mathbf{L}$ in $\mathcal{G}$.*

Proof. The walk w_{012} does not contain more non-colliders than those on π_{01} or π_{12}, so they must all be in $\mathbf{L}$. All colliders on $\pi_{0,1}$ and $\pi_{1,2}$ as well as Z_1 are in $An(Z_0, Z_1, Z_2, \mathbf{S}, \mathbf{S}')$, and therefore also all colliders of w_{012}.

Hence, the walk w_{012} is d-connected, and can be truncated to an inducing path that starts with the first arrow of π_{01} and ends with the last arrow of π_{12}. □

Claim 4.24. *Suppose $Z_0, Z_1, \ldots, Z_{k+1}$ is a path in $\mathcal{G}\lfloor^{\emptyset}_{\mathbf{L}}$ with an arrowhead at Z_{k+1} on which all $Z_1, \ldots, Z_k$ are colliders. Then there exists an inducing path from Z_0 to Z_{k+1} w.r.t. $\{Z_1, \ldots, Z_k\}, \mathbf{L}$ with an arrowhead at Z_{k+1}.*

Proof. Because all Z_i, Z_{i+1} are adjacent and all $Z_1, \ldots, Z_k$ are colliders, there exist inducing paths $\pi_{i,i+1}$ w.r.t. $\emptyset, \mathbf{L}$ from Z_i to Z_{i+1} that have arrowheads at $Z_1, \ldots, Z_k$ (Claim 4.17). The claim follows by repeatedly applying Claim 4.23 to the $\pi_{i,i+1}$'s. □

Let $V_1 \overset{*}{\leftrightarrow} V_k$ denote a (possibly empty) walk $V_1 \overset{*}{\sim} V_k$ containing only bidirected edges.

Lemma 4.25. *Suppose $A \to V_1 \overset{*}{\leftrightarrow} V_k \leftrightarrow X \to D$ or $A \leftrightarrow V_1 \overset{*}{\leftrightarrow} V_k \leftrightarrow X \to D$ is a path in $\mathcal{G}\lfloor^{\emptyset}_{\mathbf{L}}$ (possibly $k = 0$), each V_i is a parent of D and there exists an inducing path π_{XD} from X to D w.r.t. $\emptyset, \mathbf{L}$ that has arrowheads on both ends. Then A and D cannot be m-separated in $\mathcal{G}\lfloor^{\emptyset}_{\mathbf{L}}$.*

Proof. Assume the path is $A \to V_1 \overset{*}{\leftrightarrow} V_k \leftrightarrow X \to D$. The case where the path starts with $A \leftrightarrow V_1$ can be handled identically since the first arrowhead does not affect m-separation.

Assume A and D can be m-separated in $\mathcal{G}\lfloor^{\emptyset}_{\mathbf{L}}$, and let $\mathbf{Z}$ be such a separator. If V_1 is not in $\mathbf{Z}$, then the path $A \to V_1 \to D$ is not blocked, so $V_1 \in \mathbf{Z}$. Inductively, it follows: if V_i is not in $\mathbf{Z}$, but all $\forall j < i : V_j \in \mathbf{Z}$, then the path $A \to V_1 \overset{*}{\leftrightarrow} V_{i-1} \leftrightarrow V_i \to D$ is not blocked, so $V_i \in \mathbf{Z}$ for all i.

There exists an inducing path π_{AX} from A to X with an arrowhead at X w.r.t. to $\{V_1, \ldots, V_k\}, \mathbf{L}$ (Claim 4.24) which can be combined with π_{XD} to an inducing path from A to D w.r.t. to $\{V_1, \ldots, V_k, X\}, \mathbf{L}$ (Claim 4.23).

Hence, no separator relative to (A, D) can contain $\{X, V_1, \ldots, V_k\}$ (Lemma 4.16). Then there cannot exist a separator, because every separator must include $V_1, \ldots, V_k$ and the path $A \to V_1 \overset{*}{\leftrightarrow} V_k \leftrightarrow X \to D$ is open without $X \in \mathbf{Z}$. □

4.3.3 Adjustment Criterion for MAGs

We now show that the adjustment criterion for DAGs generalizes to adjustment amenable MAGs. The adjustment criterion and the constructive back-door criterion are defined like their DAG counterparts (Definitions 4.2 and 4.5), replacing "DAG" with "MAG" and d- with m-separation for the latter.

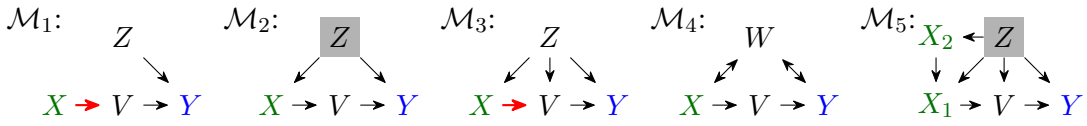


Figure 4.2: Five MAGs in which we search for an adjustment relative to (X, Y) or $(\{X_1, X_2\}, Y)$. $\mathcal{M}_1$ and $\mathcal{M}_3$ are not adjustment amenable, since the edge $X \to V$ is not visible, so no adjustment exists. In the other three MAGs, the edge is visible because of the node Z in $\mathcal{M}_2$, the node W in $\mathcal{M}_4$, and the node X_2 in $\mathcal{M}_5$. The only valid adjustment in $\mathcal{M}_2$ and $\mathcal{M}_5$ is $\{Z\}$, and in $\mathcal{M}_4$ only the empty set is a valid adjustment. If $\mathcal{M}_1$ and $\mathcal{M}_3$ were DAGs, the set $\{Z\}$ would be an adjustment in each.

Definition 4.26 (Adjustment criterion)**.** *Let $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ be a MAG, and $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. The set $\mathbf{Z}$ satisfies the adjustment criterion relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{M}$ if*

(a) *no element in $\mathbf{Z}$ is a descendant in $\mathcal{M}$ of any $W \in \mathbf{V} \setminus \mathbf{X}$ which lies on a proper causal path from $\mathbf{X}$ to $\mathbf{Y}$ and*

(b) *all proper non-causal paths in $\mathcal{M}$ from $\mathbf{X}$ to $\mathbf{Y}$ are blocked by $\mathbf{Z}$.*

proper back-door graph

Definition 4.27 (Proper back-door graph)**.** *Let $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ be a MAG, and $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. The* proper back-door graph*, denoted as $\mathcal{M}^{pbd}_{\mathbf{XY}}$, is obtained from $\mathcal{M}$ by removing the first edge of every proper causal path from $\mathbf{X}$ to $\mathbf{Y}$.*

constructive back-door criterion

Definition 4.28 (Constructive back-door criterion (CBC))**.** *Let $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ be a MAG, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. The set $\mathbf{Z}$ satisfies the* constructive back-door criterion *relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{M}$ if*

(a) $\mathbf{Z} \subseteq \mathbf{V} \setminus \mathit{Dpcp}(\mathbf{X}, \mathbf{Y})$ *and*

(b) $\mathbf{Z}$ *m-separates $\mathbf{X}$ and $\mathbf{Y}$ in the proper back-door graph $\mathcal{M}^{pbd}_{\mathbf{XY}}$.*

With these definitions we prove:

Theorem 4.29. *Given an adjustment amenable MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ and three disjoint node sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$, the following statements are equivalent:*

(i) $\mathbf{Z}$ *is an adjustment set relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{M}$.*

(ii) $\mathbf{Z}$ *fulfills the adjustment criterion (AC) w.r.t. $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{M}$.*

(iii) $\mathbf{Z}$ *fulfills the constructive back-door criterion (CBC) w.r.t. $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{M}$.*

Proof. The equivalence of (ii) and (iii) is established by observing that the proofs of Theorem 4.6 generalize to m-separation. Below we establish equivalence of (i) and (ii).

$\neg(ii) \Rightarrow \neg(i)$: If $\mathbf{Z}$ violates the adjustment criterion in $\mathcal{M}$, it does so in the canonical DAG $\mathcal{C}(\mathcal{M})$, and thus is not an adjustment in $\mathcal{M}$.

$\neg(i) \Rightarrow \neg(ii)$: In the proof, we rely on properties from Lemmas 4.13, 4.21, and 4.25 presented in Subsection 4.3.2.

Figure 4.3: Illustration of the case in the proof of Theorem 4.29 where Z descends from W_1 which in a DAG $\mathcal{G}$ is on a proper causal path from X to Y, but is not a descendant of a node on a proper causal path from X to Y in the MAG $\mathcal{M}$ after marginalizing W_1. In such cases, conditioning on Z will m-connect X and Y in $\mathcal{M}$ via a proper non-causal path.

Let $\mathcal{G}$ be a DAG with $\mathcal{G}_{\downarrow \mathbf{L}}^{\emptyset} = \mathcal{M}$ in which $\mathbf{Z}$ violates the AC. We show that (a) if $\mathbf{Z} \cap \mathit{Dpcp}(\mathbf{X}, \mathbf{Y}) \neq \emptyset$ in $\mathcal{G}$, then $\mathbf{Z} \cap \mathit{Dpcp}(\mathbf{X}, \mathbf{Y}) \neq \emptyset$ in $\mathcal{M}$ as well, or else there exists a proper non-causal path in $\mathcal{M}$ that cannot be m-separated; and (b) if $\mathbf{Z} \cap \mathit{Dpcp}(\mathbf{X}, \mathbf{Y}) = \emptyset$ in $\mathcal{G}$ and $\mathbf{Z}$ d-connects a proper non-causal path in $\mathcal{G}$, then it m-connects a proper non-causal path in $\mathcal{M}$.

(a) Suppose that in $\mathcal{G}$, $\mathbf{Z}$ contains a node Z in $\mathit{Dpcp}(\mathbf{X}, \mathbf{Y})$, and let $\mathbf{W} = \mathit{PCP}(\mathbf{X}, \mathbf{Y}) \cap \mathit{An}(Z)$. If $\mathcal{M}$ still contains at least one node $W_1 \in \mathbf{W}$, then W_1 lies on a proper causal path in $\mathcal{M}$ and Z is a descendant of W_1 in $\mathcal{M}$. Otherwise, $\mathcal{M}$ must contain a node $W_2 \in \mathit{PCP}_{\mathcal{G}}(\mathbf{X}, \mathbf{Y}) \setminus \mathit{An}(Z)$ (possibly $W_2 \in \mathbf{Y}$) such that $W_2 \leftrightarrow A$, $X \to W_2$, and $X \to A$ are edges in $\mathcal{M}$, where $A \in \mathit{An}(Z)$ (possibly $A = Z$; see Figure 4.3). Then $\mathcal{M}$ contains an m-connected proper non-causal path $X \to A \leftrightarrow W_2 \overset{*}{\to} Y$.

(b) Suppose that in $\mathcal{G}$, $\mathbf{Z} \cap \mathit{Dpcp}(\mathbf{X}, \mathbf{Y}) = \emptyset$, and there exists an open proper non-causal path from $\mathbf{X}$ to $\mathbf{Y}$. Then there must also be a proper non-causal *walk* $w_{\mathcal{G}}$ from some $X \in \mathbf{X}$ to some $Y \in \mathbf{Y}$ in $\mathcal{G}$, on which no non-collider is in $\mathbf{Z}$ and all colliders are in $\mathbf{Z}$. Let $w_{\mathcal{M}}$ denote the subsequence of $w_{\mathcal{G}}$ formed by nodes in $\mathcal{M}$. It includes all colliders on $w_{\mathcal{G}}$ because $\mathbf{Z} \cap \mathbf{L} = \emptyset$. The sequence $w_{\mathcal{M}}$ is a proper walk in $\mathcal{M}$, but is not necessarily m-connected by $\mathbf{Z}$; all colliders on $w_{\mathcal{M}}$ are in $\mathbf{Z}$ because every non-$\mathbf{Z}$ must be a parent of at least one of its neighbors, but there might be subsequences $U \sim Z_1 \overset{*}{\leftrightarrow} Z_k \sim V$ on $w_{\mathcal{M}}$ where all $Z_i \in \mathbf{Z}$, but some of the Z_i are not colliders on $w_{\mathcal{M}}$. However, then we can form from $w_{\mathcal{M}}$ an m-connected walk by bypassing some sequences of $\mathbf{Z}$-nodes (Lemma 4.21). Let $w'_{\mathcal{M}}$ be the resulting proper walk.

If $w'_{\mathcal{M}}$ is a proper non-causal walk, then there must also exist a proper non-causal path in $\mathcal{M}$ (Lemma 4.13), violating the AC. It therefore remains to show that $w'_{\mathcal{M}}$ is not a proper causal walk. This must be the case if $w_{\mathcal{G}}$ does not contain colliders, because then the first edge of $w_{\mathcal{M}} = w'_{\mathcal{M}}$ cannot be a visible directed edge out of X. Otherwise, the only way for $w'_{\mathcal{M}}$ to be proper causal is if all $\mathbf{Z}$-nodes in $w_{\mathcal{M}}$ have been bypassed in $w'_{\mathcal{M}}$ by edges pointing away from $\mathbf{X}$. In that case, one can show by several case distinctions that the first edge $X \to D$ of $w'_{\mathcal{M}}$, where $D \notin \mathbf{Z}$, cannot be visible (see Figure 4.4 for an example of such a case).

For simplicity, assume that $\mathcal{M}$ contains a subpath $A \to X \to D$ where A is not adjacent to D; the other cases of edge visibility like $A \leftrightarrow X \to D$ (Lemma 4.10) are treated analogously. In $\mathcal{G}$, there are inducing paths (possibly several π_{AX} from A to X and π_{XD} from X to D w.r.t. $\emptyset, \mathbf{L}$; π_{AX} must have an arrowhead at X. We distinguish several cases on the shape of π_{XD}. (1) A path π_{XD} has an arrowhead at X as well. Then A and D are adjacent (Lemma 4.25), a contradiction. (2) No inducing path π_{XD} has an arrowhead at X. Then $w_{\mathcal{G}}$ must start with an arrow out of X, and must contain a collider $Z \in \mathit{De}(X)$ because $w_{\mathcal{G}}$ is not causal. (a) $Z \in \mathit{De}(D)$. This contradicts $\mathbf{Z} \cap \mathit{Dpcp}(\mathbf{X}, \mathbf{Y}) = \emptyset$. So (b) $Z \notin \mathit{De}(D)$. Then by construction of $w'_{\mathcal{M}}$ (Lemma 4.21), $w_{\mathcal{M}}$ must start with an inducing $\mathbf{Z}$-trail $X \to Z \sim Z_1 \overset{*}{\leftrightarrow} Z_n \sim D$,

Figure 4.4: Case (b) in the proof of Theorem 4.29: A proper non-causal path $\pi_{\mathcal{G}} = X \leftarrow L_1 \rightarrow Z \leftarrow L_2 \rightarrow Y$ in a DAG is d-connected by $\mathbf{Z}$, but the corresponding proper non-causal path $\pi_{\mathcal{M}} = X \leftarrow Z \rightarrow Y$ is not m-connected in the MAG, and its m-connected subpath $\pi'_{\mathcal{M}} = X \rightarrow Y$ is proper causal. However, this also renders the edge $X \rightarrow Y$ invisible, because otherwise A could be m-separated from Y by $\mathbf{U} = \{X, Z\}$ in $\mathcal{M}$, but not in $\mathcal{G}$.

which is also an inducing path from X to D in $\mathcal{G}$ w.r.t. $\emptyset, \mathbf{L}$. Then $Z \sim Z_1 \overset{*}{\sim} Z_n \sim D$ must also be an inducing path in $\mathcal{G}$ w.r.t. $\emptyset, \mathbf{L}$ because $An(X) \subseteq An(Z)$. Hence, Z and D are adjacent. We distinguish cases on the path $X \rightarrow Z \sim D$ in $\mathcal{M}$. Now we can conclude: If $X \rightarrow Z \rightarrow D$, then Z lies on a proper causal path, contradicting $\mathbf{Z} \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$; If $X \rightarrow Z \leftrightarrow D$, or $X \rightarrow Z \leftarrow D$, then we get an m-connected proper non-causal walk along Z and D. □

4.3.4 The Class of the Back-Door Graph

The CBC is supposedly simpler to use than the AC because it allows one to apply existing separation algorithms. However, our algorithms assume that the input graph is a MAG or RCG, so it is not immediately clear if the algorithms can actually be used on proper back-door graphs. It might be possible that removing edges from a MAG results in a graph that is not a MAG.

But this cannot happen when removing only directed edges.

Lemma 4.30 (Closure of maximality under removal of directed edges)**.** *Given a MAG $\mathcal{M}$, every graph $\mathcal{M}'$ formed by removing only directed edges from $\mathcal{M}$ is also a MAG.*

Proof. Suppose the converse, i.e., $\mathcal{M}$ is no longer a MAG after removal of some edge $X \rightarrow D$. Then X and D cannot be m-separated even after the edge is removed because X and D are collider connected via a path whose nodes are all ancestors of X or D [RS02]. The last edge on this path must be $C \leftrightarrow D$ or $C \leftarrow D$; hence, $C \notin An(D)$, and thus we must have $C \in An(X)$. But then we get $C \in An(D)$ in $\mathcal{M}$ via the edge $X \rightarrow D$, a contradiction. □

Corollary 4.31. *For every MAG $\mathcal{M}$, the proper back-door graph $\mathcal{M}^{pbd}_{\mathbf{XY}}$ is also a MAG.*

4.4 Adjustment in RCGs

In this subsection, we propose a method to find adjustment sets in a given restricted chain graph. Figure 4.5 shows an example of an adjustment in an RCG.

A criterion for RCGs can be given without introducing a concept like adjustment amenability, but we need to specify the handling of undirected edges. In a path like $X - Y$, the undirected edge represents both a causal path $X \rightarrow Y$ or a non-causal path $X \leftarrow Y$, so we modify Definition 4.4 to allow undirected edges in the causal paths, but do not remove them in the proper back-door graph, because as non-causal paths they must be blocked.

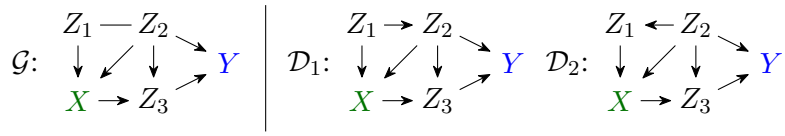


Figure 4.5: A (restricted) chain graph $\mathcal{G}$ which represents two Markov equivalent DAGs: $\mathcal{D}_1$ and $\mathcal{D}_2$. Relative to exposure X and outcome Y, $\mathbf{Z} = \{Z_2\}$ is an adjustment set both in $\mathcal{D}_1$ and in $\mathcal{D}_2$. Thus, $\mathbf{Z}$ is an adjustment set in the chain graph.

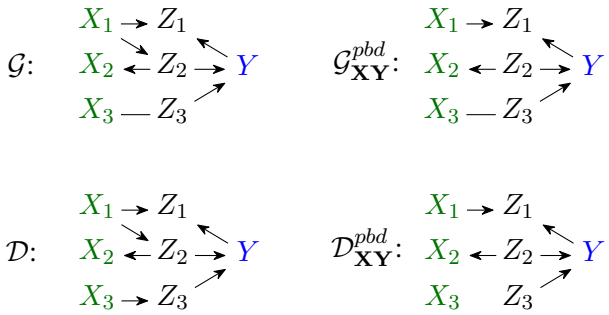


Figure 4.6: Proper back-door graphs for a (restricted) chain graph $\mathcal{G}$ and a DAG $\mathcal{D} \in CE(\mathcal{G})$, both with $\mathbf{X} = \{X_1, X_2, X_3\}, \mathbf{Y} = \{Y\}$.

Definition 4.32 (Proper back-door graph)**.** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a chain graph, and $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ be disjoint subsets of variables. The* proper back-door graph*, denoted as $\mathcal{G}^{pbd}_{\mathbf{XY}}$, is obtained from $\mathcal{G}$ by removing the first edge of every proper possibly causal path from $\mathbf{X}$ to $\mathbf{Y}$ that starts with a directed edge.* *proper back-door graph*

This is equivalent to a new definition of $PCP(\mathbf{X}, \mathbf{Y})$:

$$\begin{aligned} PCP(\mathbf{X}, \mathbf{Y}) &= \{W \in \mathbf{V} \setminus \mathbf{X} \mid W \text{ lies on a proper possibly causal path from } \mathbf{X} \text{ to } \mathbf{Y}\} \\ &= (pDe_{\overline{\mathbf{X}}}(\mathbf{X}) \setminus \mathbf{X}) \cap pAn_{\underline{\mathbf{X}}}(\mathbf{Y}). \end{aligned}$$

The back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ can still be constructed as $\mathcal{G}^{pbd}_{\mathbf{XY}} = (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \to PCP(\mathbf{X}, \mathbf{Y})))$. Figure 4.6 shows the difference between the proper back-door graph of an example RCG and a represented DAG.

Definition 4.2 then needs to be extended to handle possible descendants and almost definite status paths:

Definition 4.33 (Constructive Back-Door Criterion for RCGs). *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a chain graph, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. $\mathbf{Z}$ satisfies the* constructive back-door criterion *relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$ if* *constructive back-door criterion*

(a) $\mathbf{Z} \subseteq \mathbf{V} \setminus pDe(PCP(\mathbf{X}, \mathbf{Y}))$ *and*

(b) $\mathbf{Z}$ *blocks every almost definite status path from $\mathbf{X}$ to $\mathbf{Y}$ in the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$.*

To prove that this definition of a CBC characterizes adjustment sets, we need three utility lemmas:

Lemma 4.34. *For every node V in a chain component $\mathcal{C}$ of an RCG $\mathcal{G}$, the edges of $\mathcal{C}$ can be oriented such that for every other node W in $\mathcal{C}$ there exists a directed path $V \dot{\to} W$.*

Proof. Since all nodes in the chain component have the same parents, we can start the MCS orientation algorithm of [AMP97] at node V. It returns a sequence $\alpha_1, \ldots, \alpha_{|\mathcal{C}|}$ such that $\alpha_1 = V$, and if every edge $\alpha_i - \alpha_j$ with $i < j$ is oriented to $\alpha_i \to \alpha_j$, the resulting graph is acyclic without immoralities.

Assume there exists a node W to which no directed path $V \dot{\to} W$ exists. Let $\pi : V \dot{-} W$ be the shortest path between V and W. The path starts with a directed edge $V \to$ since $V = \alpha_1$. Because the path is not directed, there is a collider C with $V \dot{\to} B \to C \leftarrow D \dot{\leftarrow} W$. But this would be an v-structure unless B and D are connected, so $V \dot{\to} B \sim D \dot{\leftarrow} W$ would be a shorter path than π, which is also open since B and D are ancestors of C.

□

Lemma 4.35. *Let $\mathcal{G}$ be an RCG and $\mathcal{D} \in CE(\mathcal{G})$. If there exists a d-connected path $\pi : X \dot{\sim} Y$ given $\mathbf{Z}$ in $\mathcal{D}$, there exists a d-connected given $\mathbf{Z}$ almost definite status path $\pi_G : X \dot{\sim} Y$ in $\mathcal{G}$ with the following properties:*

Every node V on π_G is a node on π, or occurs as $V \leftarrow$ or as $V \to W$ with $W \in An_{\mathcal{D}}(\mathbf{Z})$ in $\mathcal{D}$.

$\pi_{\mathcal{G}}$ is not a directed path unless $\pi_{\mathcal{D}}$ is directed or a node other than X of π_G is in $pAn_{\mathcal{G}}(\mathbf{Z})$.

Proof. Let π be the shortest d-connected path between X and Y given $\mathbf{Z}$ in $\mathcal{D}$. If this path is not an almost definitive status path in $\mathcal{G}$ the path in $\mathcal{G}$ contains $U \to V - W$ or $U - V \leftarrow W$. In the first case, there has to exist an edge $U \to W$ in $\mathcal{G}$ due to Lemma 3.1, so we can replace $U \sim V \sim W$ by $U \to W$ to form a shorter path π' in $\mathcal{D}$. If the edge $V \sim W$ is $V \to W$ in $\mathcal{D}$, the node W is a collider on π' iff it is a collider on π, and π' is a directed path iff π' is one. If the edge is $V \leftarrow W$, V is a collider and an ancestor of $\mathbf{Z}$, so W is an ancestor of $\mathbf{Z}$ in $\mathcal{D}$ and a possible ancestor of $\mathbf{Z}$ in $\mathcal{G}$. After the replacement, π' is a d-connected shorter path than π.

The same can be shown in the second case, except that there π' will have an edge $U \leftarrow W$ in $\mathcal{G}$ and thus cannot be a directed path.

None of the above two cases removes edges without adding a new edge, and that edge is only directed from X to Y, if the three node subpath of π is a directed path or the last node of the edge is a possible ancestor of $\mathbf{Z}$. These possible ancestors are preserved during later replacements in that sense that the last node of the newly inserted edge will also be a possible ancestor of $\mathbf{Z}$.

It remains to show that there also exists a *d-connected* almost definite status path with these properties, i.e., such that all colliders are ancestors of $\mathbf{Z}$ in $\mathcal{G}$. Let π be a path that is of almost definite status in $\mathcal{G}$ and a d-connected path between X and Y given $\mathbf{Z}$ in $\mathcal{D}$, chosen such that it is the path with the lowest number of blocked colliders in $\mathcal{G}$. Let V be the first such collider, i.e., π contains $U \to V \leftarrow W$ in both $\mathcal{G}$ and $\mathcal{D}$, and V is an ancestor of $\mathbf{Z}$ in $\mathcal{D}$ and not in $\mathcal{G}$. Then there exist paths $U \to V \dot{\leftarrow} Z$ and $W \to V \dot{\leftarrow} Z$ in $\mathcal{G}$, with $Z \in \mathbf{Z}$, that do not contain any other node of $\mathbf{Z}$. Due to Corollary 3.3, there are also d-connected paths $U \dot{\to} Z$ and $W \dot{\to} Z$ in $\mathcal{G}$ and $\mathcal{D}$. These paths intersect in a node V', and we can replace $U \to V \leftarrow W$ by $U \dot{\to} V' \dot{\leftarrow} W$ unless the paths also intersect π in a node $V'' \notin \{U, V, W\}$. If one of them intersects the prefix $\pi[X \dot{\sim} U]$, we can replace the prefix $\pi[X \dot{\sim} W]$ with $\pi[X \dot{\sim} V''] \dot{\leftarrow} W$, and if they intersect the suffix $\pi[W \dot{\sim} Y]$, we replace the suffix $\pi[U \dot{\sim} Y]$ with $U \dot{\to} \pi[V'' \dot{\sim} Y]$, reducing the number of blocked colliders. The number of colliders can only be reduced, because, if the number of colliders remained the same and V'' became

a collider after the replacement, V'' would be a possible ancestor of V and the graph would contain a semi-directed cycle.

This is the only step introducing new nodes, and they only occur as $V \leftarrow$ or as $V \rightarrow W$ with $W \in An_{\mathcal{D}}(\mathbf{Z})$ in $\mathcal{D}$. □

To prove the correctness of the constructive back-door criterion for chain graphs, we need to show that the criterion holds for an arbitrary RCG $\mathcal{G}$ if and only if it holds for every DAG in $CE(\mathcal{G})$.

Theorem 4.36. *Let $\mathcal{G}$ be an RCG. Then the constructive back-door criterion (Definition 4.33) holds in $\mathcal{G}$ for sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ if and only if $\mathbf{Z}$ is an adjustment set relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$.*

Proof. If $\mathcal{G}$ does not contain an undirected edge, then it is a DAG and $CE(\mathcal{G}) = \{\mathcal{G}\}$, Definition 4.33 is equivalent to Definition 4.5, and the claim of Theorem 4.36 is equivalent to the claim of Theorem 4.6: In the absence of undirected edges, the definition of descendants and possible descendants are equal as well as the definition of d-connected almost definite status paths and d-connected paths.

Otherwise, we need to show if $\mathcal{G}$ does not satisfy the CBC, there $\exists \mathcal{D} \in CE(\mathcal{G})$ that does not satisfy the CBC; and if $\exists \mathcal{D} \in CE(\mathcal{G})$ that does not satisfy the CBC, $\mathcal{G}$ does not satisfy the CBC.

($\Rightarrow$): Assume $\mathcal{G}$ does not satisfy CBC(a), i.e., there exist nodes $Z \in \mathbf{Z}, W \in PCP_{\mathcal{G}}(\mathbf{X}, \mathbf{Y})$ with paths $\mathbf{X} \stackrel{+}{\rightarrow} W \stackrel{*}{\rightarrow} \mathbf{Y}$ and $W \stackrel{*}{\rightarrow} Z$. Due to Corollary 3.3, we can assume these paths have the form $\stackrel{*}{-}\stackrel{*}{\rightarrow}$, using possibly another W. The path $X \stackrel{+}{\sim} W \stackrel{*}{\sim} Z$ has this form, too, unless the paths are $X \stackrel{*}{-}\stackrel{+}{\rightarrow} W \stackrel{*}{\rightarrow} Y$ and $W \stackrel{+}{-}\stackrel{*}{\rightarrow} Z$. If the first path starts with an undirected edge $X - W' \stackrel{*}{-}\stackrel{+}{\rightarrow} W \stackrel{+}{\rightarrow} Y$, we can use W' instead of W and get a path $W' \stackrel{*}{-}\stackrel{*}{\rightarrow} Z$ with Corollary 3.3.

In all cases, these paths contain only a single undirected subpath, either $X \stackrel{*}{-}$ as a common prefix or $W \stackrel{*}{-}$. According to Lemma 4.34, there exists a DAG $\mathcal{D} \in CE(\mathcal{G})$ in which this subpath is oriented to a directed path $X \stackrel{*}{\rightarrow}$ or $W \stackrel{*}{\rightarrow}$, so Z is also a descendant of $PCP_{\mathcal{D}}(\mathbf{X}, \mathbf{Y})$ in $\mathcal{D}$ and $\mathcal{D}$ violates CBC(a).

Assume $\mathcal{G}$ does not satisfy CBC(b), i.e., there exists a (proper) d-connected almost definite status path π between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Let $\mathcal{D} \in CE(\mathcal{G})$ be a DAG that does not violate CBC(a). Due to Lemma 3.9, there exists a definite status path π' between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$ that has the same directed edges, and thus exist in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ as well. Since π' is of definite status, π' exists in $\mathcal{D}$ as d-connected path. If π' starts with $X -$ in $\mathcal{G}$, we can assume it starts with $X \leftarrow$ in $\mathcal{D}$, and the first edge is not removed in $\mathcal{D}^{pbd}_{\mathbf{XY}}$. Otherwise, π' starts with a directed edge in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, and every directed edge of $\mathcal{G}^{pbd}_{\mathbf{XY}}$ also exists in $\mathcal{D}^{pbd}_{\mathbf{XY}}$: Otherwise, there would be an edge $X \rightarrow D$ removed in $\mathcal{D}^{pbd}_{\mathbf{XY}}$ which is not removed in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, i.e., a node in $PCP_{\mathcal{D}}(\mathbf{X}, \mathbf{Y})$ that is not a node in $PCP_{\mathcal{G}}(\mathbf{X}, \mathbf{Y})$, i.e., a proper path $\mathbf{X} \stackrel{+}{\rightarrow} W \stackrel{*}{\rightarrow} \mathbf{Y}$ in $\mathcal{D}$ which does not correspond to a proper path $\mathbf{X} \stackrel{+}{\rightarrow} W \stackrel{*}{\rightarrow} \mathbf{Y}$ in $\mathcal{G}$.

Since π' is proper, no node of $\mathbf{X}$ is an internal node on π', so no removed edge occurs in π' and the edges of π' exists (oriented) in $PCP_{\mathcal{D}}(\mathbf{X}, \mathbf{Y})$.

Also every node of $\mathbf{Z}$ that is an descendant of a collider in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ is also a descendant of the collider in $\mathcal{D}^{pbd}_{\mathbf{XY}}$, π' is d-connected in $\mathcal{D}^{pbd}_{\mathbf{XY}}$, and $\mathcal{D}$ violates CBC(b).

($\Leftarrow$): Assume there is a DAG $\mathcal{D}$ that does not satisfy CBC(a). Then there exist paths $X \stackrel{+}{\rightarrow} W \stackrel{+}{\rightarrow} Y$ and $W \stackrel{*}{\rightarrow} \mathbf{Z}$, which exist in $\mathcal{G}$ as paths $X \stackrel{+}{\rightarrow} W \stackrel{+}{\rightarrow} Y$ and $W \stackrel{*}{\rightarrow} \mathbf{Z}$. Thus, $\mathcal{G}$ does not satisfy CBC(a).

Assume there is a DAG $\mathcal{D}$ that does not satisfy CBC(b). Then there exists a proper d-connected path π_D in $\mathcal{D}^{pbd}_{\mathbf{XY}}$ and $\mathcal{D}$, which corresponds to a d-connected definite status path π_G in $\mathcal{G}$. If any other edge of π_G than the first one does not exist in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, π_D is not proper or an directed edge $X \to \mathit{PCP}_{\mathcal{G}}(\mathbf{X}, \mathbf{Y})$ was inserted in π_G, which violates CBC(a) in $\mathcal{D}$ due to Lemma 4.35. If π_G does not start with a directed edge $X \to \mathit{PCP}_{\mathcal{G}}(\mathbf{X}, \mathbf{Y})$, it also exist in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, and $\mathcal{G}$ violates CBC(b). Every open collider C in $\mathcal{G}$ is also an open collider in $\mathcal{G}^{pbd}_{\mathbf{XY}}$; otherwise, there would be a path $C \overset{*}{\to} X \to \mathit{PCP}(\mathbf{X}, \mathbf{Y}) \overset{*}{\to} \mathbf{Z}$ in $\mathcal{G}$, so $\mathcal{G}$ would violate CBC(a).

If the path starts with an edge $X \to \mathit{PCP}_{\mathcal{G}}(\mathbf{X}, \mathbf{Y})$ and contains a collider, $\mathcal{G}$ violates CBC(a). If it does not contain a collider, π_D is a directed path $\mathbf{X} \overset{+}{\to} \mathbf{Y}$ or it would contain $\to\!\!-$ and not be of definite status, so it does not exist in $\mathcal{D}^{pbd}_{\mathbf{XY}}$, or π_G in $\mathcal{D}$ is a directed path and one node other than X has a node of $\mathbf{Z}$ as possible descendant due to Lemma 4.35 and $\mathcal{G}$ violates CBC(b). □

4.4.1 Properties of the Proper Back-Door Graph

As for MAGs, we need to know if the proper back-door graph of an RCG is still an RCG before we can use our separation algorithms on it. Unfortunately, in this case not every proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ of an RCG $\mathcal{G}$ is an RCG itself. However, we can show that the algorithms are still valid with two auxiliary lemmas.

The first lemma shows that the proper back-door graph is always an RCG, if $\mathbf{X}$ is a singleton $\{X\}$ or there exists at least one adjustment set.

Lemma 4.37. *If the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ of an RCG $\mathcal{G}$ is not an RCG, then*

1. $|\mathbf{X}| > 1$, *and*
2. *no adjustment set exists relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$.*

Proof. It is easy to see that a back-door graph always satisfies the first conditions of RCGness: It is a chain graph since removing an edge cannot introduce a cycle. All chain components are still chordal since they are unchanged.

If an induced subgraph $X \to B - P$ exists, an edge $X \to P$ was removed, i.e., $X \in \mathbf{X}$ and $P \in \mathit{PCP}(\mathbf{X}, \mathbf{Y})$. So there exists a possibly directed path π from P to $\mathbf{Y}$ not intersecting $\mathbf{X}$ in $\mathcal{G}$ and $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Through the induced subgraph, there is a possibly directed path $B - \pi$ from B to $\mathbf{Y}$, so the edge $X \to B$ can only exist in the back-door graph if $B \in \mathbf{X}$. With X and B in $\mathbf{X}$, the size of $\mathbf{X}$ is at least 2.

According to Corollary 3.3, all non-leading undirected edges can be removed from $B - \pi$, which creates a proper definite status path from $\mathbf{X}$ to $\mathbf{Y}$. Since this path starts with an undirected edge, it exists in the back-door graph as well and must be blocked by every adjustment set. But it can only be blocked by nodes in $\mathit{PCP}(\mathbf{X}, \mathbf{Y})$, so no adjustment set exists. □

This means we can always assume $\mathcal{G}^{pbd}_{\mathbf{XY}}$ is an RCG and use all our algorithms without testing whether $\mathcal{G}^{pbd}_{\mathbf{XY}}$ is an RCG first. If an algorithm cannot find an adjustment set, it is always correct and no adjustment exists. If an algorithm finds at least one adjustment set, we can test if it is actually an adjustment set, and then all other adjustments sets found by the algorithm are also correct.

The second lemma shows that the result of Lemma 3.10 holds in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ even if $\mathcal{G}^{pbd}_{\mathbf{XY}}$ is not an RCG.

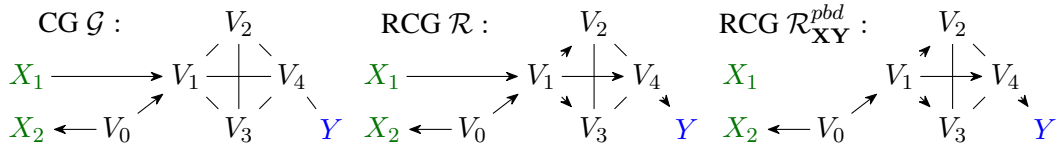


Figure 4.7: Finding an adjustment in an input chain graph $\mathcal{G}$ with $\mathbf{X} = \{X_1, X_2\}$, $\mathbf{Y} = \{Y\}$. The RCG $\mathcal{R}$ is constructed from $\mathcal{G}$ by replacing every induced subgraph $\rightarrow\!-$ with $\rightarrow\!\rightarrow$, and $\mathcal{R}$ satisfies $CE(\mathcal{G}) = CE(\mathcal{R})$. The only chain component $\{V_2, V_3, V_4\}$ is chordal. To apply the CBC, we identify the forbidden nodes $pDe(PCP(\mathbf{X}, \mathbf{Y})) = \{V_1, V_2, V_3, V_4, Y\}$, and construct the proper back-door graph $\mathcal{R}^{pbd}_{\mathbf{XY}}$. The (only) adjustment set is $\mathbf{Z} = \{V_0\}$.

Lemma 4.38. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an RCG, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. In the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$, the set $\mathbf{Z}$ blocks every almost definite status path between $\mathbf{X}$ and $\mathbf{Y}$ if and only if $\mathbf{Z}$ blocks every definite status path between $\mathbf{X}$ and $\mathbf{Y}$.*

Proof. If there exists a definite status path between $\mathbf{X}$ and $\mathbf{Y}$, it is also an almost definite status path. On the other hand, if there exists a d-connected almost definite status path π between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, π is also a d-connected almost definite status path in $\mathcal{G}$, and due to Lemma 3.9, there exists a d-connected definite status path π' in $\mathcal{G}$. Both paths have the same directed edges; thus, π' is also a d-connected definite status path in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. □

So we do not need to introduce a special definite status path based separation algorithm for the proper back-door graph, but can apply algorithm TESTSEP. Although algorithm TESTSEP was only described for RCGs, Lemma 4.38 shows that it will work on the proper back-door graph of an RCG as well.

With the help of Lemma 4.38, the algorithms can work with *almost* definite status paths, which are more convenient to handle than definite status paths because testing if a path is of definite status requires $\mathcal{O}(nm) = \mathcal{O}(n^3)$ time to verify that the nodes surrounding a definite non-collider with undirected edges on the path are not adjacent.

4.5 Adjustment in CGs

In order to reason about adjustment sets in a given chain graph $\mathcal{G}$, we need to convert $\mathcal{G}$ to a restricted chain graph, i.e., apply algorithm CONVERT-CG-TO-RCG. Proposition 3.39 proves the algorithm returns an RCG $\mathcal{R}$ with $CE(\mathcal{G}) = CE(\mathcal{R})$ in $\mathcal{O}(\deg(\mathcal{G})^2 m) = \mathcal{O}(n^4)$ if $CE(\mathcal{G}) \neq \emptyset$, so $\mathcal{G}$ and $\mathcal{R}$ have exactly the same adjustment sets. If the algorithm returns no RCG, $CE(\mathcal{G}) = \emptyset$ and there exists no valid adjustment set at all.

Afterwards the criterion of Section 4.4 can be used to find adjustment sets in $\mathcal{R}$. In the worst case, the transformation of the CG to an RCG can increase the total running time to $\mathcal{O}(n^4)$.

Figure 4.7 presents this process on an example graph.

4.6 The Final CBC

So far we have presented multiple CBCs for various classes, which one is actually The CBC?

It is the CBC for RCGs since it describes RCGs and reduces to the CBC for DAGs and MAGs in the absence of undirected edges. Let us recall

$$\begin{aligned} PCP(\mathbf{X}, \mathbf{Y}) &= \{W \in \mathbf{V} \setminus \mathbf{X} \mid W \text{ lies on a proper possibly causal path from } \mathbf{X} \text{ to } \mathbf{Y}\} \\ &= (pDe_{\overline{\mathbf{X}}}(\mathbf{X}) \setminus \mathbf{X}) \cap pAn_{\underline{\mathbf{X}}}(\mathbf{Y}). \end{aligned}$$

The back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ is constructed as $\mathcal{G}^{pbd}_{\mathbf{XY}} = (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \to PCP(\mathbf{X}, \mathbf{Y})))$.

constructive back-door criterion

Definition 4.39 (Constructive Back-Door Criterion). *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG, RCG, or adjustment amenable MAG, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables.* $\mathbf{Z}$ *satisfies the* constructive back-door criterion *relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$ if*

(a) $\mathbf{Z} \subseteq \mathbf{V} \setminus pDe(PCP(\mathbf{X}, \mathbf{Y}))$ *and*

(b) $\mathbf{Z}$ *blocks every almost definite status path from $\mathbf{X}$ to $\mathbf{Y}$ in the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$.*

Now we can combine Theorem 4.6, Theorem 4.29, and Theorem 4.36 together into a single theorem:

Theorem 4.40. *Let $\mathcal{G}$ be a DAG, RCG, or adjustment amenable MAG. Then the constructive back-door criterion holds in $\mathcal{G}$ for sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ if and only if $\mathbf{Z}$ is an adjustment set relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$.*

Proof. Definition 4.39 is identical to Definition 4.33, so it holds for RCGs and DAGs which always are RCGs. Since we only consider MAGs without undirected edges, "possible" can be ignored for our MAGs and Definition 4.39 becomes the same as Definition 4.28. □

4.7 Variations of the CBC

The CBC is surprisingly robust to modifications of its definition, so one can use slightly different criteria to characterize adjustment sets. In Lemma 4.3, we have already seen that condition (a) can be modified to forbid descendants after the removal of some edges. The following definition extends the CBC by three parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}$ that allow one to remove edges in condition (a) and from the back-door graph in condition (b).

Definition 4.41 (Parametrization of the constructive back-door criterion (CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$)))**.** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an adjustment amenable MAG or an RCG, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. Let $\mathbf{A} \subseteq \mathbf{X} \cup \mathbf{Y}$, $\mathbf{B} \subseteq \mathbf{X}$, $\mathbf{C} \subseteq pDe(PCP(\mathbf{X}, \mathbf{Y}))$. The set $\mathbf{Z}$ satisfies CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$) relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$ if*

(a') $\mathbf{Z} \subseteq \mathbf{V} \setminus De_{\overline{\mathbf{AB}}}(PCP(\mathbf{X}, \mathbf{Y}))$*, and*

(b') $\mathbf{Z}$ *d-separates $\mathbf{X}$ and $\mathbf{Y}$ in the graph $\mathcal{G}^{pbd,\mathbf{C}}_{\mathbf{XY}} := (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \to (PCP(\mathbf{X}, \mathbf{Y}) \cup \mathbf{C})))$.*

Clearly CBC(∅, ∅, ∅) = CBC. The variants CBC($\mathbf{X}$, ∅, ∅) and CBC(∅, ∅, $pDe(PCP(\mathbf{X}, \mathbf{Y}))$) might be the most interesting. Condition (a') of CBC($\mathbf{X}$, ∅, ∅) forbids the descendants of $PCP(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}_{\overline{\mathbf{X}}}$, so this condition (a') is identical to condition (a) in Definition 4.2 for DAGs. Condition (a') of CBC(∅, ∅, $pDe(PCP(\mathbf{X}, \mathbf{Y}))$) forbids exactly those nodes whose incoming edges are removed in its condition (b'), which might improve the performance in an implementation as there is no need to calculate separate sets for condition (a') and (b').

Note that the definition excludes CBC($\emptyset$, $\mathbf{Y}$, $\emptyset$), which could be considered as modifying condition (a) to forbid the descendants of $PCP(\mathbf{X}, \mathbf{Y})$ in the graph $\mathcal{G}_{\underline{\mathbf{Y}}}$. This would not lead to a valid criterion as it would allow an adjustment set $\{Z\}$ in the graph $X \to Y \to Z$, where $\{Z\}$ is not an adjustment. However, removing edges into $\mathbf{Y}$ as in the graph $\mathcal{G}_{\overline{\mathbf{Y}}}$ of CBC($\mathbf{Y}$, $\emptyset$, $\emptyset$) does not change the descendants at all, since the relevant Y are in $PCP(\mathbf{X}, \mathbf{Y})$ themselves. We can show that none of these modifications change the criterion:

Lemma 4.42. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an adjustment amenable MAG or an RCG, and let $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. Let $\mathbf{A} \subseteq \mathbf{X} \cup \mathbf{Y}$, $\mathbf{B} \subseteq \mathbf{X}$, $\mathbf{C} \subseteq pDe(PCP(\mathbf{X}, \mathbf{Y}))$. Then CBC($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$) is equivalent to the CBC.*

Proof. Let $\mathbf{Z}$ be a set that satisfies the CBC. Since $\mathbf{Z} \subseteq \mathbf{V} \setminus pDe(PCP(\mathbf{X}, \mathbf{Y}))$ due to condition (a) and we have $pDe_{\overline{\mathbf{A}\underline{\mathbf{B}}}}(PCP(\mathbf{X}, \mathbf{Y})) \subseteq pDe(PCP(\mathbf{X}, \mathbf{Y}))$, the condition (a') of the CBC($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$), i.e., $\mathbf{Z} \subseteq \mathbf{V} \setminus pDe_{\overline{\mathbf{A}\underline{\mathbf{B}}}}(PCP(\mathbf{X}, \mathbf{Y}))$ is satisfied for CBC($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$). The set $\mathbf{Z}$ separates $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, and thus also in $\mathcal{G}^{pbd,\mathbf{C}}_{\mathbf{XY}}$ because every edge or path of $\mathcal{G}^{pbd,\mathbf{C}}_{\mathbf{XY}}$ also exists in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Thus, (b') is true as well. Hence, $\mathbf{Z}$ satisfies CBC($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$).

To see the other direction, let $\mathbf{Z}$ be a set that satisfies CBC($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$), but not CBC. If $\mathbf{Z}$ does not satisfy CBC (a), there exists a node $Z \in \mathbf{V} \setminus pDe_{\overline{\mathbf{A}\underline{\mathbf{B}}}}(PCP(\mathbf{X}, \mathbf{Y}))$ that is not in $\mathbf{V} \setminus pDe(PCP(\mathbf{X}, \mathbf{Y}))$. Then there must exist a proper possibly causal path π_{XY} from $\mathbf{X}$ to $\mathbf{Y}$ on which a node $W \in \mathbf{V} \setminus \mathbf{X}$ is a possible ancestor of Z in $\mathcal{G}$, but not in $\mathcal{G}_{\overline{\mathbf{A}\underline{\mathbf{B}}}}$, i.e., there is a possibly causal path from a node of $\mathbf{X}$ over W to Z which intersects $\mathbf{A} \cup \mathbf{B}$. We can assume the nodes were chosen such that the length of the subpath between W and Z is minimal.

Let $\pi_{WY} = \pi_{XY}[W \overset{*}{\nsim} Y] = W \overset{*}{\rightarrow} Y$ denote the suffix of the path from $\mathbf{X}$ to $\mathbf{Y}$ starting in W. Note that this path might consist of only the vertex W. Additionally, for the possibly causal path from W to Z, let $\pi_{WA} = W \rightarrow A$ be its shortest prefix with $A \neq W$ which ends in $\mathbf{A} \cup \mathbf{B} \cup \mathbf{X} \subseteq \mathbf{X} \cup \mathbf{Y}$. Notice, W itself cannot be in $\mathbf{B}$, and if it is in $\mathbf{A}$, it does not change the paths. Then, from the condition (a'), we know that no vertex of π_{WA} belongs to $\mathbf{Z}$. If $A \in \mathbf{X}$, this leads to a contradiction with the condition (b') since $A \leftarrow W \overset{*}{\rightarrow} Y$ is a proper non-causal path in $\mathcal{G}^{pbd,\mathbf{C}}_{\mathbf{XY}}$ from $\mathbf{X}$ to $\mathbf{Y}$ that is not blocked by $\mathbf{Z}$. Otherwise, we have $A \in \mathbf{Y}$, so $A \in PCP(\mathbf{X}, \mathbf{Y})$ and the path from A to Z is shorter than the path from W to Z, which contradicts the choice of W.

If $\mathbf{Z}$ does not satisfy CBC (b), but satisfies CBC (a), there exists a path π between $\mathbf{X}$ and $\mathbf{Y}$ not blocked in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ by $\mathbf{Z}$ that is blocked in $\mathcal{G}^{pbd,\mathbf{C}}_{\mathbf{XY}}$ due to a removed edge $X \to C$ with $X \in \mathbf{X}, C \in \mathbf{C}$. If $X \to C$ is on π, we can assume it is the last such edge on π. If the subpath from C to Y is possibly causal, this edge is also removed in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, a contradiction. So this subpath becomes non-causal at a collider $\to C' \leftarrow$ unblocked in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, which has a descendant in $\mathbf{Z}$ that is also a descendant of C contradicting CBC (a). If the removal of an edge $X \to C$ not on π prevents the opening of a collider, C is also the ancestor of a node in $\mathbf{Z}$, which contradicts CBC (a) as well. □

4.8 The Algorithmic Framework

Using the constructive back-door criterion, we can now apply all our separation algorithms from Chapter 3 to find, test, and enumerate arbitrary, minimal, and minimum adjustment sets. Table 4.1 gives an overview of the relevant problems. Each of these problems is solved by an algorithm that calculates the set $PCP(\mathbf{X}, \mathbf{Y})$, constructs the proper back-door graph in linear

time, and solves the corresponding separator problem restricted to $\mathbf{R}' = \mathbf{R} \setminus pDe(PCP(\mathbf{X}, \mathbf{Y}))$. This works because an adjustment set relative to $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$ corresponds to a separator between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ subject to the constraint given by CBC (a). On DAGs and RCGs the resulting algorithm has a runtime that is the same as the runtime of the corresponding algorithm for separation, on MAGs the runtime increases by $\mathcal{O}(n(n+m))$ to test if the MAG is adjustment amenable.

4.8.1 Testing Adjustment Amenability

For MAGs that are not adjustment amenable, the CBC might falsely indicate that an adjustment set exists even though that set may not be valid for some represented graph. Fortunately, adjustment amenability is easily tested using the graphical criteria of Lemma 4.10. For each child D of $\mathbf{X}$ in $PCP(\mathbf{X}, \mathbf{Y})$, we can test the visibility of all edges $\mathbf{X} \to D$ simultaneously using depth-first search. This means that we can check all potentially problematic edges in time $\mathcal{O}(n + m)$. If all tests pass, we are licensed to apply the CBC as shown above. Hence, we can solve all algorithmic tasks in Table 4.1 for MAGs in the same way as for DAGs after an $\mathcal{O}(k(n + m))$ check of adjustment amenability, where $k \leq |Ch(\mathbf{X})| \leq \deg(\mathcal{G})$.

Let $Ne(V)$ denote all nodes adjacent to V, and $Sp(V)$ denote all spouses of V, i.e., nodes W such that $W \leftrightarrow V \in \mathbf{E}$. The adjustment amenability of a graph $\mathcal{M}$ w.r.t. sets $\mathbf{X}, \mathbf{Y}$ can be tested with the following algorithm:

```
function TESTADJUSTMENTAMENABILITY(M, X, Y)
    for all D in Ch(X) ∩ PCP(X, Y) do
        C := ∅
        A := ∅
        function CHECK(V)
            if C[V] then
                return A[V]
            C[V] := TRUE
            A[V] := ((Pa(V) ∪ Sp(V)) \ Ne(D) ≠ ∅)
            for all W ∈ Sp(V) ∩ Pa(D) do
                if CHECK(W) then
                    A[V] := TRUE
            return A[V]
        for all X in X ∩ Pa(D) do
            if ¬CHECK(X) then
                return FALSE
```

Analysis of the Algorithm. The algorithm checks for visibility of every edge $X \to D$ by trying to find a node Z not connected to D but connected to X via a collider path through the parents of D, according to the conditions of Lemma 4.10; note that condition (1) of Lemma 4.10 is identical to condition (2) with an empty collider path. Since CHECK performs a depth-first search by checking every node only once and then continuing to its neighbors, each iteration of the outer for-loop in the algorithm runs in linear time $\mathcal{O}(n+m)$. Therefore, the entire algorithm runs in $\mathcal{O}(k(n + m))$ where $k \leq |Ch(\mathbf{X})|$. □

4.8.2 Testing Adjustments and Minimal Adjustments

In testing problems, we are given sets of nodes $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ in a graph $\mathcal{G}$ and want to know if $\mathbf{Z}$ is an adjustment relative to $(\mathbf{X}, \mathbf{Y})$, i.e., if it satisfies the CBC. So to test the constraint of condition $CBC(a)$, we need to test if $\mathbf{Z} \cap pDe(PCP(\mathbf{X}, \mathbf{Y})) = \emptyset$. If this is not true, $\mathbf{Z}$ is not an adjustment set; otherwise, it is an adjustment set if and only if it separates $\mathbf{X}$ and $\mathbf{Y}$ in the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$.

In particular, problem TESTADJ can be solved by testing whether $\mathbf{Z} \cap pDe(PCP(\mathbf{X}, \mathbf{Y})) = \emptyset$ and if $\mathbf{Z}$ is a separator in the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$, using algorithm TESTSEP. Since $\mathcal{G}^{pbd}_{\mathbf{XY}}$ can be constructed from $\mathcal{G}$ in linear time, the total time complexity of this algorithm is $\mathcal{O}(n + m)$.

Problem TESTMINADJ can be solved by testing again if $\mathbf{Z} \cap pDe(PCP(\mathbf{X}, \mathbf{Y})) = \emptyset$ and calling algorithm TESTMINSEP to verify that $\mathbf{Z}$ is minimal within the back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$. This also leads to an optimal runtime of $\mathcal{O}(n+m)$. This approach only works since the minimal adjustment corresponds to a *minimal* separator in the proper back-door graph because every subset of an adjustment satisfies condition CBC (a). It also implies the following corollary of Lemma 3.25, which generalizes d-separation results of [TPP98] to adjustment:

Corollary 4.43. *If no single node Z can be removed from an adjustment set $\mathbf{Z}$ such that the resulting set $\mathbf{Z}' = \mathbf{Z} \setminus Z$ is no longer an adjustment set, then $\mathbf{Z}$ is minimal.*

4.8.3 Finding Adjustments and Minimal Adjustments

For the task of finding adjustment sets, we are given sets of nodes $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ in a graph $\mathcal{G}$ and need to calculate adjustments $\mathbf{Z}$ relative to $(\mathbf{X}, \mathbf{Y})$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$.

Condition CBC(a) is realized straightforwardly by imposing the constraint $\mathbf{Z} \subseteq \mathbf{R}' = \mathbf{R} \setminus pDe(PCP(\mathbf{X}, \mathbf{Y}))$, which can be given as a parameter to our separation algorithms.

Finding any adjustment set – problem FINDADJ – can be solved immediately by calling algorithm FINDSEP on the proper back-door graph. But it is also useful to have a closed-form solution for adjustment sets similar to Lemma 3.18 for separating sets. For a causal graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ and constraints $\mathbf{I}, \mathbf{R}$, we define the set

$$Adjustment(\mathbf{X}, \mathbf{Y}) = pAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \cap \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y} \cup pDe(PCP(\mathbf{X}, \mathbf{Y}))).$$

Theorem 4.44. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an adjustment amenable MAG or RCG, let $\mathbf{X}, \mathbf{Y} \subseteq V$ be distinct node sets, and $\mathbf{I}$ and $\mathbf{R}$ constraining node sets with $\mathbf{I} \subseteq \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y} \cup pDe(PCP(\mathbf{X}, \mathbf{Y})))$. Then the following statements are equivalent:*

(1) *There exists an adjustment $\mathbf{Z}$ in $\mathcal{G}$ w.r.t. $\mathbf{X}$ and $\mathbf{Y}$ with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$.*

(2) *$Adjustment(\mathbf{X}, \mathbf{Y})$ is an adjustment w.r.t. $\mathbf{X}$ and $\mathbf{Y}$.*

(3) *$Adjustment(\mathbf{X}, \mathbf{Y})$ separates $\mathbf{X}$ and $\mathbf{Y}$ in the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$.*

Proof. The implication $(3) \Rightarrow (2)$ follows directly from the criterion Definition 4.39 and the definition of $Adjustment(\mathbf{X}, \mathbf{Y})$. Since the implication $(2) \Rightarrow (1)$ is obvious, it remains to prove $(1) \Rightarrow (3)$.

Assume there exists an adjustment set $\mathbf{Z}_0$ w.r.t. $\mathbf{X}$ and $\mathbf{Y}$. From Theorem 4.40, we know that $\mathbf{Z}_0 \cap pDe(PCP(\mathbf{X}, \mathbf{Y})) = \emptyset$ and that $\mathbf{Z}_0$ separates $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Our task is to show that $Adjustment(\mathbf{X}, \mathbf{Y})$ separates $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. This follows from Lemma 3.18 used for the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ if we take $\mathbf{I}' = \mathbf{I}$, $\mathbf{R}' = \mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y} \cup pDe(PCP(\mathbf{X}, \mathbf{Y})))$. □

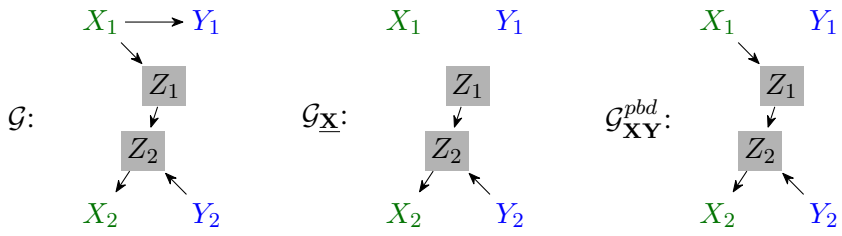


Figure 4.8: A DAG where for $\mathbf{X} = \{X_1, X_2\}$ and $\mathbf{Y} = \{Y_1, Y_2\}$, $\mathbf{Z} = \{Z_1, Z_2\}$ is a valid and minimum adjustment, but no set fulfills the back-door criterion [Pea09] (Definition 4.45), and the parents of $\mathbf{X}$ are not a valid adjustment set either.

Both algorithm FINDSEP and function *Adjustment*$(\mathbf{X}, \mathbf{Y})$ lead to a runtime of $\mathcal{O}(n+m)$ as they only need to construct the proper back-door graph and perform elementary set operations.

The remaining problems, FINDMINADJ for weakly-minimal adjustment sets, FINDMINCOSTADJ, LISTADJ, and LISTMINADJ can be solved using the corresponding algorithms for finding, resp. listing separations applied to the proper back-door graph. Since the proper back-door graph can be constructed in linear time, the time complexities to solve the problems above are the same in Table 3.1 and Table 4.1.

Unlike finding weakly-minimal adjustment sets finding strongly-minimal adjustments is NP-complete, which follows from Proposition 3.31 and the fact that the graph constructed in the proof of Proposition 3.31 contains no causal paths between X and Y, so there are no forbidden nodes and that graph is the same as its back-door graph.

4.9 Discussion and Related Work

CBC vs. Pearl's Back-Door Criterion in DAGs

Let us recall Definition 2.1 – Pearl's back-door criterion (BC) for DAGs.

back-door criterion

Definition 4.45 (Pearl's back-door criterion (BC) [Pea93; Pea09])**.** *A set of variables* $\mathbf{Z}$ *satisfies the* back-door criterion *relative to an ordered pair of variables* (X, Y) *in a DAG* $\mathcal{G}$ *if:*

(a) $\mathbf{Z} \subseteq \mathbf{V} \setminus De(X)$*, and*

(b) $\mathbf{Z}$ *blocks every path between* X *and* Y *that contains an arrow into* X*.*

Similarly, if $\mathbf{X}$ *and* $\mathbf{Y}$ *are two disjoint subsets of nodes in* $\mathcal{G}$*, then* $\mathbf{Z}$ *is said to satisfy the back-door criterion relative to* $(\mathbf{X}, \mathbf{Y})$ *if it satisfies the back-door criterion relative to any pair* (X, Y) *such that* $X \in \mathbf{X}$ *and* $Y \in \mathbf{Y}$*.*

In Definition 4.45, condition (b) is often replaced by the equivalent condition that $\mathbf{Z}$ d-separates $\mathbf{X}$ and $\mathbf{Y}$ in the back-door graph $\mathcal{G}_{\underline{X}}$.

As one can see the BC has the same structure of forbidden nodes and forbidden paths as the AC and CBC. Condition (a) of the BC is much more restrictive by forbidding all descendants of X not only those of causal paths, so there is no need to forbid non-causal paths containing colliders in condition (b). In [TL11], it was shown that for *minimal* adjustment sets in $\mathbf{X}$-loop-free DAGs the adjustment criterion and the back-door criterion of Pearl are equivalent. A DAG

Statement for arbitrary DAGs and all sets $\mathbf{Z}$:			proof
$\mathbf{Z}$ satisfies CBC	$\not\Rightarrow$	$\mathbf{Z}$ satisfies BC	$Z \leftarrow X \rightarrow Y$
$\mathbf{Z}$ satisfies CBC	$\not\Rightarrow$	$\exists \mathbf{Z}'$ satisfying BC	see Figure 4.8
$\mathbf{Z}$ satisfies CBC and $\mathbf{Z}$ is minimal	$\not\Rightarrow$	$\mathbf{Z}$ satisfies BC	see Figure 4.8
$\mathbf{Z}$ satisfies CBC and $\mathbf{Z}$ is minimal	$\not\Rightarrow$	$\exists \mathbf{Z}'$ satisfying BC	see Figure 4.8
Statement for all $\mathbf{X}$-loop-free DAGs (e.g., for singleton X) and all sets $\mathbf{Z}$:			
$\mathbf{Z}$ satisfies CBC	$\not\Rightarrow$	$\mathbf{Z}$ satisfies BC	$Z \leftarrow X \rightarrow Y$
$\mathbf{Z}$ satisfies CBC	$\Rightarrow$	$\exists \mathbf{Z}'$ satisfies BC	via minimal $\mathbf{Z}' \subseteq \mathbf{Z}$
$\mathbf{Z}$ satisfies CBC and $\mathbf{Z}$ is minimal	$\Rightarrow$	$\mathbf{Z}$ satisfies BC	see [TL11]
$\mathbf{Z}$ satisfies CBC and $\mathbf{Z}$ is minimal	$\Rightarrow$	$\exists \mathbf{Z}'$ satisfying BC	$\mathbf{Z}' = \mathbf{Z}$

Table 4.2: A summary of the relationship between the existence of a Pearl back-door (BC) adjustment set and the existence of an CBC-adjustment set in unconstrained DAGs and $\mathbf{X}$-loop-free DAGs. Symbol $\not\Rightarrow$ means that the implication does not hold, in general. On the other hand, due to the completeness property of CBC, we have that if one replaces in the left hand sides "CBC" by "BC" and in the right hand sides "BC" by "CBC", then the corresponding implications are always true.

is $\mathbf{X}$-*loop-free* for an exposure set $\mathbf{X}$ if no directed path between two different nodes of $\mathbf{X}$ contains a node not in $\mathbf{X}$. If X is a singleton, there are no two different nodes of $\mathbf{X}$ and every DAG is $\mathbf{X}$-loop-free, so the criteria are always equivalent for minimal adjustments. In this case, it is still possible that an adjustment set $\mathbf{Z}$ satisfies the CBC and not the back-door criterion, but there will always be a minimal subset $\mathbf{Z}' \subseteq \mathbf{Z}$ that satisfies the back-door criterion. Since an adjustment set satisfying the back-door criterion also satisfies the generalized back-door criterion of [MC15], and all sets of the generalized back-door criterion satisfy our CBC, all three criteria are equivalent to test the *existence* of an (minimal) adjustment set for a singleton X in DAGs. $\mathbf{X}$-*loop-free*

The situation changes if the effect of multiple exposures is estimated. Theorem 3.2.5 in [Pea09] claims that the expression for $P(\mathbf{y} \mid do(\mathbf{x}))$ is obtained by adjusting for $Pa(\mathbf{X})$ if $\mathbf{Y}$ is disjoint from $Pa(\mathbf{X})$ in graphs without latent nodes, but, as the DAG in Figure 4.8 shows, this is not true: the set $\mathbf{Z} = Pa(X_1, X_2) = \{Z_2\}$ is not an adjustment set relative to $\{X_1, X_2\}$ and $\{Y_1, Y_2\}$. In this case, one can identify the causal effect by adjusting for $\mathbf{Z} = \{Z_1, Z_2\}$ only. Indeed, for more than one exposure, no adjustment set may exist at all even without latent covariates and even though $\mathbf{Y} \cap (\mathbf{X} \cup Pa(\mathbf{X})) = \emptyset$, e.g., in the DAG $X_1 \rightarrow X_2 \leftarrow Z \leftarrow Y$ and for $\mathbf{X} = \{X_1, X_2\}$ and $\mathbf{Y} = \{Y\}$.

In the case of multiple exposures $\mathbf{X}$, it is also harder to use the back-door criterion to actually find an adjustment set. Although the back-door criterion reduces adjustment to d-separation in the back-door graph $\mathcal{G}_{\underline{X}}$, this is not the graph $\mathcal{G}_{\underline{\mathbf{X}}}$, so for each exposure $X \in \mathbf{X}$ the criterion would find a separate adjustment set, which do not lead directly to a combined adjustment set for all exposures. For an example, see Figure 4.8.

Table 4.2 summarizes some relationships between the CBC and Pearl's back-door criterion.

Adjustment in Different Classes of Graphs

[MC15] have generalized Pearl's back-door criterion to MAGs, CPDAGs, and PAGs as the so-called generalized backdoor criterion (GBC)[2].

Definition 4.46 (Generalized Backdoor Criterion, [MC15])**.** *Let* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *be disjoint set of vertices in a DAG, MAG, CPDAG, or PAG. Then* $\mathbf{Z}$ *satisfies the* generalized backdoor criterion *relative to* $(\mathbf{X}, \mathbf{Y})$ *if:*

(a) $\mathbf{Z}$ *does not contain possible descendants of* $\mathbf{X}$ *(along a definite status path); and*

(b) *For every* $X \in \mathbf{X}$*, the set* $\mathbf{Z} \cup \mathbf{X} \setminus X$ *blocks every definite status path from* X *to* $\mathbf{Y}$ *that does not start with a visible directed edge.*

For singletons $\mathbf{X} = \{X\}$ in DAGs, the GBC is identical to Pearl's back-door criterion. It improves handling of multiple exposures by allowing nodes of $\mathbf{X}$ to block non-causal paths, e.g., it correctly accepts the empty set as an adjustment set in a DAG $X_1 \leftarrow X_2 \rightarrow Y$. But it still forbids descendants of $\mathbf{X}$ that should be allowed in adjustments, like in a DAG $Z \leftarrow X \rightarrow Y$ or the example of Figure 4.8.

No complete criterion that accepts any valid adjustment set was known, till work on DAGs by [SVR10]. Table 4.3 lists the relevant publications about complete criteria.

The first complete criterion for MAGs was given by us with the CBC of Definition 4.28 in [ZLT14]. It was also the first constructive complete criterion (even for DAGs) that yields a closed-form function *Adjustment*$(\mathbf{X}, \mathbf{Y})$ that returns an adjustment accepted by the criterion if and only if an adjustment exist. We have generalized it further to chain graphs, CPDAGs, and RCGs with Definition 4.33 [ZL16b], while [Per+15; Per+16] have generalized it to CPDAGs and PAGs in their generalized adjustment criterion (GAC).

Definition 4.47 (Generalized Adjustment Criterion, [Per+15; Per+16])**.** *Let* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *be pairwise disjoint node sets in a DAG, CPDAG, MAG, or PAG* $\mathcal{G}$*. Then* $\mathbf{Z}$ *satisfies the* generalized adjustment criterion *relative to* $(\mathbf{X}, \mathbf{Y})$ *if:*

(a) $\mathbf{Z}$ *does not contain a possible descendant of any* $W \in \mathbf{V} \setminus \mathbf{X}$ *on a proper possibly causal path from* $\mathbf{X}$ *to* $\mathbf{Y}$*,*

(b) *all proper definite status non-causal paths from* $\mathbf{X}$ *to* $\mathbf{Y}$ *are blocked by* $\mathbf{Z}$*, and*

(c) $\mathcal{G}$ *is adjustment amenable relative to* $(\mathbf{X}, \mathbf{Y})$.

For DAGs, CPDAGs, and MAGs the GAC is equivalent to our CBC despite the use of definite status paths rather than almost definite status paths in condition (b).

As mentioned by [Per+15; Per+16] the criterion can be stated using walks rather than paths:

Definition 4.48 (GAC with walks)**.** *Let* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *be pairwise disjoint node sets in a DAG, CPDAG, MAG, or PAG* $\mathcal{G}$*. Then* $\mathbf{Z}$ *satisfies the generalized adjustment criterion relative to* $(\mathbf{X}, \mathbf{Y})$ *if:*

(a) *no proper causal walk from* $\mathbf{X}$ *to* $\mathbf{Y}$ *is blocked by* $\mathbf{Z}$*, and*

[2] [MC15] actually writes "backdoor" while Pearl uses "back-door". See Appendix B for a definition of PAGs.

Graph class	First complete criterion	First complete constructive criterion
DAGs	Shpitser et al. [SVR10]	van der Zander et al. [ZLT14]
MAGs	van der Zander et al. [ZLT14]	van der Zander et al. [ZLT14]
CPDAGs	Perković et al. [Per+15; Per+18]	van der Zander et al. [ZL16b]
PAGs	Perković et al. [Per+15; Per+18]	Perković et al. [Per+16; Per+18]
CGs	van der Zander et al. [ZL16b]	van der Zander et al. [ZL16b]
maximal PDAGs	Perković et al. [PKM17]	Perković et al. [PKM17]

Table 4.3: An overview of our results and and related works for directed acyclic graphs (DAGs), maximal ancestral graphs (MAGs), completed partially directed acyclic graphs (CPDAGs), partial ancestral graphs (PAG), chain graphs (CGs), and maximally oriented partially directed acyclic graphs (PDAGs). All works provide sound criteria, i.e., criteria only satisfied by adjustment sets. A criterion is complete if every adjustment set satisfies it. It is constructive if it leads to an efficient algorithm for constructing an adjustment set. PAGs and PDAGs are described in Appendix B.

(b) *all proper definite status non-causal walks from* $\mathbf{X}$ *to* $\mathbf{Y}$ *are blocked by* $\mathbf{Z}$*, and*

(c) $\mathcal{G}$ *is adjustment amenable relative to* $(\mathbf{X}, \mathbf{Y})$.

A further generalization was discovered towards maximal PDAGs [PKM17]. Unlike RCGs, PDAGs and maximal PDAGs are allowed to contain semi-directed cycles. The maximal PDAG can be constructed from a PDAG using four orientation rules.

Definition 4.49 (b-adjustment criterion, [PKM17])**.** *Let* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *be pairwise disjoint node sets in a maximal PDAG* $\mathcal{G}$*. Then* $\mathbf{Z}$ *satisfies the* b-adjustment criterion *relative to* $(\mathbf{X}, \mathbf{Y})$ *if:*

(a) $\mathbf{Z}$ *does not contain a b-possibly descendant of any* $W \notin \mathbf{X}$ *on a proper b-possibly causal path from* $\mathbf{X}$ *to* $\mathbf{Y}$*,*

(b) *all proper definite status b-non-causal paths from* $\mathbf{X}$ *to* $\mathbf{Y}$ *are blocked by* $\mathbf{Z}$*, and*

(c) $\mathcal{G}$ *is adjustment amenable relative to* $(\mathbf{X}, \mathbf{Y})$*, i.e., all proper b-possibly causal paths from* $\mathbf{X}$ *to* $\mathbf{Y}$ *start with a directed edge out of* $\mathbf{X}$.

The criterion itself is almost the same as all the other adjustment criteria, but a huge difference occurs in the definition of b-possibly causal as the definition depends on edges that are not on the path: A path $V_0, \dots, V_k$ is *b-possibly causal* if there exists no edge $V_i \leftarrow V_j$ for $i < j$ in the graph. A path that is not b-possibly causal, is *b-non-causal*.

B-non-causal paths are difficult to search in the PDAG, so the best way to check condition (b) of the b-adjustment criterion is to construct a DAG represented by the PDAG in $\mathcal{O}(n + m)$ and then use our algorithms in the DAG [PKM17].

[HPM19] have studied the accuracy of adjustment sets in linear SEMs and given an algorithm to improve the accuracy of a given adjustment set by pruning it, i.e., by removing unnecessary variables. Our algorithm FINDMINSEP can be modified to perform such a pruning [ZL19] even though the adjustment sets after the pruning are not necessarily minimal.

Conclusions

We have developed efficient algorithms for adjustment sets in DAGs, AGs, and RCGs, solving the problems of Table 4.1.

Future research could examine if the algorithms can be generalized to other graphical models, like PAGs or maximal PDAGs, while keeping the same efficient runtime.

The runtime of the algorithm to find minimum adjustment sets can probably be improved from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm)$ if one obtains a faster algorithm to find minimum separators.

5 A Comparison of Non-Parametric Identification Methods

In this chapter, we compare our constructive back-door criterion with Pearl's back-door criterion and the do-calculus empirically. For this purpose, we generate a large number of random DAGs and then count how many adjustment sets can be identified by the CBC, the BC, or the IDC algorithm of [SP06b; SP06a] that solves the identification problem as determined by the do-calculus.

The experiments have shown that in many DAGs the causal effect can be identified by plain (simple) formulas even though covariate adjustment is not possible, so it is not necessary to use the full power of the do-calculus in these cases. We present these plain formulas in Section 5.1 before the experiments themselves. Then we include these cases in the comparison with the other algorithms.

In Section 5.2, we describe how the random graphs are generated and analyze the results. In the short Section 5.3, we convert the DAGs to MAGs and count how many adjustment sets are also valid in the MAGs. In Section 5.4, we do the same after converting the DAGs to CPDAGs.

Scientific Contribution. The comparison between identification algorithms for DAGs and AGs has been published as [ZLT19]. Here we extend it with results for CPDAGs.

5.1 Beyond Covariate Adjustment in DAGs

While our complete adjustment criterion is guaranteed to find all instances in which a causal effect can be identified via covariate adjustment, it is well known that not all identifiable effects are also identifiable via adjustment. The do-calculus [Pea09] is a complete method that characterizes all identifiable causal effects in DAGs, but which comes at a price of substantially increased formula and runtime complexity. In this section, we show that many cases in which covariate adjustment is not applicable do not require the full power of the do-calculus either.

Specifically, we provide three propositions that permit the identification of total causal effects in the following three cases (which are not mutually exclusive) as shown in Figure 5.1: (1) $\mathbf{X}$ does not have a causal effect on $\mathbf{Y}$; (2) $\mathbf{X} = X$ is a singleton and all its parents are observed; (3) $\mathbf{X}$ and $\mathbf{Y}$ partition $\mathbf{V}$. While in each case the adjustment criterion may or may not be applicable, our propositions show that identifiability is always guaranteed, and the total effect can be computed by reasonably simple formulas. Moreover, each proposition provides an easy algorithm for testing whether the corresponding case applies.

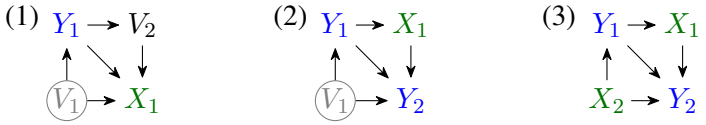


Figure 5.1: The three cases analyzed in this section, Section 5.1. Exposure and outcome nodes are marked as X and Y; latent nodes are shown in gray. In case (1), the causal effect is given by $P(y_1 \mid do(x_1)) = P(y_1)$, in case (2) by $P(y_1, y_2 \mid do(x_1)) = P(y_1)P(y_2 \mid x_1, y_1)$, and in case (3) by $P(y_1, y_2 \mid do(x_1, x_2)) = P(y_1 \mid x_2)P(y_2 \mid y_1, x_1, x_2)$.

5.1.1 Identification by plain formulas

One case in which identification is trivial is if there is no causal effect of X on Y at all. When all non-causal paths between X and Y can be blocked, then this case is covered by the CBC; however, if there is a non-causal path consisting solely of unobserved nodes, then the (void) effect is not obtainable through the adjustment formula. In such cases, however, we simply have

plain formula

$P(\mathbf{y} \mid do(\mathbf{x})) = P(\mathbf{y})$, which we will call the *plain formula*. The following proposition provides a characterization of all cases in which this plain formula works in terms of d-separation.

Proposition 5.1. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG, let $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ be disjoint subsets of variables, and let $\mathbf{R} \subseteq \mathbf{V}$ be an arbitrary set of observed variables, with $\mathbf{X} \cup \mathbf{Y} \subseteq \mathbf{R}$. Then $\mathbf{X}$ and $\mathbf{Y}$ are d-separated in $\mathcal{G}_{\overline{\mathbf{X}}}$, expressed symbolically as*

$$(\mathbf{Y} \perp\!\!\!\perp \mathbf{X})_{\mathcal{G}_{\overline{\mathbf{X}}}} \tag{5.1}$$

if and only if the effect of intervention of $\mathbf{X}$ on $\mathbf{Y}$ is given by the plain formula $P(\mathbf{y} \mid do(\mathbf{x})) = P(\mathbf{y})$, i.e., there is no causal effect from $\mathbf{X}$ on $\mathbf{Y}$. Particularly, if $Y \in An(X)$, then $(Y \perp\!\!\!\perp X)_{\mathcal{G}_{\overline{X}}}$, and thus $P(y \mid do(x)) = P(y)$.

Proof. The soundness of the statement follows directly by the application of rule 3 (intervention/deletion of actions in Theorem 2.2; for the precise definition of the do-calculus rules see Theorem 3.4.1 in [Pea09]). The completeness of the statement can be shown similarly to the completeness of the adjustment criterion [SVR10]. If $\mathbf{Y}$ and $\mathbf{X}$ are not d-separated in $\mathcal{G}_{\overline{\mathbf{X}}}$, there exists a shortest causal path $X \dashrightarrow Y$ for $X \in \mathbf{X}, Y \in \mathbf{Y}$. In the subgraph $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$ consisting only of this path, the causal effect is given by an empty adjustment set $P(y \mid do(x)) = P(y \mid x)$. If we take a model P' where $P'(y \mid x) \neq P'(y)$ for some values x, y, like e.g., in a model on binary variables X, Y with

$$P'(x) = \frac{1}{2} \quad \text{and} \quad P'(y \mid x) = \begin{cases} \frac{1}{3} & x = y, \\ \frac{2}{3} & x \neq y, \end{cases}$$

then the causal effect is not given by $P'(y)$. This model can be extended to a model P on the full graph $\mathcal{G}$ by assuming all other variables are independent, i.e., $P(\mathbf{v}) = (\frac{1}{2})^{|\mathbf{V} \setminus \mathbf{V}'|} P'(\mathbf{v}')$. This model is compatible with $\mathcal{G}$ (though not faithful, but faithfulness is not required) and we have

$$\begin{aligned} P(\mathbf{y} \mid do(\mathbf{x})) &= P(\mathbf{y} \setminus Y) P(y \mid do(x)) \\ &= P(\mathbf{y} \setminus Y) P'(y \mid do(x)) \neq P(\mathbf{y} \setminus Y) P'(y) \\ &= P(\mathbf{y} \setminus Y) P(y) \\ &= P(\mathbf{y}). \end{aligned}$$

□

5.1.2 Identification by generalized parent adjustment

Another case that permits the identification of causal effects using simple formulas occurs if the exposure $\mathbf{X} = X$ is a singleton and all its parents are observed, i.e., $Pa(X) \subseteq \mathbf{R}$. Then adjusting for the parents of X blocks all biasing paths and suffices for identification, but one needs to be careful as there might be variables $\mathbf{Y}_{pa} = Pa(X) \cap \mathbf{Y}$ that are both parents and outcome nodes. Proposition 5.2 below shows that in this case the causal effect is given by $P(\mathbf{y} \mid do(x)) = \sum_{\mathbf{z}} P(\mathbf{z}, \mathbf{y}_{pa}) P(\mathbf{y}_{np} \mid x, \mathbf{z}, \mathbf{y}_{pa})$, where $\mathbf{Y}_{pa} \cup \mathbf{Y}_{np}$ is a partition of $\mathbf{Y}$ and $\mathbf{Y}_{pa} \cup \mathbf{Z}$ is a partition of $Pa(X)$.

This is a slight generalization of identification via adjustment: we still sum over the values of variables $\mathbf{Z}$ and multiply the conditional probability with a factor, but rather than multiplying with the probability of the same variables $P(\mathbf{z})$ that are used in the sum, we multiply with a factor $P(\mathbf{z}, \mathbf{y}_{pa})$ involving additionally the variables in $\mathbf{Y}_{pa}$.

The situation is even simpler if Y is also a singleton. Then one of the sets $\mathbf{Y}_{pa}, \mathbf{Y}_{np}$ vanishes, so there are only two cases: either $Y \notin Pa(X)$ and $Pa(X)$ is an adjustment set [Pea09, Theorem 3.2.2], or $Y \in Pa(X)$ and no adjustment exists, but the causal effect is identified as $P(y \mid do(x)) = P(y)$. One can see that in the case $Y \in An(X) \setminus Pa(X)$ the effect of an intervention $do(X = x)$ can be given both by the plain expression $P(y \mid do(x)) = P(y)$ and by adjusting the parents of X.

Proposition 5.2. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG, and let $X \in \mathbf{V}$ be a node with observed parents $Pa(X) \subseteq \mathbf{R}$ and $\mathbf{Y} \subseteq \mathbf{V} \setminus X$. Furthermore, let $\mathbf{Y}_{pa} = \mathbf{Y} \cap Pa(X)$ and let $\mathbf{Y}_{np} = \mathbf{Y} \setminus Pa(X)$ be a partition of $\mathbf{Y} = \mathbf{Y}_{pa} \cup \mathbf{Y}_{np}$, and let $\mathbf{Z} = Pa(X) \setminus \mathbf{Y}_{pa}$ form with $\mathbf{Y}_{pa}$ a partition of $Pa(X) = \mathbf{Y}_{pa} \cup \mathbf{Z}$. Then*

$$P(\mathbf{y} \mid do(x)) = \begin{cases} P(\mathbf{y}_{pa}) P(\mathbf{y}_{np} \mid x, \mathbf{y}_{pa}) & \text{if } \mathbf{Z} = \emptyset, \text{ i.e., if } Pa(X) \subseteq \mathbf{Y}, \\ \sum_{\mathbf{z}} P(\mathbf{z}, \mathbf{y}_{pa}) P(\mathbf{y}_{np} \mid x, \mathbf{z}, \mathbf{y}_{pa}) & \text{if } \mathbf{Z} \neq \emptyset, \end{cases}$$

where $P(\mathbf{y}_{pa})$ (resp. $P(\mathbf{y}_{np} \mid x, \mathbf{y}_{pa})$ and $P(\mathbf{y}_{np} \mid x, \mathbf{z}, \mathbf{y}_{pa})$) should be read as 1 if $\mathbf{Y}_{pa} = \emptyset$ (resp. $\mathbf{Y}_{np} = \emptyset$).

Proof. This follows from a straightforward calculation using the do-calculus:

$$\begin{aligned} P(\mathbf{y} \mid do(x)) &= P(\mathbf{y}_{pa}, \mathbf{y}_{np} \mid do(x)) \\ &= \sum_{\mathbf{z}} P(\mathbf{z}, \mathbf{y}_{pa}, \mathbf{y}_{np} \mid do(x)) \\ &= \sum_{\mathbf{z}} P(\mathbf{z}, \mathbf{y}_{pa} \mid do(x)) P(\mathbf{y}_{np} \mid do(x), \mathbf{z}, \mathbf{y}_{pa}) \\ &= \sum_{\mathbf{z}} P(\mathbf{z}, \mathbf{y}_{pa}) P(\mathbf{y}_{np} \mid do(x), \mathbf{z}, \mathbf{y}_{pa}) && \text{do-calculus rule 3: } (\mathbf{Y}_{pa}, \mathbf{Z} \perp\!\!\!\perp X) \text{ in } \mathcal{G}_{\overline{X}} \\ &= \sum_{\mathbf{z}} P(\mathbf{z}, \mathbf{y}_{pa}) P(\mathbf{y}_{np} \mid x, \mathbf{z}, \mathbf{y}_{pa}). && \text{do-calculus rule 2: } (\mathbf{Y}_{np} \perp\!\!\!\perp X \mid \mathbf{Z}, \mathbf{Y}_{pa}) \text{ in } \mathcal{G}_{\underline{X}} \end{aligned}$$

□

5.1.3 Identification when X and Y partition V

Here we consider the case of DAGs in which $\mathbf{X}$ and $\mathbf{Y}$ partition the set of variables $\mathbf{V}$, which implies that there are no unobserved nodes. Again, in this case the CBC may not be applicable as there may be an arrow from $\mathbf{Y}$ to $\mathbf{X}$, but still the causal effect can be given by a closed-form solution as we present below.

Proposition 5.3. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and $\mathbf{X}, \mathbf{Y} \subset \mathbf{V}$ be a partition of $\mathbf{V} = \mathbf{X} \cup \mathbf{Y}$. The following statements hold:*

(a) *The causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is given by*

$$P(\mathbf{y} \mid do(\mathbf{x})) = \prod_{Y \in \mathbf{Y}} P(Y = y \mid pa_Y),$$

where pa_Y denotes the values of the parents of Y.

(b) *If no edge $X \to Y$ with $X \in \mathbf{X}, Y \in \mathbf{Y}$ exists, the causal effect is also given by the plain formula*

$$P(\mathbf{y} \mid do(\mathbf{x})) = P(\mathbf{y}).$$

(c) *The causal effect can be identified by adjustment if and only if no edge $Y \to X$ with $X \in \mathbf{X}, Y \in \mathbf{Y}$ exists.*

(d) *If identification by adjustment is possible, the adjustment set is $\mathbf{Z} = \emptyset$ and the causal effect of $\mathbf{X}$ on $\mathbf{Y}$ is given by*

$$P(\mathbf{y} \mid do(\mathbf{x})) = P(\mathbf{y} \mid \mathbf{x}).$$

Proof. Statement (a) follows from the definition of the causal effect:

$$\begin{aligned}
P(\mathbf{y} \mid do(\mathbf{x})) &= \sum_{\mathbf{x}'} P(\mathbf{X} = \mathbf{x}', \mathbf{Y} = \mathbf{y} \mid do(\mathbf{X} = \mathbf{x})) \\
&= P(\mathbf{x}, \mathbf{y} \mid do(\mathbf{x})) && \mathbf{x} \neq \mathbf{x}' \text{ makes the causal effect inconsistent} \\
&= P(\mathbf{v} \mid do(\mathbf{x})) \\
&= \prod_{Y_j \in \mathbf{Y}} P(y_j \mid pa_j) && \text{definition of the causal effect, Equation 2.2} \\
&= \prod_{Y \in \mathbf{Y}} P(Y = y \mid pa_Y).
\end{aligned}$$

For Statements (b) and (c), note that edges $\mathbf{X} \to \mathbf{X}$ or $\mathbf{Y} \to \mathbf{Y}$ do not affect d-connectedness between $\mathbf{X}$ and $\mathbf{Y}$. Hence, with the assumption in Statement (b), the sets $\mathbf{X}$ and $\mathbf{Y}$ are d-separated in the graph $\mathcal{G}_{\overline{\mathbf{X}}}$, where all edges $\mathbf{Y} \to \mathbf{X}$ and $\mathbf{X} \to \mathbf{X}$ are deleted. Then we know from Proposition 5.1 that the causal effect is identified by a plain formula. Since no node is outside of $\mathbf{X} \cup \mathbf{Y}$, the only possible adjustment set is $\mathbf{Z} = \emptyset$, which is Statement (d). Finally, an adjustment set $\mathbf{Z} = \emptyset$ always satisfies the first condition of the CBC. The back-door graph is formed by removing all edges $\mathbf{X} \to \mathbf{Y}$ as those edges form a causal path of length one. Thus, $\mathbf{Z}$ is an adjustment set if and only if no edge $\mathbf{Y} \to \mathbf{X}$ exists, which is Statement (c). □

When $\mathbf{V} = \mathbf{X} \cup \mathbf{Y}$, the R package `causaleffect` [TK17], which we used in our experiments described in the next section, returns the formula $P(\mathbf{y} \mid do(\mathbf{x})) = \prod_{Y \in \mathbf{Y}} P(y \mid an'_Y)$, where an'_Y denotes the values of the variables in $An(Y) \setminus Y$, when the package is configured to be fast and find *any* identification formula rather than a short one. Thus, it is worth to mention that $\prod_{Y \in \mathbf{Y}} P(y \mid an'_Y) = \prod_{Y \in \mathbf{Y}} P(y \mid pa_Y)$ because the parents of a node Y block all paths from other ancestors of Y to Y.

5.2 Empirical Analysis of Identifiability by Adjustment in DAGs

As mentioned before, not all identifiable total effects are identifiable via adjustment, but if covariate adjustment is possible, then it is usually preferred to other methods due to its benign statistical properties. This raises the question of how often we will actually have to go beyond covariate adjustment when identifying causal effects. The completeness and algorithmic efficiency of the CBC allowed us to perform an empirical analysis of identifiability via adjustment in random graphs, including graphs of substantial size.

The basic setup of our experiments is as follows. We (1) generate a random graph; (2) set nodes to be unobserved at random; (3) choose random disjoint subsets $\mathbf{X}, \mathbf{Y}$ of pre-specified cardinalities from the observed nodes; and (4) test whether $P(\mathbf{y} \mid do(\mathbf{x}))$ is identifiable in the resulting graph. We test the identifiability of $P(\mathbf{y} \mid do(\mathbf{x}))$ using four increasingly powerful criteria: (1) Pearl's back-door criterion [Pea09]; (2) the CBC; (3) an extended version of the CBC that also covers the special cases discussed in Section 5.1; and (4) the do-calculus, which characterizes all effects that are identifiable at all. Full details are given below.

We included the classic back-door criterion (Definition 4.45) in our analysis because it is very present in the applied literature on DAGs (e.g., [Elw13]) whereas the generalized version (GBC, Definition 4.46) is still barely mentioned. It is known that the back-door criterion is not complete and can thus fail to identify an adjustment set, which raises the question of how often the back-door criterion fails to find an adjustment set when our CBC criterion succeeds. In Section 4.9, it was shown that this is never the case for a singleton X (although the CBC may still find more adjustment sets than the BC).

Our extensions to the CBC in Section 5.1 were motivated by our observation from preliminary experiments that many cases where an effect is not identifiable by adjustment are anyway identifiable due to simple reasons like the absence of any causal path from $\mathbf{X}$ to $\mathbf{Y}$, which can be addressed quite easily without invoking the full machinery of the do-calculus.

We now proceed to give the technical details of how we set up our empirical analysis.

5.2.1 Instance Generation

We generate random DAGs with n nodes, of which $2k$ nodes are chosen to create sets $\mathbf{X}$ and $\mathbf{Y}$, each node is unobserved with a probability of approximately $P(\textit{unobserved})$, and edges are randomly placed with a probability $\frac{l}{n-1}$. These four parameters form a parametrization tuple that characterizes all DAGs of one experiment:

$$(n, l, k, P(\textit{unobserved})). \tag{5.2}$$

More specific, we set the following parameters for the experiments. The number of variables $\mathbf{V}$ is one of

$$|\mathbf{V}| = n \in \{10, 25, 50, 100, 250, 500, 1000, 2000\}.$$

These variables are divided into four sets: ordinary observed nodes $\mathbf{R}$, unobserved nodes $\mathbf{V} \setminus \mathbf{R}$, exposure nodes $\mathbf{X} \subseteq \mathbf{R}$, and outcome nodes $\mathbf{Y} \subseteq \mathbf{R}$ (with $\mathbf{X} \cap \mathbf{Y} = \emptyset$) depending on parameters

$$P(\textit{unobserved}) \in \{0, 0.25, 0.5, 0.75\} \quad \text{and} \quad |\mathbf{X}| = |\mathbf{Y}| = k \in \left\{1, 2, 5, \lfloor\sqrt{n}\rfloor, \lfloor 0.1n \rfloor\right\}.$$

To select those sets, we proceed as follows: Initially mark all variables in $\mathbf{V}$ as observed. Next, for every node mark it as unobserved with probability $P(\textit{unobserved})$ until all nodes are considered or the number of nodes which remain observed reaches the threshold value $2k$.

Finally, from the observed $\mathbf{R}$ pick randomly two disjoint subsets $\mathbf{X}$ and $\mathbf{Y}$ of size k. The expected size of $\mathbf{R}$ is bounded by $\mathbb{E}[|\mathbf{R}|] > n \cdot (1 - P(\textit{unobserved}))$, with the difference being very small for $n \gg 2k$, but substantial for $n \gtrapprox 2k$. For example, for $n = 10$ and $k = |\mathbf{X}| = |\mathbf{Y}| = 5$, all nodes are in $\mathbf{R} = \mathbf{X} \cup \mathbf{Y} = \mathbf{V}$ regardless of the probability $P(\textit{unobserved})$ – the case discussed already in Section 5.1.3.

The edges are chosen independently with individual probabilities $P(\textit{edge}) = \frac{l}{n-1}$, parameterized by $l = 2, 5, 10, 20$:

$$P(\textit{edge}) \in \left\{ \frac{2}{n-1}, \frac{5}{n-1}, \frac{10}{n-1}, \frac{20}{n-1} \right\}.$$

For small n, the probabilities are capped at 1. For example, for $n = 10$ a parameter $P(\textit{edge}) = \min\{\frac{10}{10-1}, 1\} = 1$ will only generate complete graphs. Parameter l describes the expected number of neighbors of a node in a generated DAG. This leads to an expected number of edges in generated graphs

$$\mathbb{E}[m] \approxeq n, \text{ (resp. } 2.5n, 5n, 10n, \text{depending on parameter } l)$$

as there are $\frac{n(n-1)}{2}$ possible edges in a DAG of n nodes, each existing with probability $P(\textit{edge})$.

In this section, we will report our results in detail only for $P(\textit{unobserved}) \in \{0, 0.75\}$. The remaining cases are shown in Appendix A

We have generated 10 000 graphs for each different parameter tuple using the function `GraphGenerator.randomDAG` of our DAGitty library [Tex+16] in node.js. Figure 5.2 shows example instances sampled for $n = 10$ and illustrates the four cases we are interested in.

5.2.2 Algorithms

The main goal of our experiments was to examine the influence of the instance complexity, like the density of a DAG, numbers of exposures and outcomes, and the ratio of unobserved to observed variables, on the identifiability by adjustment compared to general identifiability. Throughout, we use the following abbreviations for the algorithms we examine:

CBC: our constructive back-door criterion (Definition 4.5, Theorem 4.44). We have used our DAGitty library, i.e., the function `GraphAnalyzer.canonicalAdjustmentSet`, which implements algorithm FINDADJ based on the CBC. We have also tested another implementation of our CBC criterion, the `gac` function of the R package `pcalg` [Kal+12].

CBC$^+$: combination of CBC and plain formula (Proposition 5.1). We implement the plain formula using the DAGitty function `GraphAnalyzer.dConnected`, which implements algorithm TESTSEP (Proposition 3.17).

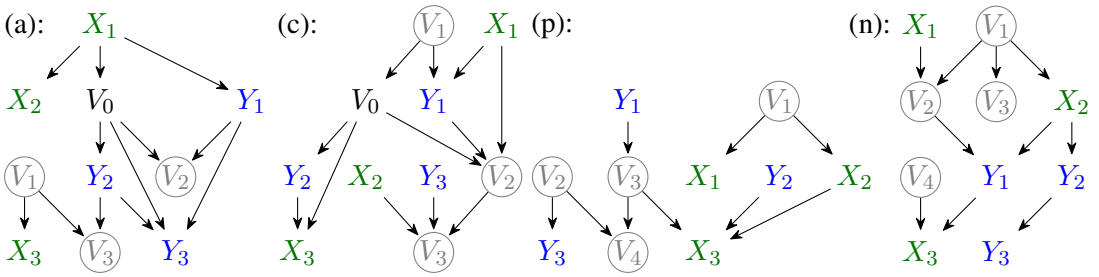


Figure 5.2: Example DAGs sampled for the parameter tuple $n = 10, P(\textit{edge}) = 2/9, P(\textit{unobserved}) = 0.5$, and $k = |\mathbf{X}| = |\mathbf{Y}| = 3$. Nodes are relabeled such that exposures are called X_1, X_2, X_3, outcomes are called Y_1, Y_2, Y_3, and all nodes except V_0 are unobserved. Case (a)djustment is identified by using the empty set and by the formula $\sum_{v_0}[P(y_1|x_1)P(v_0|x_1)P(y_3|x_1, y_1, v_0, y_2)P(y_2|x_1, v_0)]$ found by the ID-algorithm. Instance (c)omplex is identified by the complex formula $\sum_{v_0}[P(v_0|x_1)P(y_1|x_1, v_0)P(y_2|v_0)P(y_3)]$ and instance (p)lain is identified by the plain formula $P(y_1, y_2, y_3)$ although in this case no adjustment set exists. The final example is (n)onidentifiable.

BC: Pearl's back-door criterion (Definition 4.45). It has been shown that if an adjustment set $\mathbf{Z}$ that satisfies BC exists, it can be found by removing all descendants of $\mathbf{X}$ from $\mathbf{Z}$ [Per+16]. This means we can implement BC by trivial post-processing of the CBC output.

IDC: general identifiability as determined by do-calculus (see Theorem 2.2). Specifically, we use the IDC algorithm by Shpitser and Pearl [SP06a], which is well known to be complete for the identification of causal effects [SP06b; HV06], meaning that the algorithm computes a formula involving only the pre-intervention distribution that expresses the causal effect if such a formula exists; otherwise, it outputs that identification is impossible. Our experiments are based on the IDC implementation provided by the R package `causaleffect` [TK17]. Due to its high time complexity, we were only able to use this algorithm for small instances.

5.2.3 Results

The results for all methods and parameters n, k, l described above are shown in Table 5.1 (for the case $P(\textit{unobserved}) = 0$) and in Table 5.2 ($P(\textit{unobserved}) = 0.75$). We now discuss the results in more detail.

Identification by adjustment sets or plain formulas. Tables 5.1 and 5.2 provide counts for instances identifiable by adjustment alone (columns CBC) or by adjustment enhanced by using the plain formula (CBC$^+$). The number of effects only identified by the plain formula, but not by CBC, is thus given by the difference between these columns.

Figure 5.3 summarizes the counts for CBC and CBC$^+$ reported in Table 5.1 and 5.2 for $k = 1, 2, 5$ and $n \geq 25$. We omit the instances with $n = 10$ since for $k = 5$ these cases were discussed separately in Section 5.1.3. Moreover, for parameter values $l = 10$ and $l = 20$, the individual probabilities for edge selection, $P(\textit{edge}) = \max\{l/(n-1), 1\}$, imply that every node has $9 < l$ neighbors while in our analyses we want that l specifies the expected number of neighbors of a node.

		$l=2$			$l=5$			$l=10$			$l=20$		
n	k	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**
10	1	8893	8893	10000	7205	7205	10000	5034	5034	10000	4934	4934	10000
10	2	5543	6061	8618	1033	1980	4322	0	660	2197	0	686	2417
10	3	2359	3395	5817	57	548	1425	0	168	663	0	174	689
10	5	200	886	1712	0	108	216	0	36	71	0	31	65
25	1	9573	9573	10000	8936	8936	10000	7905	7905	10000	5843	5843	10000
25	2	8117	8247	9651	4243	4735	7003	1033	1553	3674	70	401	2118
25	3	6013	6424	8520	1203	1852	3524	46	212	1046	0	34	587
25	5	2298	3021	5055	39	243	646	0	11	93	0	1	53
50	1	9832	9832	10000	9476	9476	10000	8997	8997	10000	7832	7832	10000
50	2	9095	9128	9882	6688	6938	8388	2657	3049	4927	527	866	2729
50	5	5104	5535	7489	462	799	1613	3	16	198	0	2	58
50	7	2473	3120	4799	27	119	302	0	1	25	0	0	6
100	1	9907	9907	10000	9783	9783	10000	9494	9494	10000	8966	8966	10000
100	2	9585	9593	9971	8262	8353	9165	4600	4834	6162	1507	1762	3492
100	5	7425	7591	9090	1932	2336	3441	43	102	393	1	4	92
100	10	2499	3040	4479	15	48	137	0	0	2	0	0	0
250	1	9947	9947	10000	9894	9894	10000	9774	9774	10000	9621	9621	10000
250	2	9835	9840	9991	9284	9327	9696	6569	6689	7358	3151	3285	4502
250	5	8956	8994	9807	5051	5325	6261	469	544	994	7	17	164
250	15	3102	3537	4864	18	32	58	0	0	1	0	0	0
250	25	319	536	731	0	0	0	0	0	0	0	0	0
500	1	9977	9977	10000	9946	9946	10000	9883	9883	10000	9799	9799	10000
500	2	9923	9923	9996	9674	9684	9872	7704	7750	8116	4184	4266	4988
500	5	9477	9490	9948	7249	7368	8069	1170	1265	1686	43	48	216
500	22	3012	3288	4413	3	14	17	0	0	0	0	0	0
500	50	10	29	31	0	0	0	0	0	0	0	0	0
1000	1	9990	9990	10000	9973	9973	10000	9942	9942	10000	9885	9885	10000
1000	2	9965	9966	10000	9844	9845	9952	8416	8434	8640	5130	5173	5577
1000	5	9734	9736	9986	8679	8726	9173	2310	2396	2686	136	149	319
1000	32	2923	3191	4163	2	6	7	0	0	0	0	0	0
1000	100	0	0	0	0	0	0	0	0	0	0	0	0
2000	1	9999	9999	10000	9988	9988	10000	9972	9972	10000	9938	9938	10000
2000	2	9973	9973	10000	9940	9940	9981	9023	9029	9119	5928	5954	6156
2000	5	9880	9880	9996	9450	9471	9713	3608	3648	3869	287	300	469
2000	45	3000	3210	4122	4	8	8	0	0	2	0	0	0
2000	200	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.1: Numbers of instances for $P(\textit{unobserved}) = 0$, i.e., all variables are observed, that are identifiable by use of BC, CBC, or CBC$^+$ (as defined in Section 5.2.2). Gray cells highlight where the CBC was able to identify at least 400 more graphs than the BC. Since all variables are observed, all instances are identifiable, thus IDC is not used in this table.

		$l=2$				$l=5$				$l=10$				$l=20$			
n	k	**BC**	**CBC**	**CBC$^+$**	**IDC**	**BC**	**CBC**	**CBC$^+$**	**IDC**	**BC**	**CBC**	**CBC$^+$**	**IDC**	**BC**	**CBC**	**CBC$^+$**	**IDC**
10	1	6333	6333	9604	9609	1935	1935	7475	7476	978	978	5944	5944	936	936	5877	5877
10	2	2008	2339	6889	8740	103	228	2854	4137	0	113	1721	2260	0	114	1752	2294
10	3	610	980	4193	8056	0	21	1061	1995	0	9	512	763	0	10	547	789
10	5	185	859	1756	10000	0	98	190	10000	0	43	76	10000	0	26	75	10000
25	1	8414	8414	9923	9930	3647	3647	8727	8742	1340	1340	6884	6888	557	557	5696	5696
25	2	5164	5331	8939	9731	601	728	4630	6417	77	130	2501	3469	4	41	1847	2299
25	3	2350	2632	6958	9270	73	144	2141	4327	2	17	872	1518	0	6	554	780
25	5	277	449	3008	8157	0	1	456	1462	0	0	114	251	0	0	49	71
50	1	9082	9082	9975	9979	4651	4651	9237	9253	1699	1699	7547	7555	697	697	6031	6032
50	2	6985	7059	9599	9908	1098	1189	6078	7686	133	160	3353	4270	23	40	2061	2543
50	5	1440	1663	5452	9394	5	16	868	3030	0	1	178	482	0	0	73	125
50	7	254	388	2596	8648	0	0	186	1226	0	0	19	80	0	0	3	10
100	1	9527	9527	9992	9993	5585	5585	9602	9618	1985	1985	7980	7991	744	744	6414	6416
100	2	8295	8316	9884	9980	1846	1886	7303	8618	195	217	3799	4989	49	56	2413	2940
100	5	3391	3562	7636	9804	20	30	1800	4690	0	0	331	802	0	0	84	159
100	10	252	375	2364	9263	0	0	74	956	0	0	3	15	0	0	0	0
250	1	9791	9791	10000	10000	6832	6832	9814	9827	2493	2493	8564	8579	846	846	6793	6795
250	2	9205	9209	9974	9990	3099	3138	8509	9360	277	286	4914	6073	46	50	2881	3439
250	5	6110	6182	9269	9962	106	123	3344	6764	1	1	599	1323	0	0	136	248
250	15	232	306	2281	9697	0	0	16	703	0	0	0	4	0	0	0	0
250	25	3	4	221	9008	0	0	0	28	0	0	0	0	0	0	0	0
500	1	9882	9882	9999	-	7646	7646	9919	-	2935	2935	8885	-	946	946	7184	-
500	2	9596	9598	9993	-	4267	4280	9117	-	401	406	5722	-	33	34	3166	-
500	5	7774	7801	9754	-	273	285	4973	-	1	1	990	-	0	0	226	-
500	22	150	184	1757	-	0	0	3	-	0	0	0	-	0	0	2	-
500	50	0	0	4	-	0	0	0	-	0	0	0	-	0	0	0	-
1000	1	9936	9936	10000	-	8394	8394	9970	-	3181	3181	9137	-	1061	1061	7422	-
1000	2	9789	9790	9999	-	5498	5507	9568	-	525	526	6361	-	51	51	3546	-
1000	5	8797	8803	9947	-	666	676	6482	-	2	2	1471	-	0	0	245	-
1000	32	94	107	1511	-	0	0	1	-	0	0	0	-	0	0	0	-
1000	100	0	0	0	-	0	0	0	-	0	0	0	-	0	0	0	-
2000	1	9975	9975	10000	-	8914	8914	9988	-	3685	3685	9361	-	1099	1099	7613	-
2000	2	9879	9879	9999	-	6774	6777	9791	-	714	714	7048	-	57	57	3858	-
2000	5	9383	9384	9980	-	1519	1535	7906	-	0	0	2159	-	0	0	342	-
2000	45	81	90	1399	-	0	0	0	-	0	0	0	-	0	0	0	-
2000	200	0	0	0	-	0	0	0	-	0	0	0	-	0	0	0	-

Table 5.2: Numbers of instances for $P(\textit{unobserved}) = 0.75$ that are identifiable by use of BC, CBC, CBC$^+$ and IDC (as defined in Section 5.2.2). Gray cells highlight where the CBC was able to identify at least 400 more graphs than the BC. Due to its high time complexity, we were unable to run the IDC algorithm on instances labeled with "-".

Identification by plain formula and identification by adjustment are overlapping concepts. Some cases can be identified using either approach, while in other instances only one of them works. Many cases for which adjustment does not work can be solved instead by the plain formula, meaning that in those DAGs there is in fact no causal effect of $\mathbf{X}$ on $\mathbf{Y}$. This can be seen especially in dense graphs, e.g., DAGs in which each node has $l = 20$ neighbors on average, and for singleton $\mathbf{X}$ and $\mathbf{Y}$, i.e. $k = 1$. For example, for $l = 20$, $k = 1$, $P(\textit{unobserved}) = 0.75$, in DAGs with $n = 2000$ ($n = 1000$, $n = 500$) nodes, up to 65 % (63 %, 61 %) of all instances are identifiable by the plain formula but not by adjustment. Furthermore, increasing n from 25 to 2000, we observe that this percentage ranges between 51% and the maximum 65%, a rather narrow range. The counts for CBC and CBC$^+$ are illustrated in Figure 5.3 as gray squares in the columns labeled as $(l, k) = (20, 1)$ (case: $P(\textit{unobserved}) = 0.75$).

The difficulty of identification by adjustment grows with increasing k and l, but it decreases with an increasing number of nodes n, both for $P(\textit{unobserved}) = 0$ and for 0.75. In Figure 5.3, columns are sorted increasingly by the total number of identifiable effects per column. This shows that the most difficult case is $(l, k) = (20, 5)$: for $P(\textit{unobserved}) = 0$, the counts grows very slowly with n reaching the maximum value of 3% of identifiable graphs for $n = 2000$; for $P(\textit{unobserved}) = 0.75$, almost no instances are identifiable by adjustment (compare the upper panels in Figure 5.3). However, as we can see in Table 5.2, for $n = 250$, only 2.5% of cases are identifiable at all. Figures 5.4 and 5.5 summarize the difficulty of identification stratified by n (Figure 5.4) and l (Figure 5.5), respectively.

Comparison of CBC to the back-door criterion by Pearl. We were also interested in how often Pearl's back-door criterion (BC) would fail to find an adjustment set. Tables 5.1 and 5.2 show that the difference between BC and the CBC is rather small, especially for simple (or hard) instances where nearly every (or no) DAG has an adjustment set, and as expected given our results in Section 4.9, for singletons $\mathbf{X} = \{X\}, \mathbf{Y} = \{Y\}$, the counts for BC and CBC are indeed equal. However, for larger $\mathbf{X}, \mathbf{Y}$ and parameters where only a few graphs have an adjustment set, the difference between BC and CBC becomes more substantial. The greatest difference occurs for $n = 10$, $|\mathbf{X}| = |\mathbf{Y}| = 3$, $m \cong n$, and $P(\textit{unobserved}) = 0$, where in 10% of all cases there is an adjustment set whereas BC finds none. This is followed by $n = 10$, $|\mathbf{X}| = |\mathbf{Y}| = 2$, where BC fails to find existing adjustment sets in 7% to 9% of the cases, depending on $P(\textit{edge})$ and $P(\textit{unobserved})$.

Complete identification by do-calculus compared to identification by adjustment or plain formula. As explained above, in small graphs we have checked for general identifiability of causal effects using the IDC algorithm [SP06a]. Results are shown for $P(\textit{unobserved}) = 0.75$ and $n \leq 250$ in Table 5.2. Since the IDC algorithm is complete for the identification problem, the corresponding counts also show how many instances are identifiable at all. It is known that in the case $P(\textit{unobserved}) = 0$ the causal effect is always identifiable, so we skip the counts for IDC in Table 5.1.

The cases with $n = 10$, $k = |\mathbf{X}| = |\mathbf{Y}| = 5$ (Table 5.2) might seem suspicious as the number of identifiable graphs (i.e., counts in column IDC) increases drastically compared to the graphs with smaller $\mathbf{X}, \mathbf{Y}$, while in all the other cases (see Table 5.2) this number decreases with increasing cardinality of $\mathbf{X}, \mathbf{Y}$. However, this is explained by the cap on the number of unobserved nodes. When $|\mathbf{X}| + |\mathbf{Y}| = 10$ for $n = 10$, there are no nodes outside of $\mathbf{X} \cup \mathbf{Y}$ remaining that could become unobserved regardless of $P(\textit{unobserved})$, similarly to the cases in

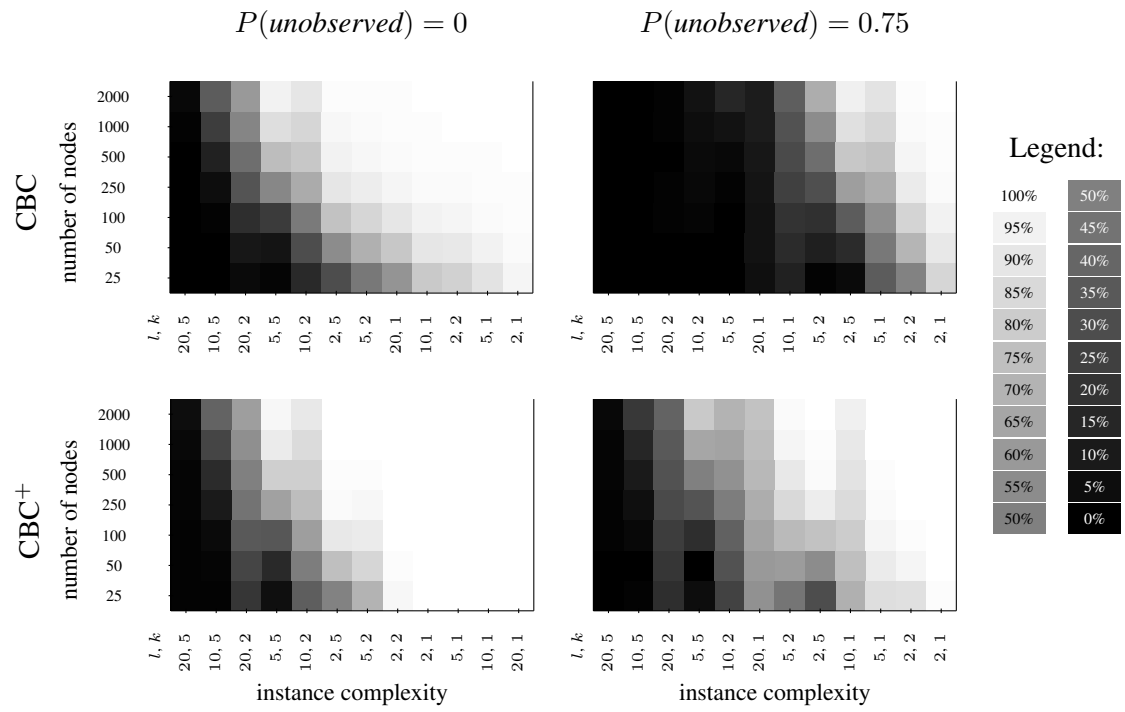


Figure 5.3: Heatmaps visualizing the number of instances that are identifiable by our constructive back-door criterion (CBC, top row), and by the use of CBC or the plain formula (CBC$^+$, bottom row). Black squares depict the worst case in which 0% of instances are identifiable by use of CBC or CBC$^+$, respectively. White squares mean that 100% of instances are identifiable. The instance complexities l, k (where the expected number of neighbors of a node equals l and $|\mathbf{X}| = |\mathbf{Y}| = k$) are sorted by total amount of identifiable instances.

Table 5.1, and all graphs must be identifiable as shown in Section 5.1.3.

Figures 5.4 and 5.5 present the data for CBC$^+$ (the same data as in the lower right panel of Figure 5.3) in comparison to identification by IDC. As we observed in Figure 5.3, the most difficult case for CBC$^+$ is $(l, k) = (20, 5)$ and the difficulty decreases with k and l when n is fixed (Figure 5.4). The situation is very similar for IDC. In Figure 5.5, we see that identifiability for both CBC$^+$ and IDC grows roughly in parallel. Similar to the results for adjustment sets, one can see that with increasing $|\mathbf{X}|, |\mathbf{Y}|, l$ the number of identifiable graphs decreases when $P(\textit{unobserved}) > 0$.

These experiments also provide a verification of our DAGitty implementation as every adjustment set found by the `causaleffect` package has been found by DAGitty, as well as a ground truth of the unidentifiable graphs since a causal effect not identified by the IDC algorithm cannot be identified by any method. Moreover, as expected, for DAGs with no unobserved variables and with $|\mathbf{X}| = |\mathbf{Y}| = 1$, all causal effects are already identified by a plain formula or adjustment without involving the IDC algorithm (see Table 5.1).

Comparative runtimes of the algorithms. Figure 5.6 (black lines) shows the time needed by DAGitty for these experiments on one core of a 2.1 GHz (up 3 GHz with Turbo Core) AMD Opteron 6272 for graphs with $P(\textit{unobserved}) = 0.75$. Graphs with a lower probability

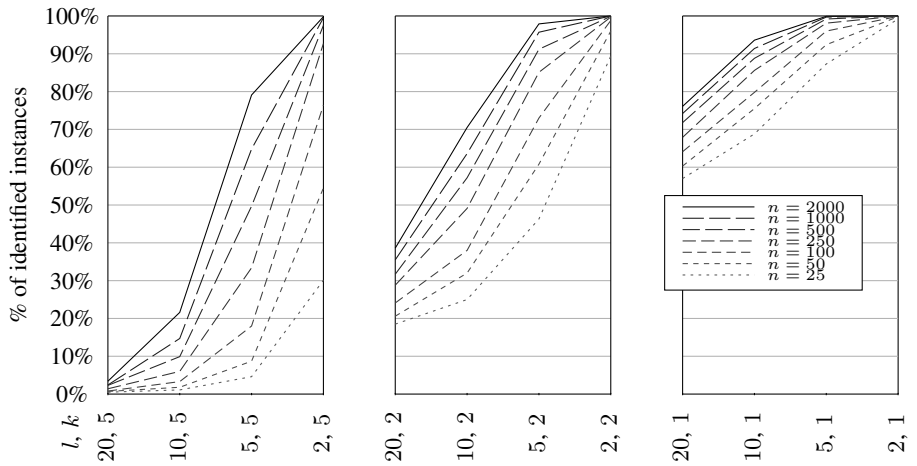


Figure 5.4: Case $P(\textit{unobserved} = 0.75)$: Percent of identifiable graphs for fixed numbers of nodes $n \in \{25, 50, 100, 250, 500, 1000, 2000\}$ and with varying expected number of node neighbors l and cardinalities $|\mathbf{X}| = |\mathbf{Y}| = k$. The horizontal axis is labeled by (l, k) sorted lexicographically. The curves show the data for CBC$^+$, i.e., for instances identifiable by adjustment or by plain formula.

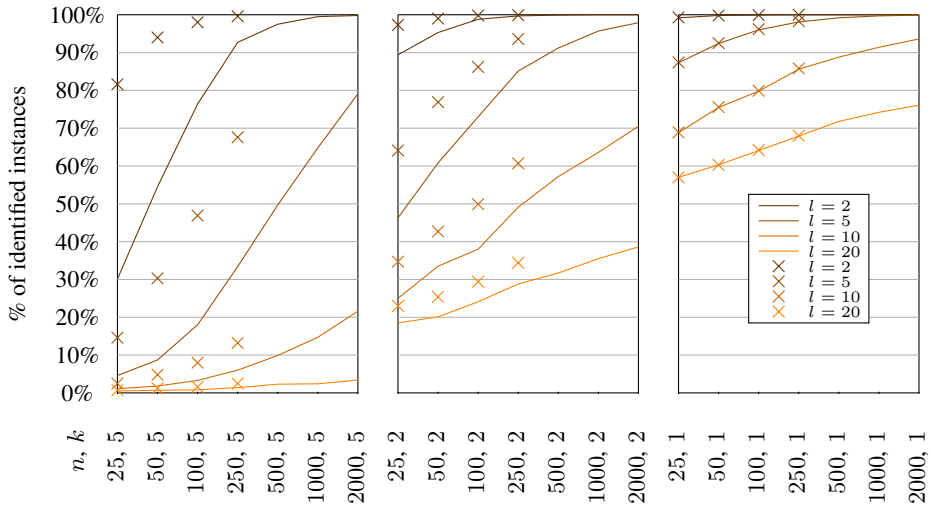


Figure 5.5: Case $P(\textit{unobserved} = 0.75)$: Percent of identifiable graphs for fixed density parameter values $l \in \{2, 5, 10, 20\}$ and with varying the number of nodes n and cardinalities $|\mathbf{X}| = |\mathbf{Y}| = k$. The horizontal axis is labeled by (n, k) sorted lexicographically. The curves show the data for CBC$^+$, i.e. for instances identifiable by adjustment or by a plain formula; Crosses show data for IDC, i.e., they show how many cases are identifiable at all. The high time complexity of the IDC algorithm precluded computations for graphs of sizes $n \geq 500$.

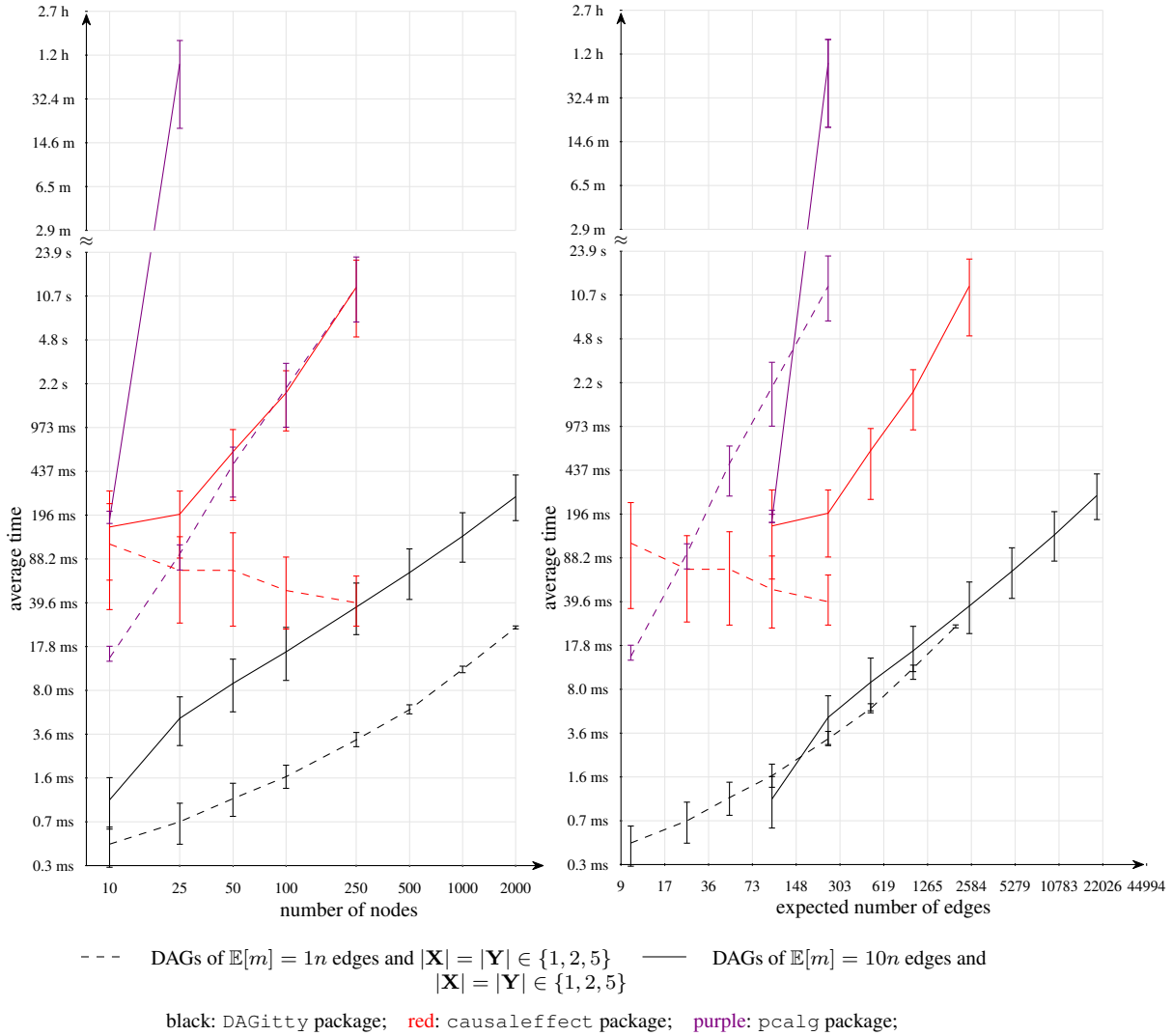


Figure 5.6: Average time needed to find an adjustment set and verify it according to the CBC in a single graph with $P(\textit{unobserved}) = 0.75$, values $n \in \{10, 25, 50, 100, 250, 500, 1000, 2000\}$, $\mathbb{E}[m] \in \{1n, 10n\}$, and various cardinalities k of $\mathbf{X}, \mathbf{Y}$. The left and the right plot show the same data, but with a different metric on the horizontal axis: the number of nodes n (left) and the expected number of edges m (right). In black, we show the statistics for `DAGitty`, in red– the data for the R package `causaleffect` and in purple– the data for the `gac` function of the R package `pcalg` R. Error bars show the minimum and maximum time taken. The plot shows that all small graphs ($n \leq 100$) are solved nearly instantaneously (time $\leq 100ms$) by `DAGitty`. Only graphs with a high number of edges and huge $\mathbf{X}, \mathbf{Y}$ can require a few seconds. Thus `DAGitty` is one to two magnitudes faster than the `causaleffect` package or the `pcalg` package.

		$l = 2$			$l = 5$			$l = 10$			$l = 20$		
n	k	**CBC**	**IDC**	**GAC**	**CBC**	**IDC**	**GAC**	**CBC**	**IDC**	**GAC**	**CBC**	**IDC**	**GAC**
10	1	0.3 ms	35.0 ms	13.6 ms	0.5 ms	53.6 ms	49.6 ms	0.6 ms	60.4 ms	303 ms	0.6 ms	59.9 ms	169 ms
10	2	0.5 ms	70.5 ms	17.9 ms	0.6 ms	100 ms	61.9 ms	0.8 ms	110 ms	173 ms	1.0 ms	110 ms	211 ms
10	3	0.5 ms	124 ms	14.1 ms	0.8 ms	182 ms	32.5 ms	1.1 ms	190 ms	121 ms	1.1 ms	187 ms	60.7 ms
10	5	0.7 ms	242 ms	15.3 ms	1.0 ms	296 ms	52.4 ms	1.6 ms	306 ms	170 ms	1.6 ms	305 ms	173 ms
25	1	0.5 ms	27.4 ms	72.1 ms	0.9 ms	46.9 ms	1.9 s	1.6 ms	78.7 ms	18.5 s	2.9 ms	89.7 ms	1.6 h
25	2	0.7 ms	54.9 ms	112 ms	1.3 ms	121 ms	680 ms	2.2 ms	193 ms	16.4 s	4.4 ms	206 ms	-
25	3	0.8 ms	70.3 ms	113 ms	1.5 ms	127 ms	695 ms	2.6 ms	168 ms	15.1 s	5.7 ms	168 ms	39.9 min
25	5	1.0 ms	132 ms	114 ms	1.7 ms	276 ms	522 ms	3.1 ms	309 ms	9.5 s	7.1 ms	305 ms	18.9 min
50	1	0.8 ms	25.9 ms	273 ms	1.6 ms	51.7 ms	-	2.8 ms	151 ms	-	5.4 ms	257 ms	-
50	2	1.1 ms	46.9 ms	557 ms	2.0 ms	157 ms	-	3.8 ms	475 ms	-	7.7 ms	691 ms	-
50	5	1.5 ms	142 ms	680 ms	2.8 ms	414 ms	-	5.6 ms	796 ms	-	14.2 ms	937 ms	-
50	7	1.7 ms	224 ms	744 ms	3.3 ms	649 ms	-	6.7 ms	1.1 s	-	17.9 ms	1.1 s	-
100	1	1.3 ms	24.5 ms	978 ms	2.5 ms	53.1 ms	-	4.8 ms	366 ms	-	9.6 ms	914 ms	-
100	2	1.6 ms	32.4 ms	1.7 s	3.0 ms	128 ms	-	6.1 ms	836 ms	-	13.6 ms	1.8 s	-
100	5	2.0 ms	91.3 ms	3.1 s	4.1 ms	440 ms	-	9.2 ms	1.7 s	-	25.3 ms	2.7 s	-
100	10	2.6 ms	248 ms	4.1 s	5.6 ms	843 ms	-	13.8 ms	2.4 s	-	47.5 ms	3.0 s	-
250	1	2.9 ms	25.9 ms	6.7 s	5.1 ms	74.1 ms	-	10.2 ms	966 ms	-	22.2 ms	5.1 s	-
250	2	3.1 ms	28.2 ms	9.2 s	6.0 ms	130 ms	-	12.9 ms	2.6 s	-	30.6 ms	12.0 s	-
250	5	3.7 ms	64.6 ms	21.9 s	7.7 ms	499 ms	-	19.1 ms	6.2 s	-	56.8 ms	20.7 s	-
250	15	5.3 ms	462 ms	48.5 s	13.3 ms	2.8 s	-	38.9 ms	19.3 s	-	144 ms	44.1 s	-
250	25	6.4 ms	644 ms	1.1 min	17.4 ms	3.2 s	-	57.0 ms	12.9 s	-	252 ms	22.9 s	-
500	1	5.2 ms	-	-	8.7 ms	-	-	18.1 ms	-	-	42.1 ms	-	-
500	2	5.5 ms	-	-	10.1 ms	-	-	22.7 ms	-	-	58.1 ms	-	-
500	5	6.2 ms	-	-	12.3 ms	-	-	33.8 ms	-	-	106 ms	-	-
500	22	9.0 ms	-	-	24.5 ms	-	-	89.5 ms	-	-	391 ms	-	-
500	50	13.4 ms	-	-	46.3 ms	-	-	192 ms	-	-	886 ms	-	-
1000	1	11.1 ms	-	-	17.2 ms	-	-	35.4 ms	-	-	83.2 ms	-	-
1000	2	11.8 ms	-	-	19.0 ms	-	-	42.6 ms	-	-	114 ms	-	-
1000	5	12.4 ms	-	-	21.8 ms	-	-	61.8 ms	-	-	205 ms	-	-
1000	32	17.1 ms	-	-	51.3 ms	-	-	225 ms	-	-	1.1 s	-	-
1000	100	33.5 ms	-	-	143 ms	-	-	715 ms	-	-	3.4 s	-	-
2000	1	24.6 ms	-	-	35.1 ms	-	-	70.8 ms	-	-	178 ms	-	-
2000	2	25.1 ms	-	-	37.7 ms	-	-	83.6 ms	-	-	241 ms	-	-
2000	5	26.0 ms	-	-	41.8 ms	-	-	120 ms	-	-	409 ms	-	-
2000	45	34.5 ms	-	-	107 ms	-	-	549 ms	-	-	2.8 s	-	-
2000	200	92.3 ms	-	-	505 ms	-	-	2.7 s	-	-	13.4 s	-	-

Table 5.3: Average time to run DAGitty (CBC), the R package causaleffect (IDC) or the `gac` function of the R package `pcalg` (GAC) on one graph with $P(\textit{unobserved} = 0.75)$. Omitted values indicate experiments we did not run or were unable to run due to runtime issues. On some parametrizations we only ran the `gac` function on the first 100 graphs rather than the full set of 10000 graphs due to time constraints.

$P(\textit{unobserved})$ are processed slightly faster. For small sets $\mathbf{X}$ and $\mathbf{Y}$, the time increases roughly linearly with the number of edges m. For larger sets, the time also increases with the size of $\mathbf{X}, \mathbf{Y}$, which could either mean that DAGitty does not reach the optimal asymptotic runtime of $\mathcal{O}(n+m)$ due to inefficient set operations, or that the time actually only depends on $\mathcal{O}(\textit{An}(\mathbf{X}, \mathbf{Y}))$, which can be much smaller than $\mathcal{O}(m)$ when the sets and degrees are small. However, for all models of a size currently used in practice, DAGitty finds the adjustment set nearly instantaneously.

The runtimes of the `causaleffect` package are shown as a red plot in Figure 5.6. Since the IDC algorithm is far more complex than the expression of Theorem 4.44, it performs generally one to two orders of magnitude slower than the implementation in DAGitty, or equivalently, in the same time, DAGitty can process graphs that are one to two orders of magnitude larger. Due to this speed difference, it was not possible for us to run the IDC algorithm experiments on the larger graphs.

We have also investigated a different implementation of the CBC in the R package pcalg [Kal+12]. The `gac` function in that package implements the CBC criterion for DAGs and other graph classes. Unlike DAGitty, the pcalg package does not find an adjustment set, but only verifies whether a given set meets the criterion. Hence, after loading the graphs in R and calculating the adjacency matrices required by pcalg, we compute the canonical adjustment set $\textit{Adjustment}(\mathbf{X}, \mathbf{Y})$ in R as

```
Dpcp = De(G, intersect(setminus(De(GNoInX, x), x), An(GNoOutX, y)))
z = setminus(An(G, union(x,y)), union(union(x,y),union(Dpcp, obs)))
```

with sets `x`, `y`, observed nodes `obs`, graphs `G` $= \mathcal{G}$, `GNoInX` $= \mathcal{G}_{\overline{X}}$, `GNoOutX` $= \mathcal{G}_{\underline{X}}$ and helper functions `An` and `De` implemented using the `subcomponent` function of the R package `igraph`. We then compare the time required by `pcalg` to test whether `z` is a valid adjustment set to the time required by DAGitty to find and test a set. The runtimes of the `gac` function are plotted in purple in Figure 5.6. They show that the `gac` function is several orders of magnitude slower than DAGitty. These results are expected given that the `pcalg` package tests the CBC by tracing all m-connected paths using backtracking, an approach that suffers from exponential worst-case complexity; in fact, this backtracking algorithm is even slower than the general implementation of the do-calculus in the `causaleffect` package. Only the cases with small n are shown as the remaining computations did not terminate in a reasonable time.

5.3 Empirical Analysis of Identifiability by Adjustment in MAGs

We test for each DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, generated as described in Section 5.2, whether the causal effect in the MAGs $\mathcal{G}[^{\emptyset}_{\emptyset}$ and $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ can be identified via adjustment. Hereby, $\mathbf{L} = \mathbf{V} \setminus \mathbf{R}$ is the set of unobserved nodes, which is used to determine the MAG $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$. Thus, all nodes of $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ are considered as observed and all of them are allowed to occur in adjustment sets. The MAG $\mathcal{G}[^{\emptyset}_{\emptyset}$ is syntactically equal to $\mathcal{G}$ and $\mathbf{L}$ specifies forbidden nodes for adjustments. The MAGs $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ can be constructed directly according to the definition of MAGs by testing all pairs of nodes in the DAG to determine which pair of nodes can be d-separated. However, we have implemented the DAG-to-MAG conversion algorithm from [Zha08] in DAGitty, which is based on Theorem 4.2 in [RS02] that two nodes A, B are adjacent in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}$ if and only if there exists an inducing path between A and B with respect to $\emptyset, \mathbf{L}$ (see Definition 4.14), because testing if an inducing path exists appears easier than testing if two nodes are d-separable.

Since $\mathcal{G}$ and $\mathcal{G}[^{\emptyset}_{\emptyset}$ are syntactically equal, the only difference between using the CBC in a

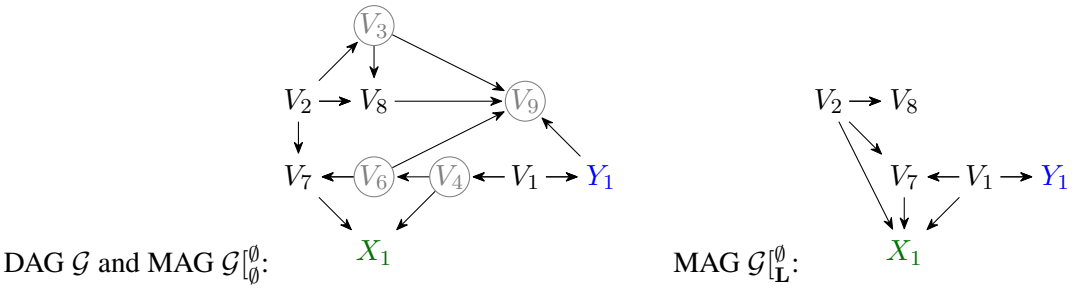


Figure 5.7: Example of a DAG and resulting MAG sampled for the parameter tuple $n = 10, P(\textit{edge}) = 2/9, P(\textit{unobserved}) = 0.75$, and $k = |\mathbf{X}| = |\mathbf{Y}| = 1$. Nodes are relabeled such that exposures are called X_1 and outcomes are called Y_1. The nodes $\mathbf{L} = \{V_3, V_4, V_6, V_9\}$ are unobserved. $\{V_1\}$ is an adjustment set in $\mathcal{G}, \mathcal{G}[^{\emptyset}_{\emptyset}]$ and $\mathcal{G}[^{\emptyset}_{\mathbf{L}}]$.

DAG $\mathcal{G}$ and the CBC in $\mathcal{G}$ interpreted as a MAG is that the last case needs a test for adjustment amenability of the graph. Figure 5.7 shows an example.

The results of our experiments are shown in Table 5.4 and 5.5. As expected we find fewer adjustment sets in the MAGs than in the DAGs since any adjustment set found in the MAG is valid for all represented DAGs. There are also always fewer adjustment sets in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}]$ than in $\mathcal{G}[^{\emptyset}_{\emptyset}]$, because with fewer nodes and edges in $\mathcal{G}[^{\emptyset}_{\mathbf{L}}]$, there are fewer visible edges and $\mathcal{G}[^{\emptyset}_{\mathbf{L}}]$ might not be adjustment amenable even if $\mathcal{G}[^{\emptyset}_{\emptyset}]$ is. For example, in the DAG $\mathcal{G} = L \to X \to Y$ and the MAG $\mathcal{G}[^{\emptyset}_{\emptyset}] = L \to X \to Y$, the empty set is an adjustment set. However, in the MAG $\mathcal{G}[^{\emptyset}_{L}] = X \to Y$, there exists no valid adjustment set.

For $n = 10$, $l = 10$ (20), we have $P(\textit{edge}) = \max\{10/9, 1\} = 1$, so the experiments generate only *complete* DAGs, such that all nodes are adjacent in the DAG and corresponding MAGs. This implies that there is either an edge $X \leftarrow Y$, in which case adjustment is impossible in both the DAG and the MAG, or there is an edge $X \to Y$, which then would need to be visible for adjustment in the MAG due to Lemma 4.12. However, there are no visible edges in complete graphs since a visible edge $X \to Y$ would require a node not adjacent to Y. Thus, no adjustment sets are found for these parameters.

The runtimes of these experiments are listed in Table 5.6. The time required to find the adjustment set in the MAG is similar to finding it in the original DAG and quick enough to be negligible. We did not run the experiments on larger graphs since constructing a MAG from a DAG has quadratic runtime complexity in the worst case (as a quadratic number of edges needs to be added), which makes constructing the MAGs too slow even with the supposedly faster DAG-to-MAG algorithm.

5.4 Empirical Analysis of Identifiability by Adjustment in RCGs

In this section, we empirically evaluate our adjustment criterion for RCGs. In the previous section, we have converted each DAG $\mathcal{G}$ generated in Section 5.2 to a MAG, so for this evaluation, we could convert each DAG to an RCG. However, since every DAG is already an RCG, the results would be exactly the same as those of Section 5.2. So rather than converting $\mathcal{G}$ to an RCG, we convert it to a CPDAG and apply the adjustment criterion to the CPDAG. Effectively, this means for each DAG $\mathcal{G}$ we search for an adjustment set that is valid for all Markov equivalent DAGs $\mathcal{G}' \in [\mathcal{G}]$. The conversion of a DAG to a CPDAG is implemented in DAGitty by converting

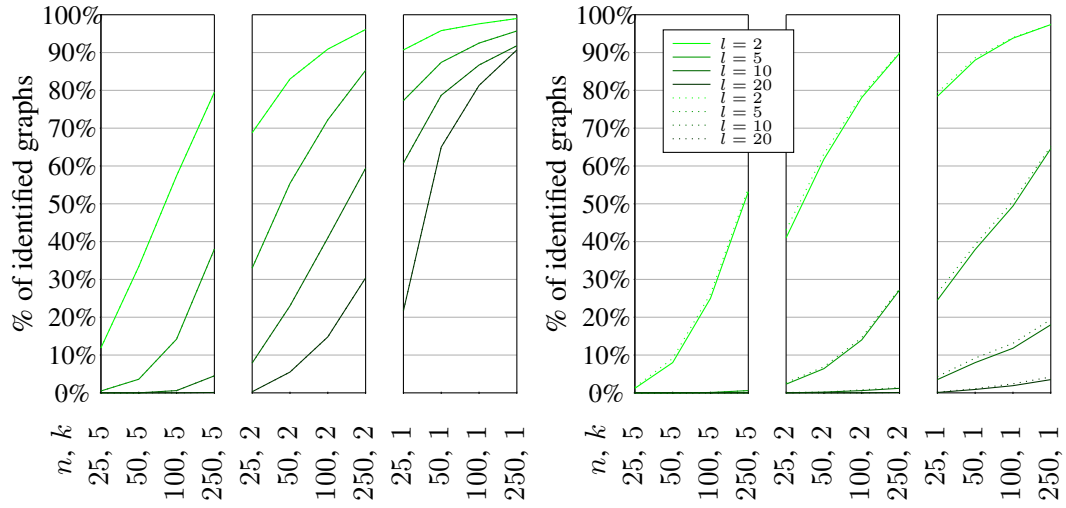


Figure 5.8: Percent of identifiable MAGs for $P(\textit{unobserved}) = 0$ (left) and $P(\textit{unobserved}) = 0.75$ (right). The horizontal axis is labeled by (n, k) sorted lexicographically by the number of nodes n and cardinalities $|\mathbf{X}| = |\mathbf{Y}| = k$. The dotted lines show the results for $\mathcal{G}[^{\emptyset}_{\emptyset}]$ and the solid lines for $\mathcal{G}[^{\emptyset}_{\mathbf{L}}]$ with various density parameter values $l \in \{2, 5, 10, 20\}$.

		$l = 2$		$l = 5$		$l = 10$		$l = 20$	
n	k	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$
10	1	8893	7711	7205	4600	5034	0	4934	0
10	2	6061	3773	1980	529	660	0	686	0
10	3	3395	1243	548	32	168	0	174	0
10	5	886	64	108	0	36	0	31	0
25	1	9573	9066	8936	7731	7905	6080	5843	2168
25	2	8247	6876	4735	3300	1553	789	401	33
25	3	6424	4399	1852	913	212	61	34	0
25	5	3021	1186	243	51	11	0	1	0
50	1	9832	9582	9476	8742	8997	7869	7832	6501
50	2	9128	8305	6938	5538	3049	2301	866	549
50	5	5535	3339	799	361	16	5	2	0
50	7	3120	1234	119	21	1	0	0	0
100	1	9907	9756	9783	9249	9494	8674	8966	8126
100	2	9593	9092	8353	7216	4834	4098	1762	1476
100	5	7591	5742	2336	1416	102	59	4	1
100	10	3040	1117	48	9	0	0	0	0
250	1	9947	9898	9894	9573	9774	9176	9621	9069
250	2	9840	9613	9327	8534	6689	5942	3285	3035
250	5	8994	7954	5325	3802	544	453	17	12
250	15	3537	1360	32	6	0	0	0	0
250	25	536	50	0	0	0	0	0	0

Table 5.4: Number of instances for $P(\textit{unobserved}) = 0$ that are identified using CBC in the DAG or after converting the DAG to a MAG.

		$l=2$			$l=5$			$l=10$			$l=20$		
n	k	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$	**MAG**$^{\emptyset}_{\mathbf{L}}$	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$	**MAG**$^{\emptyset}_{\mathbf{L}}$	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$	**MAG**$^{\emptyset}_{\mathbf{L}}$	**CBC**	**MAG**$^{\emptyset}_{\emptyset}$	**MAG**$^{\emptyset}_{\mathbf{L}}$
10	1	6333	5398	5157	1935	645	543	978	0	0	936	0	0
10	2	2339	1371	1204	228	8	4	113	0	0	114	0	0
10	3	980	391	304	21	0	0	9	0	0	10	0	0
10	5	859	56	56	98	0	0	43	0	0	26	0	0
25	1	8414	7945	7838	3647	2635	2446	1340	429	346	557	18	13
25	2	5331	4309	4098	728	268	234	130	6	3	41	0	0
25	3	2632	1647	1491	144	16	14	17	0	0	6	0	0
25	5	449	149	119	1	0	0	0	0	0	0	0	0
50	1	9082	8856	8805	4651	3925	3788	1699	923	798	697	112	93
50	2	7059	6320	6177	1189	689	643	160	22	16	41	1	0
50	5	1663	906	805	16	0	0	1	0	0	0	0	0
50	7	388	130	102	0	0	0	0	0	0	0	0	0
100	1	9527	9395	9375	5585	5066	4943	1985	1311	1176	744	243	194
100	2	8316	7880	7808	1886	1462	1398	217	77	64	56	3	3
100	5	3562	2620	2501	30	11	10	0	0	0	0	0	0
100	10	375	109	84	0	0	0	0	0	0	0	0	0
250	1	9791	9743	9735	6832	6517	6462	2493	1944	1802	846	424	354
250	2	9209	9005	8980	3138	2760	2715	286	136	123	50	6	6
250	5	6182	5424	5321	123	63	57	1	0	0	0	0	0
250	15	306	98	82	0	0	0	0	0	0	0	0	0
250	25	4	1	0	0	0	0	0	0	0	0	0	0

Table 5.5: Number of instances for $P(\textit{unobserved}) = 0.75$ that are identified using CBC in the DAG or after converting the DAG to a MAG, either a MAG$^{\emptyset}_{\mathbf{L}}$ with latent nodes being removed from the graph or a MAG$^{\emptyset}_{\emptyset}$ with latent nodes left and marked as latent.

each directed edge that is not strongly protected to an undirected edge.

Tables 5.7 and 5.8 list our results, and Figure 5.9 plots them. Table 5.9 gives the average runtimes.

Most instances that are identifiable in DAGs are also identifiable in CPDAGs.

For $p(\textit{unobserved}) = 0.75$, there are always strictly more instances identifiable by adjustment in CPDAGs than in MAGs, except in cases where no instance is identifiable in both graphical models and the case of $n = 250, k = 15, l = 2$ where the MAG$^{\emptyset}_{\emptyset}$ can identify one additional instance. For $p(\textit{unobserved}) = 0$, the numbers are more varied. With increasing k and l, there are also more instances identifiable in CPDAGs than in MAGs, but with increasing n, more instances in MAGs become identifiable than in CPDAGs.

Surprisingly, the conversion of a DAG to a CPDAG is more than a hundred times faster than the conversion to a MAG. The time to search for an adjustment is approximately the same.

		$P(unobserved) = 0$				$P(unobserved) = 0.75$							
		$l = 2$	$l = 5$	$l = 10$	$l = 20$	$l = 2$		$l = 5$		$l = 10$		$l = 20$	
n	k	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\mathbf{L}}$	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\mathbf{L}}$	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\mathbf{L}}$	$\mathbf{MAG}^{\emptyset}_{\emptyset}$	$\mathbf{MAG}^{\emptyset}_{\mathbf{L}}$
10	1	0.5, 0.3	1.1, 0.5	1.6, 0.8	1.7, 0.8	0.2, 0.2	0.5, 0.4	0.2, 0.2	1.1, 0.6	0.3, 0.3	1.7, 0.9	0.3, 0.3	1.7, 0.9
10	2	0.7, 0.5	1.5, 0.9	2.2, 1.8	2.4, 1.8	0.2, 0.3	0.5, 0.5	0.4, 0.5	1.1, 0.8	0.4, 0.5	1.7, 1.6	0.4, 0.5	1.7, 1.6
10	3	0.8, 0.6	1.3, 1.1	2.1, 2.5	2.4, 2.7	0.5, 0.5	0.7, 0.6	0.9, 1.0	1.4, 1.0	0.9, 1.1	2.1, 2.4	1.1, 1.2	2.4, 2.5
10	5	0.8, 0.9	1.6, 1.7	2.3, 4.7	2.4, 4.7	0.7, 0.9	0.7, 0.8	1.5, 1.7	1.4, 1.6	2.4, 4.9	2.4, 4.9	2.4, 4.7	2.4, 4.7
25	1	3.4, 0.5	10, 1.0	22, 1.8	50, 3.9	0.8, 0.3	3.5, 0.5	2.0, 0.4	9.6, 1.1	3.4, 0.6	24, 2.0	5.2, 0.6	49, 4.1
25	2	3.7, 0.7	10, 1.4	22, 2.6	49, 7.0	0.8, 0.4	3.6, 0.7	2.1, 0.6	9.8, 1.4	3.5, 0.9	24, 2.8	5.4, 1.1	50, 7.4
25	3	3.8, 0.8	10, 1.7	23, 3.3	50, 10	1.0, 0.5	3.7, 0.9	2.4, 0.9	9.6, 1.7	3.9, 1.4	24, 3.5	6.0, 1.6	50, 11
25	5	3.7, 1.1	10, 2.2	22, 4.7	49, 17	1.7, 0.8	3.7, 1.1	4.5, 2.1	9.6, 2.2	6.8, 3.9	24, 4.5	10, 4.9	49, 15
50	1	14, 0.8	38, 1.6	92, 3.1	271, 6.3	2.9, 0.3	13, 0.9	12, 0.7	36, 1.8	23, 1.2	97, 3.3	38, 1.6	267, 6.8
50	2	13, 0.9	38, 2.1	91, 4.2	265, 9.4	3.0, 0.4	14, 1.1	12, 0.9	36, 2.3	22, 1.9	93, 4.4	37, 2.9	260, 9.8
50	5	14, 1.4	38, 3.2	92, 7.0	271, 19	3.1, 0.7	13, 1.5	12, 2.0	36, 3.3	23, 5.6	98, 7.5	40, 8.4	273, 20
50	7	13, 1.7	35, 3.8	95, 8.6	263, 25	4.0, 0.9	13, 1.8	16, 3.5	36, 4.0	29, 11	98, 9.4	47, 16	274, 26
100	1	51, 1.3	146, 2.7	361, 5.5	1129, 12	12, 0.4	50, 1.5	74, 1.4	138, 3.1	168, 3.1	380, 6.1	299, 5.1	1158, 12
100	2	51, 1.5	147, 3.4	348, 7.2	1120, 17	12, 0.5	49, 1.8	74, 1.7	138, 3.6	163, 5.1	366, 7.7	292, 11	1135, 17
100	5	51, 2.1	147, 5.0	348, 11	1117, 31	11, 0.8	49, 2.3	75, 3.1	138, 5.1	163, 15	367, 12	292, 36	1138, 32
100	10	51, 2.9	147, 7.3	357, 18	1122, 57	12, 1.2	50, 3.1	76, 7.1	138, 7.6	171, 43	383, 20	294, 93	1147, 59
250	1	314, 2.4	894, 4.5	2967, 10	12194, 23	64, 0.8	303, 3.1	663, 5.6	915, 5.6	1829, 18	3044, 12	3341, 40	12460, 25
250	2	313, 2.6	889, 5.4	2941, 14	12327, 33	65, 0.9	308, 3.4	633, 6.1	907, 6.5	1770, 26	2934, 15	3315, 93	12119, 34
250	5	312, 3.1	882, 7.8	2944, 22	12441, 66	65, 1.2	308, 4.0	637, 8.5	905, 8.7	1775, 75	2938, 23	3302, 366	12279, 63
250	15	312, 4.9	889, 15	2932, 49	12049, 169	64, 2.1	303, 5.9	649, 25	935, 16	1835, 412	2975, 49	3357, 1860	12550, 177
250	25	314, 6.8	872, 22	2938, 76	12219, 275	65, 3.1	310, 8.0	640, 54	908, 25	1783, 931	2925, 80	3397, 3645	12767, 292

Table 5.6: Time (in milliseconds) to first construct the MAG from the DAG and then check for the existence of an adjustment set in that MAG, for $P(\textit{unobserved}) = 0$, respectively $P(\textit{unobserved}) = 0.75$.

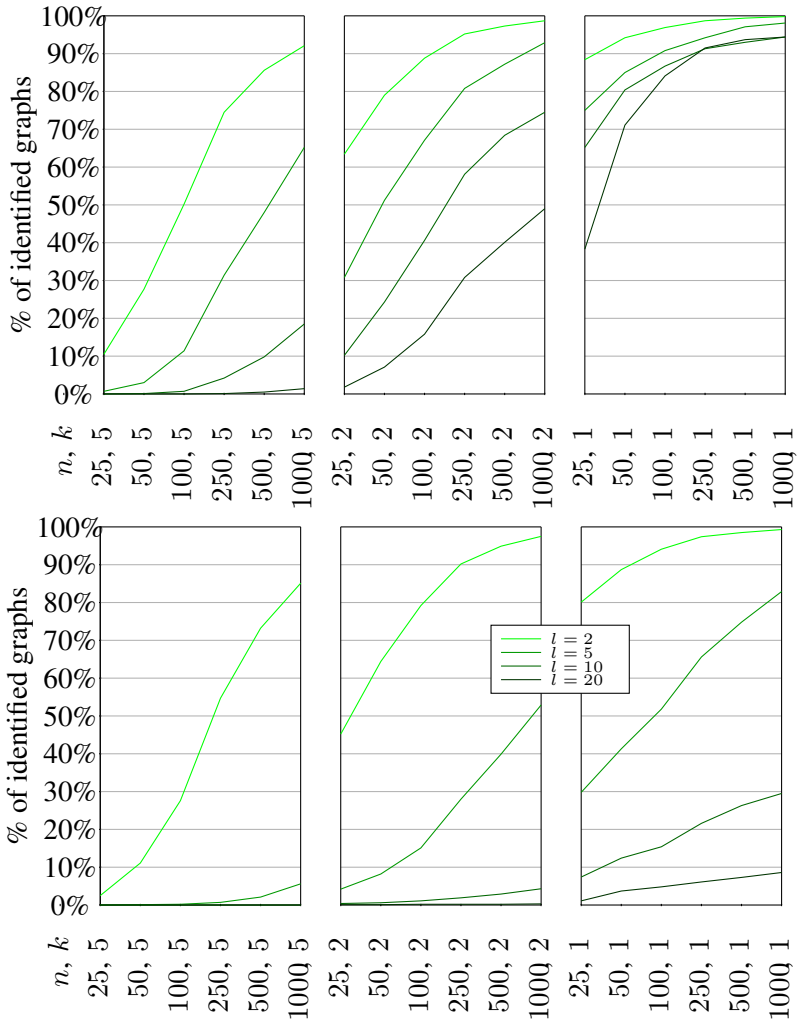


Figure 5.9: Percent of identifiable CPDAGs for $P(\textit{unobserved}) = 0$ (top) and $P(\textit{unobserved}) = 0.75$ (bottom). The horizontal axis is labeled by (n, k) sorted lexicographically by the number of nodes n and cardinalities $|\mathbf{X}| = |\mathbf{Y}| = k$.

		$l = 2$		$l = 5$		$l = 10$		$l = 20$	
n	k	**CBC**	**CPDAG**	**CBC**	**CPDAG**	**CBC**	**CPDAG**	**CBC**	**CPDAG**
10	1	8893	7145	7205	4428	5034	0	4934	0
10	2	6061	3454	1980	718	660	0	686	0
10	3	3395	1378	548	158	168	0	174	0
10	5	886	251	108	54	36	0	31	0
25	1	9573	8843	8936	7496	7905	6522	5843	3833
25	2	8247	6340	4735	3081	1553	1019	401	178
25	3	6424	3898	1852	895	212	101	34	16
25	5	3021	1050	243	67	11	3	1	1
50	1	9832	9421	9476	8497	8997	8043	7832	7106
50	2	9128	7896	6938	5116	3049	2430	866	710
50	5	5535	2774	799	295	16	7	2	1
50	7	3120	996	119	28	1	0	0	0
100	1	9907	9691	9783	9076	9494	8673	8966	8406
100	2	9593	8881	8353	6713	4834	4061	1762	1577
100	5	7591	5023	2336	1142	102	67	4	1
100	10	3040	832	48	13	0	0	0	0
250	1	9947	9869	9894	9421	9774	9132	9621	9152
250	2	9840	9522	9327	8085	6689	5810	3285	3084
250	5	8994	7447	5325	3138	544	415	17	13
250	15	3537	873	32	2	0	0	0	0
250	25	536	22	0	0	0	0	0	0
500	1	9977	9937	9946	9709	9883	9297	9799	9369
500	2	9923	9731	9684	8715	7750	6844	4266	4006
500	5	9490	8555	7368	4800	1265	980	48	48
500	22	3288	679	14	0	0	0	0	0
500	50	29	0	0	0	0	0	0	0
1000	1	9990	9976	9973	9813	9942	9441	9885	9438
1000	2	9966	9871	9845	9293	8434	7447	5173	4901
1000	5	9736	9214	8726	6524	2396	1852	149	142
1000	32	3191	527	6	0	0	0	0	0
1000	100	0	0	0	0	0	0	0	0

Table 5.7: Number of instances for $P(\textit{unobserved}) = 0$ that are identified using CBC in the DAG or after converting the DAG to a CPDAG.

		$l = 2$		$l = 5$		$l = 10$		$l = 20$	
n	k	**CBC**	**CPDAG**	**CBC**	**CPDAG**	**CBC**	**CPDAG**	**CBC**	**CPDAG**
10	1	6333	5644	1935	1052	978	0	936	0
10	2	2339	1701	228	69	113	0	114	0
10	3	980	583	21	7	9	0	10	0
10	5	859	241	98	46	43	0	26	0
25	1	8414	8009	3647	2979	1340	745	557	112
25	2	5331	4520	728	421	130	45	41	7
25	3	2632	1920	144	58	17	0	6	2
25	5	449	248	1	0	0	0	0	0
50	1	9082	8874	4651	4134	1699	1236	697	372
50	2	7059	6436	1189	823	160	61	41	13
50	5	1663	1113	16	6	1	0	0	0
50	7	388	192	0	0	0	0	0	0
100	1	9527	9408	5585	5185	1985	1545	744	484
100	2	8316	7923	1886	1513	217	113	56	20
100	5	3562	2762	30	17	0	0	0	0
100	10	375	160	0	0	0	0	0	0
250	1	9791	9744	6832	6560	2493	2157	846	608
250	2	9209	9022	3138	2804	286	188	50	20
250	5	6182	5467	123	73	1	0	0	0
250	15	306	132	0	0	0	0	0	0
250	25	4	0	0	0	0	0	0	0
500	1	9882	9851	7646	7476	2935	2628	946	733
500	2	9598	9489	4280	3991	406	289	34	17
500	5	7801	7319	285	209	1	0	0	0
500	22	184	76	0	0	0	0	0	0
500	50	0	0	0	0	0	0	0	0
1000	1	9936	9926	8394	8289	3181	2949	1061	857
1000	2	9790	9752	5507	5292	526	431	51	27
1000	5	8803	8513	676	564	2	1	0	0
1000	32	107	36	0	0	0	0	0	0
1000	100	0	0	0	0	0	0	0	0

Table 5.8: Number of instances for $P(\textit{unobserved}) = 0.75$ that are identified using CBC in the DAG or after converting the DAG to a CPDAG.

		$P(unobserved) = 0$				$P(unobserved) = 0.75$			
n	k	$l = 2$	$l = 5$	$l = 10$	$l = 20$	$l = 2$	$l = 5$	$l = 10$	$l = 20$
10	1	0.1, 0.4	0.3, 0.5	1.3, 0.9	1.3, 0.9	0.1, 0.4	0.3, 0.5	1.3, 1.0	1.3, 0.9
10	2	0.1, 0.5	0.4, 0.8	1.4, 1.3	1.4, 1.3	0.1, 0.5	0.3, 0.6	1.3, 1.0	1.3, 1.0
10	3	0.1, 0.6	0.4, 0.9	1.3, 1.0	1.4, 1.3	0.1, 0.6	0.4, 0.9	1.3, 1.1	1.3, 1.1
10	5	0.2, 0.7	0.4, 1.1	1.4, 1.3	1.4, 1.3	0.1, 0.7	0.4, 1.1	1.5, 1.2	1.4, 1.2
25	1	0.3, 0.5	0.8, 0.9	1.7, 1.5	9.9, 2.7	0.3, 0.6	0.8, 1.0	1.8, 1.6	10, 2.8
25	2	0.3, 0.8	0.8, 1.4	1.9, 2.3	10, 4.4	0.3, 0.7	0.8, 1.3	1.9, 2.1	10, 3.9
25	3	0.3, 0.9	0.8, 1.6	1.9, 2.7	11, 5.7	0.4, 0.9	0.9, 1.6	2.0, 2.7	11, 5.9
25	5	0.3, 1.1	0.8, 1.9	1.9, 3.5	10, 8.0	0.4, 1.2	0.8, 1.9	2.0, 3.3	10, 7.5
50	1	0.6, 0.8	1.6, 1.6	3.4, 2.9	9.6, 5.6	0.7, 0.9	1.6, 1.8	3.5, 3.1	9.6, 5.9
50	2	0.6, 1.0	1.6, 2.1	3.4, 3.9	9.6, 8.0	0.7, 1.1	1.6, 2.2	3.5, 4.0	9.9, 8.2
50	5	0.6, 1.6	1.6, 3.0	3.5, 5.9	9.6, 14	0.7, 1.6	1.7, 3.1	3.6, 5.9	9.8, 15
50	7	0.6, 1.8	1.6, 3.4	3.4, 6.8	9.6, 18	0.7, 1.8	1.7, 3.5	3.6, 7.2	9.7, 19
100	1	1.2, 1.3	3.1, 2.7	6.8, 5.2	18, 11	1.3, 1.5	3.2, 3.0	7.0, 5.6	18, 11
100	2	1.2, 1.5	3.1, 3.4	6.8, 6.8	18, 15	1.4, 1.8	3.2, 3.6	7.0, 7.0	18, 15
100	5	1.2, 2.2	3.2, 4.8	6.8, 10	18, 26	1.4, 2.4	3.2, 4.8	7.0, 10	18, 27
100	10	1.3, 3.0	3.2, 6.4	6.9, 15	18, 45	1.4, 3.1	3.3, 6.5	7.0, 15	18, 48
250	1	2.2, 2.2	6.8, 4.9	15, 10.0	38, 22	2.5, 2.7	6.2, 5.3	13, 10	36, 22
250	2	2.3, 2.6	6.0, 5.5	13, 13	35, 30	2.6, 3.1	6.2, 6.4	14, 13	33, 30
250	5	2.3, 3.3	6.0, 8.0	13, 20	35, 57	2.7, 3.9	6.3, 8.6	14, 19	35, 56
250	15	2.4, 5.3	6.0, 13	13, 41	36, 147	2.6, 5.7	6.2, 13	14, 31	37, 150
250	25	2.4, 6.8	6.1, 18	13, 62	36, 238	2.7, 7.5	6.3, 20	14, 59	36, 238
500	1	5.2, 4.7	14, 8.5	30, 19	75, 43	5.0, 5.4	12, 9.5	27, 20	77, 46
500	2	4.5, 4.5	11, 9.1	27, 23	74, 61	5.0, 5.9	12, 11	28, 26	75, 63
500	5	4.5, 5.4	12, 13	28, 36	73, 110	5.1, 6.8	12, 14	28, 37	74, 113
500	22	4.5, 9.2	11, 26	28, 98	71, 406	5.1, 11	12, 28	28, 100	74, 415
500	50	4.6, 15	12, 50	29, 207	71, 928	5.3, 17	13, 53	30, 219	74, 928
1000	1	8.9, 7.9	28, 15	59, 33	161, 85	10, 12	29, 20	62, 41	157, 89
1000	2	9.1, 8.4	25, 17	57, 44	158, 119	10, 12	26, 22	59, 50	154, 118
1000	5	9.2, 9.4	25, 22	57, 67	159, 214	10, 13	26, 26	59, 72	154, 208
1000	32	9.2, 16	26, 59	57, 238	158, 1065	11, 20	26, 62	59, 250	155, 1058
1000	100	9.5, 36	27, 163	57, 751	156, 3373	11, 41	27, 170	58, 763	152, 3403

Table 5.9: Time (in milliseconds) to first construct the CPDAG from the DAG and then check for the existence of an adjustment set in that CPDAG, for $P(\textit{unobserved}) = 0$, respectively $P(\textit{unobserved}) = 0.75$.

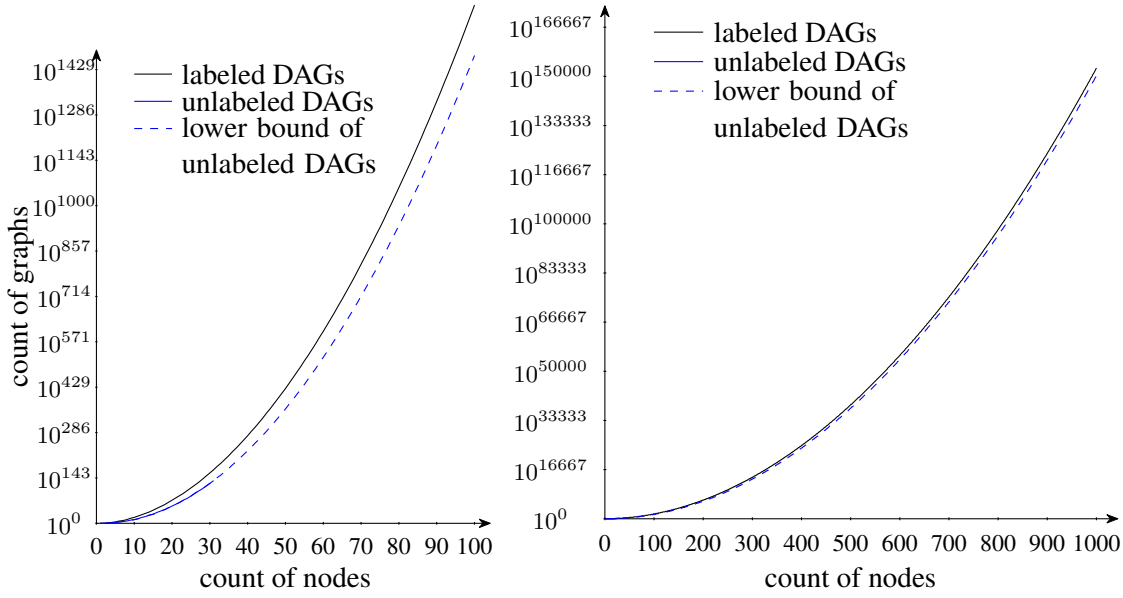


Figure 5.10: Count of different labeled and unlabeled DAGs against the count of nodes n. The left side shows $n \leq 100$, the right side $n \leq 1000$. These counts are sequences A003024 and A003087 in [Oei]. When counting unlabeled DAGs, one counts DAGs that are isometric to each other only once. For example, for $n = 2$, there exists three labeled DAGs: $V_1 \to V_2$, $V_1 \leftarrow V_2$, and $V_1 \quad V_2$ without an edge. Since the first two DAGs can be obtained from each other by changing the names of the nodes, there only exist two unlabeled DAGs.
The count of labeled DAGs is an upper bound of the count of unlabeled DAGs. Since we have not been able to calculate the count of unlabeled DAGs of more than 31 nodes, we calculate a lower bound by dividing the count of labeled DAGs by the factorial of the number of nodes. The factorial of the number of nodes is the count of the permutations of all nodes, that is, the maximal count of possible labelings. Not all of those permutations yield distinct graphs, e.g., there is only one graph with zero edges for any number of nodes. The right side of the figure shows that the lower and upper bound are very close.

5.5 Discussion

In summary, our experimental results show that many causal effects in random DAGs cannot be identified by covariate adjustment. Nevertheless, many of these cases are easily addressed by extending the CBC slightly, and then most effects become identifiable without having to resort to the do-calculus. In MAGs and CPDAGs, there are fewer effects identifiable by adjustment than in DAGs.

We have sampled 10 000 graphs for each parameter tuple, which is a large amount of data, but only a tiny fraction of the total count of DAGs as shown in Figure 5.10. We hope that typical graphs are more likely to be sampled and the results are representative anyway.

It might be worthwhile to repeat the experiments on graphs used in actual epidemiological or economical studies. However, such data is not readily available, which would make it hard to collect enough graphs for quantitative results. Real studies usually involve rather small graphs, possibly to a lack of efficient algorithms for larger graphs. Our algorithms and DAGitty might help future researchers to study larger models.

6 Identification via Instrumental Variables in SEMs

In the previous chapters, we have considered non-parametric models in which a node Y with a single parent X in a causal graph $X \to Y$ represents nothing more than a factor $P(y|x)$ in the factorization of any probability distribution compatible with the graph and $P(y|x)$ can be an arbitrary probability distribution function.

In this chapter, we study linear *Structural Equation Models* (SEMs) in which each random variable is a normally distributed real number and the edges encode linear relationships between the variables. This allows one to not only obtain qualitative information about separation and independence between the random variables, but also precise quantitative information about their correlations from the graphical model. For this reason, SEMs in the form of *linear regression models* are frequently used in social sciences and economics to analyze causal and statistical relations between random variables whose interactions are assumed to be linear [Bol89; Dun75].

Our goal is again to identify the causal effect of variables $\mathbf{X}$ on $\mathbf{Y}$. Of course, the general results of the previous chapters are valid for SEMs, so we already know how to use adjustment sets to calculate the total causal effect in SEMs. Moreover, the additional assumption of linear relationships allows one to identify causal effects in SEMs that cannot be identified in non-parametric models. Unfortunately, no efficient, sound, and complete graphical criterion is known for the identification problem in SEMs, so we investigate sound, non-complete criteria, which provide a way to calculate the causal effect if the criterion is satisfied.

For example, the above DAG $X \to Y$ represents two random variables X and Y with linear equations $X = \varepsilon_X$ and $Y = \lambda X + \varepsilon_Y$. This equation means a unit change of the single parent X will change the mean of Y by λ, so λ is the *direct causal effect* of X on Y. λ is also called *path coefficient* or *parameter*, and a common notation is to write the coefficient between variables as a parameter above the edge between them, e.g., $X \overset{\lambda}{\to} Y$.

direct causal effect
path coefficient
parameter

ε_X and ε_Y are *error terms*, which themselves are (Gaussian) random variables. Thus, the value of Y is not exactly λX and the model is not deterministic. The error terms are independent of all other (observed) variables of the model but might be correlated with the error terms of other variables. The equation is a causal equation, so although a change of X changes the left-hand-side Y, changing Y by an intervention will not change X. Rather an intervention on Y would remove all incoming edges to Y, severing the relationship to the parents of Y and removing them from the equation.

error terms

The mean of Y in this case is $\mathbb{E}[Y] = \mathbb{E}[\lambda X + \epsilon_Y] = \lambda\,\mathbb{E}[X] + \mathbb{E}[\epsilon_Y]$, and the covariance between X and Y is $\mathrm{Cov}(X, Y) = \mathrm{Cov}(X, \lambda X + \varepsilon_Y) = \lambda\,\mathrm{Cov}(X, X)$, because the mean and covariance are linear functions and the error term ε_Y is assumed to be independent of all other observed variables. Hence, $\lambda = \frac{\mathrm{Cov}(X,Y)}{\mathrm{Cov}(X,X)}$ and we can calculate the coefficient λ from the observed data. Here *observed data* means precisely the covariances between all pairs of

identifiable variables. Thus, λ is *identifiable*, and *identified* by the equation $\lambda = \frac{\mathrm{Cov}(X,Y)}{\mathrm{Cov}(X,X)}$.
identified

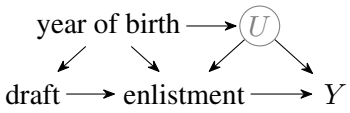
In this case, the identification is trivial because there is no confounding between X and Y, i.e., there is no d-connected path between X and Y not involving the edge $X \to Y$.

Figure 6.1: $\mathcal{G}_1$: The classic IV model. $\mathcal{G}_2$: Z is not an IV, but is a *conditional instrument* given W. U and W are unobserved variables.

If there is confounding between X and Y which cannot be removed by conditioning, like in the SEMs of Figure 6.1, the standard identification approach is to use another variable that *instrumental variable* is not confounded with Y as an *instrumental variable* (*IV* or *instrument*). For example, in $\mathcal{G}_1$ *IV* the covariances between the observed variables are $\mathrm{Cov}(X,Y) = \gamma + \omega_1\omega_2$, $\mathrm{Cov}(Y,Z) = \beta\gamma$, *instrument* and $\mathrm{Cov}(X,Z) = \beta$. We cannot calculate γ from $\mathrm{Cov}(X,Y)$ due to the confounder U and the unknown parameters ω_1 and ω_2, but we can calculate $\gamma = \frac{\mathrm{Cov}(Y,Z)}{\mathrm{Cov}(X,Z)}$. So, Z is the instrumental variable of this example.

If the instrumental variable is also confounded with Y, we can block the paths between it and Y by conditioning. For instance, Z in $\mathcal{G}_2$ is confounded with Y, but by conditioning on W we can identify the direct causal effect γ as $\gamma = \frac{\mathrm{Cov}(Y,Z|W)}{\mathrm{Cov}(X,Z|W)}$. In such cases, Z is a *conditional* *conditional instrument* *instrumental variable* (*conditional instrument*) [Pea01], and we say that W *instrumentalizes* Z. *instrumentalizes*

The idea of applying instruments together with conditioning variables predates its graphical definition. Take the seminal work of Angrist on the labor market impact of voluntary military service [Ang98; AP08, Chapter 4]: During the "Vietnam draft lottery", randomly chosen men were called to serve in the war. As not every drafted person enlisted, this is a classical example of a randomized trial with imperfect compliance: The IV Z, in this case, is the draft, and the independent variable X is enlistment. Several dependent variables Y could be studied with this setup, e.g., the average earnings loss due to enlistment. However, because different numbers of men were drafted for each birth cohort, the IV was only exogenous conditioned on the year of birth (this would be our $\mathbf{W}$ for the conditional instrumental variable). A DAG describing this scenario could look like this:



We will investigate the problems of testing, finding, and enumerating (conditional) IVs, the same problems we have investigated for adjustment sets.

Testing a conditional IV, i.e., deciding whether a given variable is a conditional IV, presents the difficulty of searching for the set $\mathbf{W}$, which is surprisingly hard. We will prove that it is, in fact, an NP-complete problem. However, if the set $\mathbf{W}$ is a subset of the ancestors, the test can be performed in linear time. This implies a new definition of IVs, which we term ancestral IVs. It turns out that an ancestral IV exists if and only if a conditional IV exists in a graph. We use this definition to obtain efficient algorithms to find conditional IVs in $O(n(n+m))$ time.

A different kind of instrumental variables, so called instruments relative to the total effect, has been proposed by Pearl [Pea09]. We investigate the algorithmics of these IVs in the same

way and compare them to conditional IVs. We show that the testing of IVs relative to the total effect remains NP-complete when all variables in the model are observed. On the other hand, conditional IVs can be tested, found, and even enumerated in linear time in fully observed models.

Later we study instrumental sets, which combine multiple IVs to identify causal effects that cannot be identified by just one IV [Bri04; Bri10; BP02]. Each IV in the instrumental set is more general than a classic IV, but less general than a conditional IV.

The next Section 6.1 formally introduces SEMs and basic definitions. Section 6.2 gives criteria to decide whether a variable Z is an instrumental variable, both classical and novel criteria. In Section 6.3, we implement these criteria as efficient algorithms or show that some criteria are NP-complete. Section 6.4 and Section 6.5 investigate instrumental sets in the same way, i.e., Section 6.4 gives the criteria for instrumental sets and Section 6.5 the algorithms as well as an NP-completeness proof.

Scientific Contribution. We have given the first efficient algorithms and complexity results for conditional IVs [ZTL15] and instrumental sets [ZL16a]. The results have also been presented at [ZTL19]. Our runtime improvements for minimal separator algorithms [ZL19] have also improved the runtime of the algorithms for instrumental variables.

6.1 Preliminaries

Mixed graphs, Semi-Markovian models

The causal graph of a SEM is usually a semi-Markovian graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ with directed and bidirected edges. We only consider graphs in which the subgraph of all directed edges is acyclic. Bidirected edges are, however, allowed between any pair of nodes without restriction. The directed edges encode linear relations between observed variables and the bidirected edges represent unobserved confounders as correlations between the error terms. When we label the random variables as $V_1, \ldots, V_n$, the linear equations of the model are: *linear equation model*

$$V_j = \sum_{i \neq j} \lambda_{ji} V_i + \varepsilon_j, \quad j = 1, \ldots, n, \tag{6.1}$$

where parameter λ_{ji} is the *direct causal effect* of variable V_i on variable V_j. If there is no edge from V_i to V_j in $\mathcal{G}$ the parameter λ_{ji} is zero. When $V_1, \ldots, V_n$ are ordered topologically, we have $\lambda_{ji} = 0$ for all $i \geq j$. *direct causal effect*

The *error terms* are assumed to be normally distributed. The correlation $\mathrm{Cov}(\varepsilon_i, \varepsilon_j)$ between two error terms ε_i and ε_j is represented by the coefficient $\omega_{i,j}$ of a bidirected edge $V_i \leftrightarrow V_j$. If there is no edge $V_i \leftrightarrow V_j$, we have $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = \omega_{i,j} = 0$. A pair of correlated error terms $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) \neq 0$ represents an unknown confounder influencing variables V_i and V_j. *error terms*

The variables themselves are all considered as observed in a semi-Markovian SEM, so the observed data consists of the *mean* $\mathbb{E}[V_i]$ and *covariance* $\mathrm{Cov}(V_i, V_j)$ of all variables. For simplicity, we will assume all variables are normalized to $\mathbb{E}[V_i] = 0$ and $\mathrm{Cov}(V_i, V_i) = 1$. After this normalization, the *correlation coefficient* $\rho_{V_i,V_j} = \frac{\mathrm{Cov}(V_i,V_j)}{\sqrt{\mathrm{Cov}(V_i,V_i)\,\mathrm{Cov}(V_j,V_j)}}$ and the *regression coefficient* $r_{V_i,V_j} = \frac{\mathrm{Cov}(V_i,V_j)}{\mathrm{Cov}(V_j,V_j)}$ are all identical to the covariance, i.e., $\mathrm{Cov}(V_i, V_j) = \rho_{V_i,V_j} = r_{V_i,V_j}$. Hence, we can use the terms correlation and covariance as synonyms. *mean* *covariance*

Graph $\mathcal{G}$ is syntactically an ancestral graph with the same d-separation rules, so all separation algorithms of Chapter 3 can be applied to it. If variables V_i and V_j are separated, their covariance $\mathrm{Cov}(V_i, V_j)$ is zero.

Covariance matrix and algebraic interpretation

SEMs can be represented algebraically by adjacency matrices of polynomials. The mixed graph is equivalent to two adjacency matrices: The matrix $\Lambda = (\lambda_{ij})_{ij}$ contains all the coefficients of all directed edges and the matrix $\Omega = (\omega_{ij})_{ij}$ the coefficients of all bidirected edges. An entry of these matrices is zero if the edge is absent, and a monomial if the edge is present.

covariance matrix

When the covariances between every pair of variables are written as a matrix $\Sigma = (\sigma_{ij})_{ij} = (\mathrm{Cov}(V_i, V_j))_{ij}$, the covariances can be calculated as [Bol89]

$$\Sigma = (I - \Lambda)^{-1}\Omega(I - \Lambda)^{-T}, \tag{6.2}$$

where I is the identity matrix. The inverse in the equation can be calculated because, for an acyclic graph, Λ is a triangle matrix. The entries of Σ are rational functions, that is, fractions with polynomial nominator and denominator.

This equation fully encodes all statistical knowledge about a SEM. When $\vec{\epsilon} = (\epsilon_{V_1}, \ldots, \epsilon_{V_n})^T$ is a vector of (Gaussian) random variables normally distributed with covariance matrix Ω and the parameters Λ are arbitrary, then $\vec{\mathbf{V}} = (V_1, \ldots, V_n)^T$ is a vector of random variables normally distributed with covariance matrix Σ. This follows because the value of the variables $\mathbf{V}$ is determined by the error terms as $\vec{\mathbf{V}} = (I - \Lambda)^{-1}\vec{\epsilon}$.

Equation 6.2 is a rational equation system in the variables λ_{ij}, ω_{ij}, and σ_{ij}. A direct causal effect λ_{ij} is now identifiable if and only if we can solve this equation system for λ_{ij} by finding an expression equal to λ_{ij} that only contains variables σ_{ij}.

One can solve this equation system with standard algebraic methods using a computer algebra system (CAS). This provides a sound and complete method to identify any direct causal effect that is identifiable. However, the standard polynomial equation solving algorithms, usually Gröbner bases, have double exponential runtimes and are often too slow to solve the equation system. Gröbner bases have been used for surveys of all graphs with up to four nodes [GPSS10] and five nodes [FDD12]. [GPSS10] notes the runtime varies between seconds and 75 days for graphs with four nodes.

generic solutions

We only search for *generic solutions* that hold for almost all matrices of covariances resp. probability distributions. For example, for the model $\mathcal{G}_1$ of Figure 6.1, the causal effect γ is identified by $\gamma = \mathrm{Cov}(Y, Z)/\mathrm{Cov}(X, Z)$ although $\mathrm{Cov}(X, Z)$ might be 0 in some probability distributions compatible to the model. The usual formal definition of generic solutions is that the subset of parameters for which the solution is invalid must have a Lebesgue measure of zero. In the example, there are four parameters, $(\beta, \gamma, \lambda_1, \lambda_2) \in \mathbb{R}^4$. The solution is invalid only for $\mathrm{Cov}(X, Z) = \beta = 0$, so the set of parameters with an invalid solution is $\{(0, \gamma, \lambda_1, \lambda_2)\}$, which is an algebraic set isometric to $\mathbb{R}^3$, and a three-dimensional subset of a four-dimensional space has measure 0. Every proper algebraic subset has measure zero, and it is likely that the set of not identifiable parameters is always an algebraic set [FDD12].

Wright's Path Analysis

SEMs provide an easy way to calculate the covariances between two variables X and Y via Wright's path analysis [Wri34]. Every active path between X and Y contributes the product of

all its edge coefficients to the covariance. That is when we write the product of the coefficients of all edges on a path π as $\prod_{c:\pi} c$ and the sum over all active paths between X and Y as $\sum_{\pi: X \overset{*}{\nleftrightarrow} Y}$, the covariance is:

$$\mathrm{Cov}(X, Y) = \sum_{\pi: X \overset{*}{\nleftrightarrow} Y} \prod_{c:\pi} c$$

The covariances calculated using Wright's path analysis are of course the same covariances we have in the entries of the matrix Σ of Equation 6.2. Wright's path analysis might be easier to understand than the matrix inversion and matrix product, but an algorithm that enumerates all active paths or walks will require exponential time. The summation over paths is only valid if all variables are normalized to have unit variance. In models with unnormalized variables, one has to sum over walks and include the variance of the fork (or highest node) of each walk as a factor.

SEMs as DAGs

In the remainder of this chapter, we will assume a SEM is given as a DAG $\mathcal{G} = (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E})$ with observed and unobserved variables rather than as a mixed graph with bidirected edges. To apply our results for graphs with bidirected edges, every bidirected edge $V_i \leftrightarrow V_j$ needs to be replaced with $V_i \leftarrow U_{ij} \rightarrow V_j$ where U_{ij} is a new, unique, unobserved node. We will still draw bidirected edges in some exemplary figures.

6.2 Single Instrumental Variables

In this section, we give the formal definitions of instrumental variables, which can identify the causal effect of one variable X on one child variable Y. Table 6.1 presents a summary of the different definitions.

Definition	**Correlation**	**Independence**	**Restriction on W**
6.1 Instrument	$(Z \not\perp\!\!\!\perp X)_{\mathcal{G}}$	$(Z \perp\!\!\!\perp Y)_{\mathcal{G}_c}$	
6.2 Conditional instrument	$(Z \not\perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}}$	$(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_c}$	$\mathbf{W} \subseteq \mathbf{O} \setminus De(Y)$
6.4 Ancestral instrument	$(Z \not\perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}}$	$(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_c}$	$\mathbf{W} \subseteq (\mathbf{O} \setminus De(Y)) \cap An(Y, Z)$
6.8 Active instrument	active path in $\mathcal{G}_c$	$(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_c}$	$\mathbf{W} \subseteq \mathbf{O} \setminus De(Y)$
Relative to the total effect:			
6.3 Instrument r.t.t.t.e.	$(Z \not\perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}}$	$(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_{\overline{X}}}$	$\mathbf{W} \subseteq \mathbf{O}$
6.5 Ancestral instrument r.t.t.t.e.	$(Z \not\perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}}$	$(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_{\overline{X}}}$	$\mathbf{W} \subseteq \mathbf{O} \cap An(Y, Z)$
6.9 Active instrument r.t.t.t.e.	active path in $\mathcal{G}$	$(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_{\overline{X}}}$	$\mathbf{W} \subseteq \mathbf{O}$

Table 6.1: Overview about the different concepts of instrumental variables in Definition 6.1 to Definition 6.9. The correlation conditions are expressed as a graphical criterion. "Active path" means there should exists one path $Z \overset{+}{\nsim} X$ in the graph that is neither blocked by $\emptyset$ nor by $\mathbf{W}$.

Subsection 6.2.1 and Subsection 6.2.2 define the instruments and conditional instruments that we have already discussed in the introduction, Subsection 6.2.3 instruments relative to the total effect. Subsection 6.2.4 and Subsection 6.2.5 introduce a new kind of instrumental variables, which are a special case of conditional IVs and which we will use to develop efficient IV algorithms.

6.2.1 Instrumental Variables

The classic definition of IVs is:

instrumental variable

Definition 6.1 (Instrumental variable)**.** *Z is an* instrumental variable *relative to $X \to Y$ if*

(a) Z is d-connected to X, and

(b) Z is d-separated from Y in $\mathcal{G}_c = \mathcal{G} \setminus (X \to Y)$.

graph $\mathcal{G}_c$

Graph $\mathcal{G}_c$ is a commonly used notation for the DAG $\mathcal{G}$ without the edge $X \to Y$, which is motivated by the notation for SEMs in which the direct causal effect between two random variables is written above the arrow like $X \overset{c}{\to} Y$.

Condition (a) is generically equivalent to "Z correlates with X", so condition (a) can be tested statistically in the observed data itself without knowledge of the graphical model. Condition (b) on the other hand is statistically untestable from observed data, arguably, it cannot even be expressed in statistical language [Pea09].

identified

If Z is an IV relative to $X \to Y$, the causal effect c is *identified* as $c = \frac{\mathrm{Cov}(Y,Z)}{\mathrm{Cov}(X,Z)}$.

6.2.2 Conditional Instruments

A conditional IV generalizes instrumental variables by conditioning on a set $\mathbf{W}$:

conditional instrument

Definition 6.2 (Conditional Instrument [Pea01])**.** *Z is a* conditional instrument *relative to $X \to Y$ if there exists a set $\mathbf{W} \subseteq \mathbf{O}$ such that*

(a) Z is d-connected to X given $\mathbf{W}$,

(b) $\mathbf{W}$ d-separates Z and Y in $\mathcal{G}_c = \mathcal{G} \setminus (X \to Y)$, and

(c) $\mathbf{W}$ consists of non-descendants of Y.

instrumentalizes

When $\mathbf{W}$ and Z satisfy these conditions, we say that $\mathbf{W}$ instrumentalizes Z.

For instance, Z is a conditional instrument in $\mathcal{G}_2$ of Figure 6.1 using $\mathbf{W} = W$. Again, condition (a) is statistically testable, and it implies the existence of a path π from Z to X that is d-connected by $\mathbf{W}$. Conditions (a) and (b) are direct generalizations of the same conditions in the standard IV definition. Restriction (c) is necessary because adjustment for descendants of Y would bias estimation by the reversal paradox.

identified

If Z is a conditional IV relative to $X \to Y$, the causal effect c is *identified* as

$$c = \frac{\mathrm{Cov}(Y, Z \mid \mathbf{W})}{\mathrm{Cov}(X, Z \mid \mathbf{W})}.$$

6.2.3 Instruments Relative to the Total Effect

Pearl has introduced a new kind of instrumental variables which he calls instruments relative to the total effect [GP98; Pea09, Section 7.4]. His definition replaces $(Z \perp\!\!\!\perp Y)_{\mathcal{G}_c}$ by $(Z \perp\!\!\!\perp Y)_{\mathcal{G}_{\overline{X}}}$ and allows descendants of Y in $\mathbf{W}$. Thus, he obtains:

instrument relative to the total effect

Definition 6.3 (Instrument Relative to the Total Effect [Pea09])**.** *A variable Z is an* instrument relative to the total effect *of X on Y if there exists a set of measurements $\mathbf{W}$ such that*

(a) $(Z \not\perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}}$, *and*

(b) $(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_{\overline{X}}}$.

Pearl mentions this definition in the context of generalizing instrumental variables to non-linear systems. However, he does not explain how instruments relative to the total effect can be used to actually calculate a causal effect. We are also not aware of any scientific work using instruments relative to the total effect. Pearl cites [GP98] as the source of this definition, but that paper does not give further information, besides a different formulation of condition (a) as "every path connecting Z and Y must pass through X unless it contains arrows pointing head-to-head". It is possible that his definition is erroneous and cannot be used to calculate causal effects practically. Nevertheless, we will analyze instruments relative to the total effect together with normal conditional instruments since the comparison between them reveals interesting facts about the complexity of instrumental variables.

6.2.4 Ancestral Instruments

When one wants to use conditional IVs, it is difficult to choose the set $\mathbf{W}$. In fact, we will show in Section 6.3.4 that it is an NP-complete problem to find $\mathbf{W}$. In order to obtain efficient algorithms, we need a new kind of IVs whose $\mathbf{W}$ is easier to find. Thus, we give a less general definition of conditional IVs which requires that $\mathbf{W}$ consists only of ancestors of Y or Z.

Definition 6.4 (Ancestral Instrument)**.** *Variable Z is an* ancestral instrument *relative to $X \to Y$ if there exists a set $\mathbf{W} \subseteq \mathbf{O}$ such that* *ancestral instrument*

(a) *Z is d-connected to X given $\mathbf{W}$,*

(b) *$\mathbf{W}$ d-separates Z and Y in $\mathcal{G}_c = \mathcal{G} \setminus (X \to Y)$, and*

(c) *$\mathbf{W}$ consists of ancestors of Y or Z or both which are non-descendants of Y.*

We can restrict Definition 6.3 similarly:

Definition 6.5 (Ancestral Instrument Relative to the Total Effect)**.** *A variable Z is an* ancestral instrument relative to the total effect *of X on Y if there exists a set of measurements $\mathbf{W}$ such that* *ancestral instrument relative to the total effect*

(a) $(Z \not\perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}}$,

(b) $(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_{\overline{X}}}$, *and*

(c) $\mathbf{W} \subseteq An(Y, Z)$.

From this definition, it is obvious that ancestral IVs are a special case of conditional IVs. Also, each standard IV is an ancestral IV (using $\mathbf{W} = \emptyset$).

There exist conditional instruments relative to $X \to Y$ which do not satisfy the conditions of an ancestral conditional instrument; for instance, Z in this DAG with $\mathbf{W} = \{W\}$:

$$Z \to W \leftarrow X \to Y$$

This might seem to imply that ancestral IVs are less "powerful" than generic conditional ones. Importantly and perhaps surprisingly, the following theorem shows that this is not the case. Every causal effect that can be identified with a conditional IV can be identified with an ancestral IV.

Theorem 6.6. *For a given DAG $\mathcal{G}$ and variables X and Y, a conditional IV Z relative to $X \to Y$ exists if and only if an ancestral IV Z' relative to $X \to Y$ exists.*

Theorem 6.7. *For a given DAG $\mathcal{G}$ and variables X and Y, an instrumental variable Z relative to the total effect of X on Y exists if and only if an ancestral instrumental variable Z' relative to the total effect of X on Y exists.*

The combined proof of Theorem 6.6 and Theorem 6.7:

Proof. Let $\mathcal{G}_{\not\perp} = \mathcal{G}_{\perp\!\!\!\perp} = \mathcal{G}_c, \mathbf{O}' = \mathbf{O} \setminus De(Y)$ for the proof of Theorem 6.6 and $\mathcal{G}_{\not\perp} = \mathcal{G}, \mathcal{G}_{\perp\!\!\!\perp} = \mathcal{G}_{\overline{X}}, \mathbf{O}' = \mathbf{O}$ for the proof of Theorem 6.7.

Let Z be a conditional IV relative to $X \to Y$ (resp. the total effect of X on Y), π a path from X to Z in $\mathcal{G}_{\not\perp}$, and $\mathbf{W} \subseteq \mathbf{O}'$ the set opening π and d-separating Y and Z in $\mathcal{G}_{\perp\!\!\!\perp}$. Assume $Z, \pi, \mathbf{W}$ are chosen such that π has a minimum number of colliders and $\mathbf{W}$ has minimum size (i.e., any other path would have at least as many colliders, and if there is a path with the same number of colliders, it is opened by a set at least as large as $\mathbf{W}$).

Then no node W can be removed from $\mathbf{W}$ without opening a path between Z and Y in $\mathcal{G}_{\perp\!\!\!\perp}$ or blocking path π. In the former case, W lies on a path opened by $\mathbf{W} \setminus W$ and we know from Lemma 3.13 that $W \in An(Y \cup Z \cup (\mathbf{W} \setminus W))$.

In the latter case, there is a collider on π only opened by W. Let C be the closest opened collider to X, possibly $C = W$. The walk $\pi[X \overset{*}{\sim} C] \overset{*}{\to} W$ is active given $\mathbf{W} \setminus W$ in $\mathcal{G}_{\not\perp}$, and due to Lemma 3.11, there is a corresponding path between X and W that has fewer colliders than π. This means there exists a path τ between Y and W that is active given $\mathbf{W} \setminus W$ in $\mathcal{G}_{\perp\!\!\!\perp}$ because otherwise W would be a conditional IV relative to $X \to Y$, contradicting that π has a minimum number of colliders.

Now we show that τ ends with $\leftarrow W$, because otherwise the walk $\tau \overset{*}{\leftarrow} \pi[C \overset{*}{\sim} Z] = Y \overset{*}{\sim}\!\!\to W \overset{*}{\leftarrow} \pi[C \overset{*}{\sim} Z]$ would be active given $\mathbf{W}$ in $\mathcal{G}_{\perp\!\!\!\perp}$ and Z would not be a conditional instrument. Clearly, $\pi[C \overset{*}{\sim} Z]$ is active in $\mathcal{G}_{\not\perp}$ given $\mathbf{W}$ and cannot contain an edge adjacent to X and removed in $\mathcal{G}_{\perp\!\!\!\perp}$ (here we discuss the case $\mathcal{G}_{\perp\!\!\!\perp} = \mathcal{G}_{\overline{X}}$ as it is trivial for $\mathcal{G}_{\not\perp} = \mathcal{G}_{\perp\!\!\!\perp}$), since it is a subpath of π starting at X, so if it is not also active in $\mathcal{G}_{\perp\!\!\!\perp}$, there is a collider C' on π that is only opened by a descendant of X. Then there is an active path $C' \overset{*}{\to} X$, and $X \overset{*}{\leftarrow} \pi[C' \overset{*}{\sim} Z]$ has fewer colliders than π, which contradicts our choice of π. For the same reason, the path $W \overset{*}{\leftarrow} C$ does not contain X and is active given $\mathbf{W} \setminus W$.

So, if W is not an ancestor of Y, τ contains a collider (as a descendant of W) that has a descendant in $\mathbf{W} \setminus W$. Then every node in $\mathbf{W}$ is either an ancestor of Y or Z, or is an ancestor of some other node in $\mathbf{W}$. Since DAGs do not contain cycles, every node in $\mathbf{W}$ is an ancestor of Y or Z. Thus, Z is an ancestral instrument.

The other direction is trivial. □

The result that we lose nothing by restricting ourselves to ancestral IVs reminds one of Theorem 4.44 that if a causal effect can be identified by any adjustment, then it can also be identified by adjusting only for ancestors of the variables of interest. Also for finding minimal and minimum separators between variables in Section 3.3, it was sufficient to only consider their ancestors.

Section 6.3 will show that ancestral instruments are algorithmically appealing: unlike non-ancestral instruments, they and their conditioning set $\mathbf{W}$ can be found efficiently in a given DAG.

6.2.5 Active Instruments

We have also studied another restriction of conditional instruments – the active instruments:

Definition 6.8 (Active Conditional Instrument)**.** *Variable Z is an* active conditional instrument *relative to $X \to Y$ if there exist a set $\mathbf{W}$ and a path π from Z to X such that* *active conditional instrument*

(a) $\mathbf{W}$ *does not block the path π in $\mathcal{G}_c = \mathcal{G} \setminus (X \to Y)$,*

(b) $\mathbf{W}$ *d-separates Z from Y in $\mathcal{G}_c$ and $\mathbf{W}$ consists of non-descendants of Y, and*

(c) path π is active in $\mathcal{G}_c$ given $\emptyset$.

Obviously, every active conditional instrument is a conditional instrument. An active conditional instrument is also an ancestral instrument since $\mathbf{W}$ remains a separator when we remove all nodes outside $An(Y, Z)$ from $\mathbf{W}$ due to Lemma 3.18 and removing nodes will not block path π, which has no colliders.

Active instruments are not of much interest on their own, however, we will later see that they have an interesting relation to instrumental sets (see Subsection 6.4.4).

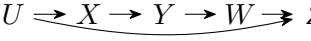
One can also define an active version of instruments relative to the total effect:

Definition 6.9 (Active Instrument Relative to the Total Effect)**.** *Variable Z is an* active conditional instrument *relative to the total effect of X on Y if there exist a set $\mathbf{W}$ and a path π from Z to X such that* *active conditional instrument*

(a) $\mathbf{W}$ *does not block the path π in $\mathcal{G}$,*

(b) $(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}_{\overline{X}}}$*, and*

(c) path π is active in $\mathcal{G}$ given $\emptyset$.

This is also an ancestral instrument relative to the total effect although we are not aware of any purpose of this definition. Since it allows descendants of Y in $\mathbf{W}$, there is no relation to instrumental sets. These descendants are also the reason why there exist active (ancestral) instruments relative to the total effect that are not active (ancestral) instruments. For example in the DAG  with $\mathbf{W} = \{W\}$ and $\pi = Z \leftarrow U \to X$, Z is an instrument of the former kind, but not of the latter kind.

6.3 Algorithmics of Instrumental Variables

In this section, we will analyze the following problems, given a DAG $\mathcal{G}$ and variables X and Y:

Testing Given a variable Z, can we find a set $\mathbf{W} \subseteq \mathbf{O}$ that renders Z into an IV?

Finding Can we find a variable Z and a set $\mathbf{W} \subseteq \mathbf{O}$ that renders Z into an IV?

Enumerating Can we find a maximum set of variables $\mathbf{Z}$ such that, for each $Z \in \mathbf{Z}$, there exists a set $\mathbf{W} \subseteq \mathbf{O}$ that renders Z into an IV?

Graph with unobserved nodes, $\mathbf{O} \subsetneq \mathbf{V}$

Definition	**Testing**	**Finding**	**Enumerating**
6.1 Instrument	$O(n+m)$	$O(n+m)$	$O(n+m)$
6.2 Conditional instrument	NP-complete	$O(n(n+m))$	NP-complete
6.4 Ancestral instrument	$O(n+m)$	$O(n(n+m))$	$O(n(n+m))$
6.3 Instrument (Total effect)	NP-complete	$O(n(n+m))$	NP-complete
6.5 Ancestral instrument (Total effect)	$O(n+m)$	$O(n(n+m))$	$O(n(n+m))$

Fully observed DAG with $\mathbf{O} = \mathbf{V}$, or **partially observed DAG** with $Pa(Y) \subseteq \mathbf{O}$:

Definition	**Testing**	**Finding**	**Enumerating**
6.1 Instrument	$O(n+m)$	$O(n+m)$	$O(n+m)$
6.2 Conditional instrument	$O(n+m)$	$O(n+m)$	$O(n+m)$
6.4 Ancestral instrument	$O(n+m)$	$O(n+m)$	$O(n+m)$
6.3 Instrument (Total effect)	NP-complete	$O(n(n+m))$	NP-complete
6.5 Ancestral instrument (Total effect)	$O(n+m)$	$O(n(n+m))$	$O(n(n+m))$

Table 6.2: Algorithmic complexity of finding or testing instrumental variables. Testing means, given X, Y, and Z, find one set $\mathbf{W}$; finding means, given X and Y, find one pair Z and $\mathbf{W}$; enumerating means, given X and Y, find all possible Z, each Z with its own $\mathbf{W}$.

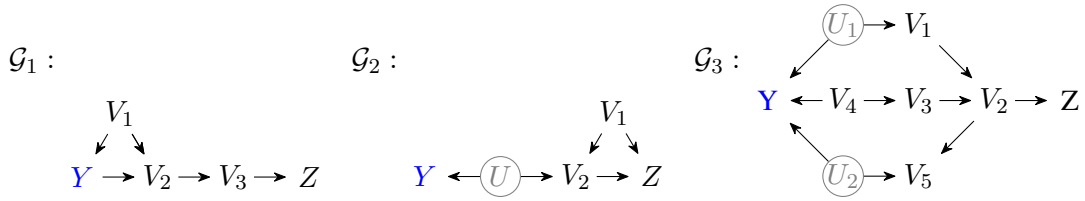


Figure 6.2: Three DAGs with unobserved variables $\{U, U_1, U_2\}$. The nearest separators relative to (Y, Z) are $\{V_2\}$ and $\{V_1, V_2\}$ in $\mathcal{G}_1$, $\{V_1, V_2\}$ in $\mathcal{G}_2$, and $\{V_1, V_4\}$ in $\mathcal{G}_3$.

Each problem is defined separately for each definition of IV, i.e., IV in the list above can mean a conditional IV relative to $X \to Y$, an IV relative to the total effect of X on Y, an ancestral conditional IV, or an ancestral IV relative to the total effect.

Table 6.2 summarizes our results.

To solve one of these problems, it is necessary to find a set $\mathbf{W}$ d-separating Y and Z as well as a path π between X and Z, such that $\mathbf{W}$ does not block π. If we know $\mathbf{W}$, it is not hard to find π, and if we know π, it is not hard to find $\mathbf{W}$, as d-separating paths and sets are well-understood. However, finding both together can be hard.

We will thus start by introducing the concept of nearest separators, separators that do not block any unnecessary paths, i.e., they only block paths that must be blocked by every separator. Then we know a nearest separator does not block π, since any other separator will also block π. Yet nearest separators do not depend on π and can be found without knowing π.

6.3.1 Nearest Separators

A separator relative to (Y, Z) needs to block every path π' between Y and Z at a non-collider on that path π'. The intuitive idea of a nearest separator is to choose non-colliders that are close to Y. So Y is surrounded by a set of nodes $\mathbf{W}$, and any path that reaches one of those nodes W will reach Y unless it is blocked at W. Since those nodes of $\mathbf{W}$ are so close to Y, any path that is not blocked by every separator can move around $\mathbf{W}$ without intersecting $\mathbf{W}$.

Definition 6.10. *Given nodes Y and Z in $\mathbf{V}$, a set $\mathbf{W} \subseteq \mathbf{O} \setminus \{Y, Z\}$ is a* nearest separator *relative to (Y, Z) iff* *nearest separator*

(i) $\mathbf{W}$ *d-separates Y and Z,*

(ii) *for any $W \in \mathbf{W}$ and any set $\mathbf{W}' \subseteq \mathbf{O} \setminus \{W, Y, Z\}$, it holds:*
if $\mathbf{W}'$ does not d-separate W and Z, then $\mathbf{W}'$ does not d-separate Y and Z.

A nearest separator is not necessarily a minimal separator and a minimal separator is not necessarily a nearest separator. For example, the sets $\{V_2\}$ and $\{V_1, V_2\}$ in graph $\mathcal{G}_1$ in Figure 6.2 are (the only) nearest separators, whereas $\{V_2\}$ and $\{V_3\}$ are (the only) minimal separators. A nearest separator can include V_1 because a path π from V_1 to Z can always be extended to a path $Y \leftarrow \pi$ from Y to Z. A nearest separator does not need to include V_1, since such a path π to V_1 is already blocked at V_2. A nearest separator cannot include V_3, because any path from V_3 to Y can be blocked at V_2.

A more complex example is the graph $\mathcal{G}_2$ in Figure 6.2 with an unobserved node $U \notin \mathbf{O}$, where the only nearest separator is $\{V_1, V_2\}$. V_2 needs to be included since the path $Y \leftarrow U \rightarrow V_2$ cannot be blocked at an unobserved node, and V_1 needs to be included because the inclusion of V_2 opens V_2 as a collider. Similarly, in $\mathcal{G}_3$ the nearest separator is $\{V_1, V_4\}$ because the path through node U_1 needs to be blocked at V_1. V_5 must not be included in the nearest separator, since it is not on an active path Y and Z, although it is reachable from both sides. The only minimum separator in $\mathcal{G}_3$ is $\{V_2\}$, which shows minimum and nearest separators are entirely unrelated concepts.

Nearest separators help to find instrumental variables because we know that a nearest separator relative to (Y, Z) will not block a path between X and Z unnecessarily, regardless of which node is X.

Theorem 6.11. *Let $\mathbf{W}$ be a nearest separator relative to (Y, Z). Then, for every $X \in \mathbf{V}$ and every set $\mathbf{W}' \subseteq \mathbf{O} \setminus \{X, Y, Z\}$ with $(Z \perp\!\!\!\perp Y \mid \mathbf{W}')_{\mathcal{G}}$ that opens a path π between X and Z, the set $\mathbf{W} \cup \mathbf{W}'$ has the following properties:*

1. *$\mathbf{W} \cup \mathbf{W}'$ opens the path π,*
2. *$X \notin \mathbf{W} \subseteq \mathbf{W} \cup \mathbf{W}'$, and*
3. *$(Z \perp\!\!\!\perp Y \mid \mathbf{W} \cup \mathbf{W}')_{\mathcal{G}}$.*

Proof. Let X be a node in $\mathbf{V}$, and let π be a path connecting X and Z such that π is open given a set $\mathbf{W}' \subseteq \mathbf{O}$ with $(Z \perp\!\!\!\perp Y \mid \mathbf{W}')_{\mathcal{G}}$.

If π is not open given $\mathbf{W} \cup \mathbf{W}'$, there is a non-collider on π in $\mathbf{W}$. Let $W \in \mathbf{W} \setminus \mathbf{W}'$ be the non-collider closest to Z on π. Then $\pi[W \overset{*}{\sim} Z]$ is not blocked by $\mathbf{W}'$, so $\mathbf{W}'$ would not d-separate Y and Z.

If $X \in \mathbf{W} \cup \mathbf{W}'$, we have $X \in \mathbf{W}$, and $\mathbf{W}'$ would not d-separate Y and Z since it does not d-separate X and Z.

If $\mathbf{W} \cup \mathbf{W}'$ would not d-separate Y and Z, there is a path π' between Y and Z that is open given $\mathbf{W} \cup \mathbf{W}'$, but not given $\mathbf{W}$ or $\mathbf{W}'$. Then no non-collider is in $\mathbf{W} \cup \mathbf{W}'$, so neither $\mathbf{W}$ nor $\mathbf{W}'$ opens all colliders or π' would be open given $\mathbf{W}$ or $\mathbf{W}'$. Let C be the collider on π' that is closest to Z, opened by $\mathbf{W}$ and not opened by $\mathbf{W}'$. Every collider closer to Z is either opened by both or only by $\mathbf{W}'$, so $\pi'[C \overset{*}{\sim} Z]$ is active given $\mathbf{W}'$. Since C is opened by $\mathbf{W}$, it has a descendant $W \in \mathbf{W}$ and the path $C \overset{*}{\to} W$ is open given $\mathbf{W}'$. So $W \overset{*}{\leftarrow} \pi'[C \overset{*}{\sim} Z]$ is open given $\mathbf{W}'$, and $\mathbf{W}'$ would not d-separate Y and Z.

□

If the node X is Z in the above theorem, π is a path of length 0 and $\mathbf{W}'$ can be any separator between Y and Z. The node X cannot be Y, because then there is no $\mathbf{W}'$ that separates Y and Z while still leaving an open path between X and Z.

An informal and intuitive interpretation of the theorem is: If a nearest separator $\mathbf{W}$ blocks a path between an X and Z (at a non-collider), every other separator between Y and Z blocks that path, too. This is because the nodes of a nearest separator are the nearest to Y, so all other separators have nodes that are nearer to Z, and thus stand in the way of paths trying to reach Z. Hence, if $\mathbf{W}$ blocks all paths between an X and Z, X cannot be an instrumental variable.

The theorem is very technically and states properties of $\mathbf{W} \cup \mathbf{W}'$, because the path π can start at any node and contain colliders that cannot be opened by $\mathbf{W}$ alone, because almost any node with two parents can be one of those colliders. To use a nearest separator, one needs to find a set $\mathbf{W}'$ which might still be hard, so we would prefer to have a statement about properties of $\mathbf{W}$ itself, a statement like $\mathbf{W}$ does not block π. This is not true due to the arbitrary colliders, but if we restrict the colliders to be ancestors of Y and Z, we can show that if there is a path π not blocked by $\mathbf{W}'$, there is a path π' not blocked by $\mathbf{W}$:

Corollary 6.12. *Given nodes Y and Z in $\mathbf{V}$, and a nearest separator $\mathbf{W}$ relative to (Y, Z), for every $X \in \mathbf{V}$ for which there is a set $\mathbf{W}' \subseteq \mathbf{O} \cap An(Y, Z) \setminus \{X, Y, Z\}$ with $(Z \perp\!\!\!\perp Y \mid \mathbf{W}')_{\mathcal{G}}$ and $(Z \not\perp\!\!\!\perp X \mid \mathbf{W}')_{\mathcal{G}}$, the set $\mathbf{W}$ does not d-separate X and Z.*

Proof. Let π be a path connecting X and Z given $\mathbf{W}'$. Assume $\mathbf{W}$ d-separates X and Z, i.e., $\mathbf{W}$ does not open π. From Theorem 6.11, we know $\mathbf{W} \cup \mathbf{W}'$ opens π, so every non-collider on π is not in $\mathbf{W}$, so there is a collider that is not opened by $\mathbf{W}$.

All colliders opened by $\mathbf{W}'$ are ancestors of Y or Z, so, for every unopened collider C, there is a path $C \overset{*}{\to} Y$ or $C \overset{*}{\to} Z$ not blocked by $\mathbf{W}$. If every unopened collider is an ancestor of Y, let C be the unopened collider closest to Z. Then $\pi[C \overset{*}{\sim} Z]$ is open given $\mathbf{W}$ as is $Y \overset{*}{\leftarrow} \pi[C \overset{*}{\sim} Z]$, so $\mathbf{W}$ is not a separator. If every unopened collider is an ancestor of Z, let C be the unopened collider closest to X. Then $\pi[X \overset{*}{\sim} C]$ is open given $\mathbf{W}$ and $\pi[X \overset{*}{\sim} C] \overset{*}{\to} Z$ is open given $\mathbf{W}$. Otherwise, let C_Y be an unopened collider in $An(Y)$ and C_Z an unopened collider in $An(Z)$, chosen such that no other unopened collider occurs between C_Y and C_Z on π. Then $\pi[C_Y \overset{*}{\sim} C_Z]$ (respectively $\pi[C_Z \overset{*}{\sim} C_Y]$) is not blocked by $\mathbf{W}$ as is $Y \overset{*}{\leftarrow} \pi[C_Y \overset{*}{\sim} C_Z] \overset{*}{\to} Z$, so $\mathbf{W}$ is not a separator. □

Now that we have motivated nearest separators, the question remains, how do we find such a wondrous thing?

In [ZTL15], we have presented a greedy algorithm that iteratively searches paths from Y to Z in the moral graph $(\mathcal{G}_{An(Y,Z)})^m$ and forms a nearest separator from the first observed nodes

of all these paths. In Appendix C, we prove that it is also possible to use paths in $\mathcal{G}$ directly, leading to a rather straight-forward algorithm:

function FIND-NEAREST-SEPARATOR-SLOW($\mathcal{G}, Y, Z$)
  $\mathbf{W} := \emptyset$
  while $\exists \pi :=$ a path from Y to Z active given $\mathbf{W}$ **do**
    if $\exists V \in \pi$: $V \in \mathbf{O} \setminus \{Y, Z\}$ and V is not a collider on π **then**
      $\mathbf{W} := \mathbf{W} \cup \{\text{first such } V \text{ on } \pi\}$
    else
      return $\perp$
  return $\mathbf{W}$

Despite the simplicity of this algorithm, it has a runtime of $O(n(n+m))$ because it searches an entire path π from Y to Z for every node in the resulting set. However, only the first observed node is actually used and the remaining nodes on the path are ignored. Thus, the runtime can be improved by just finding the nodes reachable from Y and Z without explicitly constructing the path. This suggests a breadth-first-search approach that starts at Y, visits adjacent nodes, continues through unobserved nodes, and stops at observed nodes after adding them to $\mathbf{W}$, but great care needs to be taken to not add unnecessary nodes. For example, in a DAG $V \leftarrow Y \leftarrow Z$ or $Y \rightarrow V \leftarrow Z$ the node V is reachable from Y and Z, but it must not be in a nearest separator, since it is not on an active d-path from Y to Z. On the other hand, adding nodes to $\mathbf{W}$ might open other paths that were not open at the beginning, e.g., in the DAG $\mathcal{G}_2$ of Figure 6.2 the node V_1 is not even reachable from Y given $\emptyset$, but it must be included in the nearest separator.

It turns out that the relevant paths are directed paths to Y or Z, i.e., nearest separators consist of ancestors of Y or Z that are reachable from Y without visiting observed non-colliders. Although nearest separators and minimal separators are orthogonal concepts, this is actually the same condition that we have shown for minimal separators in Section 3.3.2, and also the idea behind ancestral IVs. So we already have an algorithm to find a nearest separator:

function FIND-NEAREST-SEPARATOR($\mathcal{G}, Y, Z$)
  return FINDMINSEP($\mathcal{G}, Y, Z, \emptyset, \mathbf{O}$)

Proposition 6.13. *Given nodes Y and Z, algorithm* FIND-NEAREST-SEPARATOR(Y, Z) *finds a nearest separator $\mathbf{W}$ relative to (Y, Z) with $\mathbf{W} \subseteq An(Y, Z)$ if Y and Z are d-separable, or returns $\perp$ otherwise. The runtime is $O(n + m)$.*

Proof. With the first call to REACHABLE, algorithm FINDMINSEP computes a set $\mathbf{Z}'' \subseteq \mathbf{A} = pAn(Y \cup Z) = An(Y \cup Z)$ in a DAG on which every node $W \in \mathbf{Z}''$ is reachable from Y by a walk only containing nodes in $\mathbf{A}$ and every non-collider is not in $\mathbf{Z}' = \mathbf{O} \cap (\mathbf{A} \setminus (Y \cup Z))$. Y and Z can only occur as end nodes on the walk. Thus, for every $W \in \mathbf{Z}''$, there exists a walk w_W that is active given $\mathbf{Z}'' \setminus W$ and contains no observed non-collider. These paths still exist after the removal of further nodes by the second call to REACHABLE. Let $\mathbf{W}$ be the final set returned by FINDMINSEP.

If $\mathbf{W}$ is not a nearest separator, there is a $W \in \mathbf{W}$ and another separator $\mathbf{W}' \subseteq \mathbf{O} \setminus \{W, Y, Z\}$ with $W \not\perp\!\!\!\perp Z \mid \mathbf{W}'$ and $Y \perp\!\!\!\perp Z \mid \mathbf{W}'$. Let π be a path from W to Z d-connected given $\mathbf{W}'$.

Without an observed non-collider on w_W, the combined walk $w_W\pi : Y \overset{*}{\rightsquigarrow} Z$ contains no node of $\mathbf{W}'$ as a non-collider. All colliders on w_W and W are in $An(Y \cup Z)$, and all colliders on π are in $An(\mathbf{W}')$, so all colliders on $w_W\pi$ are in $An(Y \cup Z \cup \mathbf{W}')$. Due to Lemma 3.12, $w_W\pi$ is active given $\mathbf{W}'$ and $\mathbf{W}'$ is not a separator. □

Corollary 6.14. *If a set* $\mathbf{W}$ *returned by* FIND-NEAREST-SEPARATOR(Y, Z) *contains a descendant of* Y*, then every set that* d*-separates* Y *and* Z *contains a descendant of* Y*.*

Proof. Let $D \in \mathbf{W}$ be the descendant of Y. Since the graph contains no cycles, D is not an ancestor of Y. Hence, D is an ancestor of Z and there is a path $Y \dashrightarrow D \dashrightarrow Z$. This path can only be blocked by descendants of Y. □

6.3.2 Finding ancestral and conditional instrumental variables

Having introduced the concept of nearest separators and having shown how to find them, we are now ready to present an algorithm to test whether a certain variable is an ancestral IV.

function WITNESS-ANCESTRAL-INSTRUMENT($\mathcal{G} = (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E}), X, Y, Z$)
  $\mathcal{G}_c := (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E} \setminus (X \to Y))$
  $\mathbf{W} :=$ FIND-NEAREST-SEPARATOR$(\mathcal{G}_c, Y, Z)$
  if $(\mathbf{W} = \bot) \vee (\mathbf{W} \cap De(Y) \neq \emptyset) \vee (X \in \mathbf{W}) \vee (Z \perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}_c}$ **then**
    return $\bot$
  return $\mathbf{W}$

Theorem 6.15. *For given nodes* X*,* Y*, and* Z *in a DAG* $\mathcal{G}$*, algorithm* WITNESS-ANCESTRAL-INSTRUMENT *returns a set of variables* $\mathbf{W}$ *that satisfies the properties of an ancestral conditional instrument relative to* $X \to Y$ *if such a set exists; otherwise, the algorithm returns* $\bot$*. The running time is* $\mathcal{O}(n + m)$*.*

Proof. We first prove that if the algorithm returns a set $\mathbf{W} \neq \bot$, it has found $\mathbf{W}$ satisfying the conditions of Definition 6.4. $\mathbf{W}$ is a nearest separator $\subseteq An(Y, Z)$, and thus d-separates Y and Z in $\mathcal{G}_c$. The conditions of $\mathbf{W}$ not d-separating X and Z in $\mathcal{G}_c$ as well as $\mathbf{W}$ containing no descendants of Y are explicitly tested by algorithm WITNESS-ANCESTRAL-INSTRUMENT.

Next, let $\mathbf{A} = \mathbf{O} \cap An(Y, Z) \setminus \{X, Y, Z\}$. Assume Z is an ancestral conditional instrument, i.e., there exists a set $\mathbf{W}' \subseteq \mathbf{A} \setminus De(Y)$ such that $(Y \perp\!\!\!\perp Z \mid \mathbf{W}')_{\mathcal{G}_c}$ and a path π_0 between X and Z which is open in $\mathcal{G}_c$ given $\mathbf{W}'$. Then the algorithm returns a set $\mathbf{W}$: Due to Lemma 6.13 and the definition of nearest separators, FIND-NEAREST-SEPARATOR returns a set $\mathbf{W} \subseteq \mathbf{A}$ with $X \notin \mathbf{W}$. Due to Lemma 6.12, $\mathbf{W}$ does not d-separate X and Z. $\mathbf{W}$ does not contain a descendant $D \in De(Y)$ or $\mathbf{W}'$ would contain D as well due to Corollary 6.14.

The runtime follows from the runtime of algorithm FIND-NEAREST-SEPARATOR, elementary set/graph operations and a d-separation test. □

Further, we can use algorithm WITNESS-ANCESTRAL-INSTRUMENT to find a conditional instrumental variable relative to $X \to Y$.

function FIND-CONDITIONAL-INSTRUMENT($\mathcal{G} = (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E}), X, Y$)
  for all Z in $\mathbf{O} \setminus (X \cup De(Y))$ **do**
    Let $\mathbf{W} :=$WITNESS-ANCESTRAL-INSTRUMENT$(\mathcal{G}, X, Y, Z)$
    if $\mathbf{W} \neq \bot$ **then return** $(Z, \mathbf{W})$
  return $\bot$

The soundness of the algorithm and its time complexity $\mathcal{O}(n(n + m))$ follow from Theorem 6.15. The completeness is a consequence of Theorem 6.6. We obtain the following result.

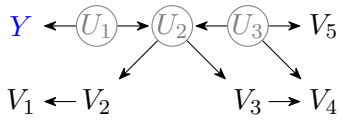


Figure 6.3: A DAG with three unobserved nodes $\{U_1, U_2, U_3\}$. The minimal, nearest separator relative to (Y, V_1) is $\{V_2\}$. The only nearest separator relative to (Y, V_5) is $\{\}$.

Corollary 6.16. *There exists an algorithm which, given X and Y, returns a node Z and a node set $\mathbf{W}$ in time $\mathcal{O}(n(n+m))$ such that $\mathbf{W}$ instrumentalizes Z if such $\mathbf{W}$ and Z exist; otherwise, it returns $\perp$.*

These algorithms are complete for effect identification using conditional IVs in the same sense our algorithms of Chapter 4 are complete for adjustment sets: if it is possible to estimate a causal effect in a DAG using a conditional IV, then we can find such an IV using our algorithm.

Algorithm FIND-CONDITIONAL-INSTRUMENT spends most of the time recalculating the nearest separator for different Z candidates, which begs the question of whether it is possible to improve the runtime by caching the calculation and avoiding the recalculation. If one could find a nearest separator between Y and Z from a nearest separator between Y and some other node Z' in *constant time*, one could find the ancestral instrument in linear time. Unfortunately, there is no obvious way to design such an algorithm. For example, Figure 6.3 shows a problematic DAG. It is necessary to include V_2 in a separator between Y and V_1. However, V_2 or V_3 must not be included in a separator between Y and V_5 (or V_4), as it would open the path through U_3. These cases cannot be distinguished in constant time, because they look locally exactly the same, both V_2 and V_3 are children of U_2 and have one other child.

Recall that algorithm FIND-NEAREST-SEPARATOR finds a nearest separator by visiting reachable ancestors of Y and Z, until it reaches observed non-colliders. The obvious modifications to find a nearest separator without knowing Z are either to visit ancestors of Y or visit all descendants of ancestors of Y. Visiting only ancestors of Y will not work, because it would not include V_2 in the example. Visiting the descendants of the ancestors would fail when it includes both V_2 and V_3.

Nevertheless, in the special case of graphs without unobserved nodes, ancestral instruments can be found in linear-time (see Subsection 6.3.6).

6.3.3 Instrumental Variables Relative to the Total Effect

The same methods can be used to find instrumental variables relative to the total effect:

function WITNESS-ANCESTRAL-INSTRUMENT-TOTAL-EFFECT($\mathcal{G}, X, Y, Z$)
  $\mathcal{G}_{\overline{X}} := (\mathbf{O} \setminus X \subseteq \mathbf{V}, \mathbf{E} \setminus (Pa(X) \to X))$
  $\mathbf{W} :=$ FIND-NEAREST-SEPARATOR($\mathcal{G}_{\overline{X}}, Y, Z$)
  if $(\mathbf{W} = \perp) \vee (Z \perp\!\!\!\perp X \mid \mathbf{W})_{\mathcal{G}}$ **then**
    return $\perp$
  return $\mathbf{W}$

Theorem 6.17. *For given nodes X, Y, and Z in a DAG $\mathcal{G}$, algorithm* WITNESS-ANCESTRAL-INSTRUMENT-TOTAL-EFFECT *returns a set of variables $\mathbf{W}$ that satisfies the properties of an ancestral instrument relative to the total effect of X on Y if such a set exists; otherwise, it returns $\perp$. The running time of the algorithm is $\mathcal{O}(n+m)$.*

Proof. We first prove that if the algorithm returns a set $\mathbf{W} \neq \perp$, it has found $\mathbf{W}$ satisfying the conditions of Definition 6.5. $\mathbf{W}$ is a nearest separator $\subseteq An(Y, Z)$, and thus d-separates Y and Z in $\mathcal{G}_{\overline{X}}$. The condition of $\mathbf{W}$ not d-separating X and Z in $\mathcal{G}$ is explicitly tested by algorithm WITNESS-ANCESTRAL-INSTRUMENT-TOTAL-EFFECT.

Next, let $\mathbf{A} = \mathbf{O} \cap An(Y, Z) \setminus \{X, Y, Z\}$. Assume Z is an ancestral instrument, i.e., there exists a set $\mathbf{W}' \subseteq \mathbf{A}$ such that $(Y \perp\!\!\!\perp Z \mid \mathbf{W}')_{\mathcal{G}_{\overline{X}}}$, and a path π_0 between X and Z which is open in $\mathcal{G}$ given $\mathbf{W}'$. Assume π_0 has the minimal number of colliders. If π_0 starts with $X \leftarrow P$, let $\pi_0' = \pi_0[P \overset{*}{\sim} Z]$; otherwise, let $\pi_0' = \pi_0$. π_0' is also open in $\mathcal{G}_{\overline{X}}$; otherwise, there was a collider C that is an ancestor of X, but then $X \overset{+}{\leftarrow} \pi_0[C \overset{*}{\sim} Z]$ would have fewer colliders than π_0. Algorithm FIND-NEAREST-SEPARATOR returns a set $\mathbf{W} \subseteq \mathbf{A}$ that does not block π_0' due to Corollary 6.12 and does not contain P in the case of $\pi_0' \neq \pi_0$. Therefore, $\mathbf{W}$ does not block π_0 in $\mathcal{G}$, and the algorithm returns a set $\mathbf{W}$.

The runtime follows from the runtime of algorithm FIND-NEAREST-SEPARATOR, elementary set/graph operations, and a d-separation test. □

function FIND-INSTRUMENT-TOTAL-EFFECT($\mathcal{G}, X, Y$)
  for all Z in $\mathbf{O} \setminus \{X, Y\}$ **do**
    Let $\mathbf{W}$:=WITNESS-ANCESTRAL-INSTRUMENT-TOTAL-EFFECT($\mathcal{G}, X, Y, Z$)
    if $\mathbf{W} \neq \perp$ **then return** $(Z, \mathbf{W})$
  return $\perp$

The soundness of the algorithm and its time complexity $\mathcal{O}(n(n+m))$ follows from Theorem 6.17. The completeness is a consequence of Theorem 6.7. We obtain the following result.

Corollary 6.18. *There exists an algorithm which, given X and Y, returns a node Z and a node set $\mathbf{W}$ in time $\mathcal{O}(n(n+m))$ such that $\mathbf{W}$ instrumentalizes Z relative to the total effect if such $\mathbf{W}$ and Z exist; otherwise, it returns $\perp$.*

6.3.4 Instrumentalization is NP-hard in general

We have now solved the problem posed in the previous section: find a variable Z and a set $\mathbf{W}$ such that $\mathbf{W}$ instrumentalizes Z. Now it is natural to wonder about a slightly different problem: given Z, find a set $\mathbf{W}$ that instrumentalizes Z. We refer to this as the *instrumentalization problem* or testing an instrumental variable. Intuitively, this new problem might seem to be easier than finding an IV because Z is already fixed and, unlike in FIND-CONDITIONAL-INSTRUMENT, there is no need to check multiple candidate Z. Perhaps surprisingly, the opposite turns out to be true: Instrumentalization is computationally *harder* than finding an IV.

instrumentalization problem

Theorem 6.19. *Determining whether, for given $X, Y, Z \in \mathbf{V}$ in a DAG $\mathcal{G}$ with $\mathbf{O} = \mathbf{V}$, node Z is an instrument relative to the total effect of X on Y is an NP-complete problem.*

Proof. Obviously, the conditions of Definition 6.3 can be verified in polynomial time after guessing $\mathbf{W} \subseteq \mathbf{O}$. Thus, the problem is in NP. To prove the NP-hardness, we show a reduction from the 3-SAT problem, which is the canonical complete problem for NP [GJ79b].

Assume $\varphi = \bigwedge_{j=1}^{m} \mathcal{C}_j$ is an instance of 3-SAT, which is a Boolean formula in conjunctive normal form over n variables $x_1, \ldots, x_n$ where each clause $\mathcal{C}_j$ is limited to exactly three literals

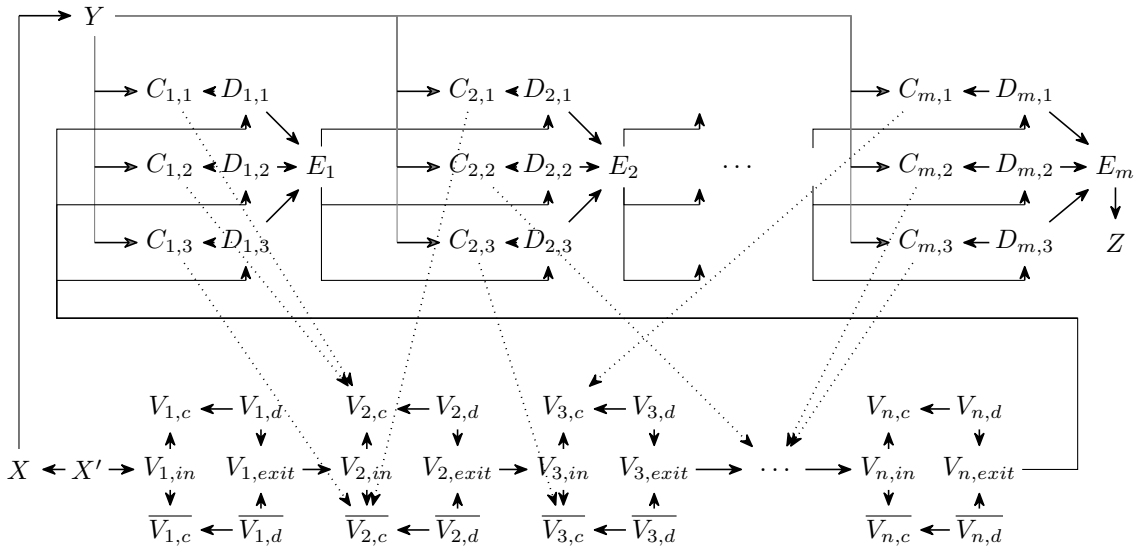


Figure 6.4: Reduction of 3-SAT to the instrumentalization problem. Each subgraph $C_{i,j}, D_{i,j}, E_i$ stands for a clause and each subgraph $V_{i,\cdot}$ for a variable in the input formula. Dotted lines connect each clause to the variables of the literals contained in the clause. In this example, the first clause would be $(x_2 \wedge x_2 \wedge \overline{x_2})$ and the second clause would contain $\overline{x_2}$ and $\overline{x_3}$.

from $x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n$. We construct the DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ for φ as follows. Throughout this proof, n and m always refer to the variables and clauses in the 3-SAT instance and never to properties of $\mathcal{G}$.

For every clause $\mathcal{C}_i$, we define nodes $\{C_{i,1}, C_{i,2}, C_{i,3}, D_{i,1}, D_{i,2}, D_{i,3}, E_i\}$ in $\mathbf{V}$, i.e., every occurring literal has two C and D nodes. Each variable of the Boolean formula is represented as six nodes $\{V_{i,in}, V_{i,c}, V_{i,d}, \overline{V_{i,c}}, \overline{V_{i,d}}, V_{i,exit}\}$ in the graph. Thus

$$\begin{aligned}\mathbf{V} \;=\; & \{X, X', Y, Z\} \cup \\ & \{C_{i,1}, C_{i,2}, C_{i,3} \mid 1 \leq i \leq m\} \cup \\ & \{D_{i,1}, D_{i,2}, D_{i,3} \mid 1 \leq i \leq m\} \cup \\ & \{E_i \mid 1 \leq i \leq m\} \cup \\ & \{V_{i,in}, V_{i,c}, V_{i,d}, \overline{V_{i,c}}, \overline{V_{i,d}}, V_{i,exit} \mid 1 \leq i \leq n\}.\end{aligned}$$

We define the edges of $\mathcal{G}$ such that, for each variable x_i, the nodes $V_{i,in}$ and $V_{i,exit}$ are connected by two paths $V_{i,in} \to V_{i,c} \leftarrow V_{i,d} \to V_{i,exit}$ and $V_{i,in} \to \overline{V_{i,c}} \leftarrow \overline{V_{i,d}} \to V_{i,exit}$; and every $V_{i,exit}$ is connected to the next $V_{i+1,in}$ such that a path from $V_{1,in}$ to $V_{n,exit}$ will contain either $V_{i,c}$ or $\overline{V_{i,c}}$ as a collider for each variable (see the bottom of Figure 6.4). When the jth literal of the ith clause is the variable x_k, there is a path $Y \to C_{i,j} \leftarrow D_{i,j} \to E_i$ and an edge $C_{i,j} \to V_{k,c}$ ($\overline{V_{k,c}}$), such that any set that opens collider $V_{k,c}$ ($\overline{V_{k,c}}$) will open a path from Y to $D_{i,j}$ that cannot be blocked by the set. D_i and E_i form a long path through all nodes

created for clauses (see the top of Figure 6.4). Thus

$$\begin{aligned}\mathbf{E} \;=\; & \{X \to Y\} \cup \\ & \{Y \to C_{i,j} \leftarrow D_{i,j} \to E_i \mid 1 \le i \le m \wedge 1 \le j \le 3\} \cup \\ & \{E_i \to D_{i+1,j} \mid 1 \le i \le m-1 \wedge 1 \le j \le 3\} \cup \\ & \{E_m \to Z\} \cup \\ & \{V_{i,in} \to \overline{V_{i,c}} \leftarrow \overline{V_{i,d}} \to V_{i,exit} \mid 1 \le i \le n\} \cup \\ & \{V_{i,in} \to V_{i,c} \leftarrow V_{i,d} \to V_{i,exit} \mid 1 \le i \le n\} \cup \\ & \{V_{i,exit} \to V_{i+1,in} \mid 1 \le i \le n-1\} \cup \\ & \{X \leftarrow X' \to V_{1,in}, V_{m,exit} \to D_{1,1}, V_{m,exit} \to D_{1,2}, V_{m,exit} \to D_{1,3}\} \cup \\ & \{C_{i,j} \to V_{k,c} \mid j\text{th literal in clause } j \text{ is } x_k\} \cup \\ & \{C_{i,j} \to \overline{V_{k,c}} \mid j\text{th literal in clause } j \text{ is } \overline{x}_k\}.\end{aligned}$$

Let $\pi_{v,i}$ be the path $V_{i,in} \to V_{i,c} \leftarrow V_{i,d} \to V_{i,exit}$ and $\overline{\pi_{v,i}}$ be $V_{i,in} \to \overline{V_{i,c}} \leftarrow \overline{V_{i,d}} \to V_{i,exit}$.

Assume Z is an instrument relative to the total effect of X on Y, so there exists a set $\mathbf{W}$ blocking all paths between Y and Z in $\mathcal{G}_{\overline{X}}$ and a path π connecting X and Z in $\mathcal{G}$ opened by $\mathbf{W}$.

π is $X \leftarrow X' \to \tau_1 \to \tau_2 \to \ldots \to \tau_n \to D_{1,i_1} \to E_1 \to \ldots \to D_{m,i_m} \to E_m \to Z$ with $\tau_i \in \{\pi_{v,i}, \overline{\pi_{v,i}}\}$ and $i_j \in \{1,2,3\}$: Each τ_i is entered at $V_{i,in}$ and can only be left at $V_{i,c} \leftarrow C_{k,j}$ ($\overline{V_{i,c}}$) or $V_{i,exit}$. If π would include $V_{i,c} \leftarrow C_{k,j}$ ($\overline{V_{i,c}} \leftarrow C_{k,j}$), the path $Y \to \pi[C_{k,j} \overset{*}{\leftarrow} Z]$ was active; thus, τ_i is left at $V_{i,exit}$. From $V_{n,exit}$, only D_{1,i_1} can be reached; and, from each D_{j,i_j}, only C_{j,i_j} or E_j can be reached. Again π cannot include $C_{j,i_j} \to$ or the path $Y \to \pi[C_{j,i_j} \overset{*}{\leftarrow} Z]$ was active. From E_j, only a previous $D_{j,\cdot}$ or a next $D_{j+1,\cdot}$ can be reached. Since a path can only visit a node once, it cannot go back to a visited $D_{\cdot}$ or $E_{\cdot}$, nor can it move through a $C_{\cdot}$.

For every D_{j,i_j} on π, the path $Y \to C_{j,i_j} \leftarrow \pi[D_{j,i_j} \overset{*}{\leftarrow} Z]$ is open if a descendant of C_{j,i_j} was in $\mathbf{W}$, so the only child $Ch(C_{j,i_j}) \in \{V_{k,c}, \overline{V_{k,c}}\}$ is not in $\mathbf{W}$ and π does not contain $\to Ch(C_{j,i_j}) \leftarrow$. Since π visits one of each $D_{j,\cdot}$, every clause j has a literal i_j that is satisfied by the following assignment:

$$x_k = \begin{cases} \text{TRUE} & \text{if } \overline{V_{k,c}} \in \mathbf{W}, \\ \text{FALSE} & \text{if } V_{k,c} \in \mathbf{W}. \end{cases}$$

The assignment is well-defined since π contains every τ_i, i.e., either $V_{i,c} \in \mathbf{W}$ or $\overline{V_{i,c}} \in \mathbf{W}$ is true.

For the other direction of the proof, assume there exists an assignment x_i satisfying formula ϕ. Let $i_j \in \{1,2,3\}$ be the first satisfied literal of clause j, $\mathbf{W} = \{\overline{V_{i,c}} \mid x_i = \text{TRUE}\} \cup \{V_{i,c} \mid x_i = \text{FALSE}\} \cup \{C_{i,j}, D_{i,j} \mid j \neq i_i\}$, and $\pi = X \leftarrow X' \to \tau_1 \to \tau_2 \to \ldots \to \tau_n \to D_{1,i_1} \to E_1 \to \ldots \to D_{m,i_m} \to E_m \to Z$, with $\tau_i = \pi_{v,i}$ if $x_i = \text{FALSE}$, or $\tau_i = \overline{\pi_{v,i}}$ else. Clearly this $\mathbf{W}$ opens π.

It remains to show that $Y \perp\!\!\!\perp Z \mid \mathbf{W}$ in $\mathcal{G}_{\overline{X}}$. The edge $X' \to X$ is removed in $\mathcal{G}_{\overline{X}}$, so any path π' from Y to Z would start with $Y \to C_{i,j}$. If $j \neq i_i$, the path $Y \to C_{i,j} \to$ is blocked at $C_{i,j}$, and the path $Y \to C_{i,j} \leftarrow D_{i,j}$ is blocked at $D_{i,j}$. If $j = i_i$, the i_ith literal satisfies clause C_i, so, if the literal is x_k, the child $V_{k,c}$ of $C_{i,j}$ is not in $\mathbf{W}$, and similarly for $\overline{x_k}$. So the path $Y \to C_{i,j} \leftarrow D_{i,j}$ is blocked at $C_{i,j}$ and the path $Y \to C_{i,j} \to V_{k,c} \leftarrow$ (resp. $Y \to C_{i,j} \to \overline{V_{k,c}} \leftarrow$) is blocked at $V_{k,c}$ (resp. $\overline{V_{k,c}}$). □

Corollary 6.20. *Determining whether, for given $X, Y, Z \in \mathbf{V}$ in a DAG $\mathcal{G}$ with $Pa(Y) \subsetneq \mathbf{O}$, node Z is a conditional instrument relative to $X \to Y$ is an NP-complete problem.*

Proof. In the proof of Theorem 6.19, the set $\mathbf{W}$ does not satisfy the conditions of a conditional IV since it contains a descendant of Y. We can transform the graph $\mathcal{G}$ of this proof to a graph $\mathcal{G}'$, such that the Z in $\mathcal{G}$ is an instrument relative to the total effect of X on Z if and only if Z is a conditional instrument relative to $X \to Y$ in $\mathcal{G}'$. Thereby Y is replaced by an unobserved node Y' and the former edge $X \to Y$ is replaced by $X \to Y \leftarrow Y'$. Then Y has no descendants, and any path connecting Y (X) with Z in $\mathcal{G}$ connects Y (X) with Z in $\mathcal{G}'$. □

This proof shows that a single unobserved parent of Y is sufficient to make the problem of testing a conditional IV NP-complete. This hardness remains when each $Y \leftarrow Y' \to C_{i,j}$ is replaced by a bidirected edge $Y \leftrightarrow C_{i,j}$, which shows that the instrumentalization problem is also hard in semi-Markovian models without unobserved nodes but with bidirected edges.

6.3.5 Testing instruments in completely unobserved graphs

A rather trivial case occurs when the set of observed nodes $\mathbf{O}$ is empty except for the examined nodes X, Y, and Z. Then $\mathbf{W} = \emptyset$ and the instrumentalization problem can be solved by just two d-separation tests. Especially the definition of an instrument, the definition of a conditional instrument, and the definition of an ancestral conditional instrument become equivalent.

Lemma 6.21. *Determining whether, for given $X, Y, Z \in \mathbf{V}$ in a DAG $\mathcal{G}$ with $\mathbf{O} = \{X, Y, Z\}$, node Z is an instrument relative to the total effect of X on Y can be solved in $O(n + m)$ time.*

6.3.6 Finding instruments in observed graphs

In this subsection, we consider the special case $Pa(Y) \subseteq \mathbf{O}$, which permits efficient, linear-time algorithms for (ancestral) conditional instrumental variables.

Above we have described the algorithm WITNESS-ANCESTRAL-INSTRUMENT that is sound and complete to test whether a variable Z is an ancestral instrumental variable. The functionality of this algorithm was based on finding a set $\mathbf{W} :=$ FIND-NEAREST-SEPARATOR$(\mathcal{G}_c, Y, Z)$ that d-separates Y and Z without blocking a path between Y and X, no matter which node is X, as long as X is not a descendant of Y. In the special case $Pa(Y) \subseteq \mathbf{O}$, this set $\mathbf{W}$ is always $Pa(Y) \setminus X$, which is easy to see by examining the evaluation of algorithm FIND-NEAREST-SEPARATOR: The algorithm starts at the parents of Y and children that are in $An(Y, Z)$ in $\mathcal{G}_c$. However, no children are such ancestors; hence, the algorithm will only visit the parents, and when they are all observed, return them and not continue to other nodes.

The set $\mathbf{W} = Pa(Y) \setminus X$ does not depend on Z, which greatly simplifies algorithm FIND-CONDITIONAL-INSTRUMENT. Rather than recalculating $\mathbf{W}$ for every candidate node Z, we can calculate $\mathbf{W}$ once and then immediately find all ancestral IVs as nodes not d-separated from X. This is expressed by the following lemma:

Lemma 6.22. *For given X and Y in a DAG $\mathcal{G}$ with $Pa(Y) \subseteq \mathbf{O}$, a node $Z \in \mathbf{O} \setminus \{X, Y\}$ is an ancestral instrument relative to $X \to Y$ if and only if*

(i) *Z is not a descendant of Y,*

(ii) *Z is not in $Pa(Y)$, and*

(iii) *$Pa(Y) \setminus X$ does not d-separate X and Z in $\mathcal{G}_c$.*

Proof. This follows from the soundness and completeness of algorithm WITNESS-ANCESTRAL-INSTRUMENT and the remarks above. Note that $Pa(Y) \setminus X$ d-separates Y and Z, if $Z \notin Pa(Y)$ and Z is not a descendant of Y: If a path π from Y to Z in $\mathcal{G}_c$ starts with $Y \leftarrow$, it is blocked by $Pa(Y) \setminus X$. If it starts with $Y \rightarrow$, it is either a directed path to Z or contains a (first) collider C. The former case cannot occur, because $Z \notin De(Y)$. In the latter case, the collider can not be opened by a parent of Y. □

This leads to the following algorithm to find all ancestral instruments in linear time:

function ENUMERATE-ANCESTRAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y$)
  Assertion: $Pa(Y) \subseteq \mathbf{O}$
  $\mathcal{G}_c := (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E} \setminus (X \rightarrow Y))$
  $\mathbf{W} := Pa(Y) \setminus X$
  $accept :=$ **function**$(e, U, f, V)\{$
    return (U is a collider on $eUf \wedge U \in \mathbf{W}$)
      $\vee$ (U is a non-collider on $eUf \wedge U \notin \mathbf{W}$)
  $\}$
  return REACHABLE($\mathcal{G}_c, X, accept$) $\setminus$ ($De(Y) \cup Pa(Y) \cup X$)

The call to REACHABLE finds all nodes not d-separated from X by $\mathbf{W}$ as described in the analysis of TESTSEP.

When we can find all ancestral instrumental variables, we can also find a single one in the same running time which improves the general algorithm FIND-ANCESTRAL-INSTRUMENT for observed graphs:

function FIND-ANCESTRAL-INSTRUMENT-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y$)
  Assertion: $Pa(Y) \subseteq \mathbf{O}$
  $\mathbf{Z} :=$ ENUMERATE-ANCESTRAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y$)
  return random node of $\mathbf{Z}$

To summarize:

Theorem 6.23. *For given X and Y in a DAG $\mathcal{G}$ with $Pa(Y) \subseteq \mathbf{O}$, the algorithm* ENUMERATE-ANCESTRAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS *returns all ancestral instruments relative to $X \rightarrow Y$, and algorithm* FIND-ANCESTRAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS *finds one of them.*

The running time of the algorithms is $\mathcal{O}(n + m)$.

For non-ancestral conditional instruments, it is not possible to find one common set $\mathbf{W}$ for all IVs, since the nodes required to open a path from X to one IV might block the path to another IV. For example, in the DAG $Y \leftarrow X \rightarrow Z_1 \overset{\frown}{\leftarrow Z_2} Z_3$ opening the path from X to Z_2 using $\mathbf{W} = \{Z_1\}$ will block the path to Z_3 and using $\mathbf{W} = \{Z_3\}$ leads to a path to Z_3 with an undefined state. However, the arguments in the proof of Lemma 6.22 still work, i.e., the set $Pa(Y) \setminus X$ is sufficient to block any path between Y and any conditional IV Z, and adding a non-descendant of Y to $\mathbf{W}$ will not open a path between Y and Z. Hence, we can also

find all conditional IVs in linear-time by performing a reachability search for non-d-separated nodes and assuming that every collider that might be opened by any set is open. This yields the following lemma:

Lemma 6.24. *For given X and Y in a DAG $\mathcal{G}$ with $Pa(Y) \subseteq \mathbf{O}$, a node $Z \in \mathbf{O} \setminus \{X, Y\}$ is a conditional instrument relative to $X \to Y$ if and only if*

(i) *Z is not a descendant of Y,*

(ii) *Z is not a parent of Y, and*

(iii) *there exists a walk from X to Z in $\mathcal{G}_c$, such that every non-collider is not in $Pa(Y) \setminus X$ and every collider is in $\mathbf{O} \setminus De(Y)$.*

Proof. Let w be the shortest walk between X and Z as in condition (iii). Let $\mathbf{W} = (Pa(Y) \setminus X) \cup \{\text{colliders on } w\}$. This set satisfies the three conditions of Definition 6.2: First, the walk w is open given $\mathbf{W}$, since w contains no non-collider in $Pa(Y) \setminus X$ and $\mathbf{W}$ contains all colliders, unless one collider C also occurs as a non-collider on w, but then removing the subwalk between the first and last occurrence of C from w would give a shorter walk satisfying condition (iii), contradicting the choice of w.

Second, $\mathbf{W}$ as chosen by the algorithm blocks all paths between Y and Z in $\mathcal{G}_c$. Assume there is a path π' not blocked. If π' starts with $Y \leftarrow$, it is blocked because $Pa(Y) \setminus X \subseteq \mathbf{W}$. If π' starts with $Y \rightarrow$, it is either a directed path to Z or contains a (first) collider C. The former case cannot occur, because $Z \notin De(Y)$. In the latter case, the collider can only be opened by a descendant of C in $\mathbf{W}$, which would be a forbidden descendant of Y.

Third, $\mathbf{W}$ only contains parents of Y as well as colliders that are no descendants of Y and in $\mathbf{O}$.

In the other direction, if Z is a conditional instrument relative to $X \to Y$, there exists a set $\mathbf{W}'$ satisfying the conditions of Definition 6.2 and a walk w' between X and Z opened by $\mathbf{W}'$ with all colliders in $\mathbf{W}'$. Z is not a descendant of Y, because the path $Y \dashrightarrow Z$ could only be blocked by a descendant of Y, and thus is not blocked by $\mathbf{W}'$. Z is not a parent of Y and w' does not contain a non-collider $P \in Pa(Y) \setminus X$, because the walk $Y \leftarrow w'[P \overset{*}{\sim} Z]$ would d-connect Y and Z. □

The lemma could be stated with paths, but we use walks, so the relevant nodes can be easily found by a call to REACHABLE.

Using algorithm REACHABLE, we can define a linear-time enumeration algorithm to find all conditional instrumental variables similarly to the above algorithm for ancestral IVs:

function ENUMERATE-CONDITIONAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y$)
  Assertion: $Pa(Y) \subseteq \mathbf{O}$
  $\mathcal{G}_c := (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E} \setminus (X \to Y))$
  $\mathbf{W}_n := Pa(Y) \setminus X$
  $\mathbf{W}_c := \mathbf{O} \setminus De(Y)$
  $accept :=$ **function**$(e, U, f, V)\{$
    return (U is a collider on $eUf \wedge U \in \mathbf{W}_c$)
      $\vee$ (U is a non-collider on $eUf \wedge U \notin \mathbf{W}_n$)
  $\}$
  return REACHABLE$(\mathcal{G}_c, X, accept) \setminus (De(Y) \cup Pa(Y))$

This enumeration algorithm can then be applied to find and test conditional IVs:

function FIND-CONDITIONAL-INSTRUMENT-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y$)
Assertion: $Pa(Y) \subseteq \mathbf{O}$
$\mathbf{Z} :=$ ENUMERATE-CONDITIONAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y$)
return random node of $\mathbf{Z}$

function TEST-CONDITIONAL-INSTRUMENT-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y, Z$)
Assertion: $Pa(Y) \subseteq \mathbf{O}$
$\mathbf{Z} :=$ ENUMERATE-CONDITIONAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS($\mathcal{G}, X, Y$)
return $Z \in \mathbf{Z}$

To summarize:

Theorem 6.25. *For given X and Y in a DAG $\mathcal{G}$ with $Pa(Y) \subseteq \mathbf{O}$, the algorithm* ENUMERATE-CONDITIONAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS *returns all conditional instruments relative to $X \to Y$, and the algorithm* FIND-CONDITIONAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS *finds one of them. For an additionally given node Z, the algorithm* TEST-CONDITIONAL-INSTRUMENTS-WITH-OBSERVABLE-PARENTS *returns* TRUE *if and only if Z is a conditional IV relative to $X \to Y$.*

The running time of the algorithms is $\mathcal{O}(n + m)$.

Therefore, there is a fundamental difference between the complexity of testing whether a given variable Z is a conditional IV in the case of no unobserved parents and a single unobserved parent P. The former case is solvable in linear-time and the latter case is NP-complete as we know from Corollary 6.20. The hardness seems to result from paths that are opened by descendants of P and then are complicated to block again.

The same hardness occurs when descendants of Y are allowed in $\mathbf{W}$ like for instrumental variables relative to the total effect. As Theorem 6.19 shows testing an IV relative to the total effect is NP-complete even if all nodes are observed. Hence, we cannot give linear-time algorithms to find IVs relative to the total effect in the observed case, not even to find ancestral IVs relative to the total effect.

6.3.7 Enumerating Instrumental Variables

The problems of enumerating all instrumental variables and testing whether a certain variable Z is an instrumental variable can be reduced to each other for every kind of IV. Once an enumeration algorithm has returned a list $\mathbf{Z}$ of all valid IVs, one can test whether a variable Z is an IV by checking the condition $Z \in \mathbf{Z}$, so enumeration is at least as hard as testing. On the other hand, there are at most n IVs, so the runtime of an enumeration algorithm is at most n times the runtime required to test one variable.

Hence, we know that enumerating all IVs is NP-complete for instruments relative to the total effect in graphs with and without unobserved nodes, and for conditional instruments in graphs with one unobserved parent of Y.

Enumerating all ancestral instruments is, however, always possible in $O(n(n + m))$, i.e., it is not harder than finding one ancestral IV.

$$
\begin{aligned}
Z_1 &= \varepsilon_1 \\
Z_2 &= \varepsilon_2 \\
X_1 &= a_1 Z_1 + a_2 Z_2 + \varepsilon_3 \\
X_2 &= b_1 Z_1 + b_2 Z_2 + \varepsilon_4 \\
Y &= c_1 X_1 + c_2 X_2 + \varepsilon_5 \\
\mathrm{Cov}(\varepsilon_3, \varepsilon_5) &= \alpha_1 \neq 0 \\
\mathrm{Cov}(\varepsilon_4, \varepsilon_5) &= \alpha_2 \neq 0
\end{aligned}
$$

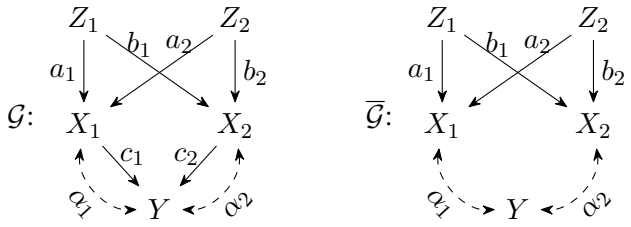


Figure 6.5: A linear model, its causal graph $\mathcal{G}$ [BP02; Bri04], and $\overline{\mathcal{G}}$ for $\mathbf{X} = \{X_1, X_2\}$.

6.4 Instrumental Sets

Often one cannot find a (conditional) IV, because there is no variable d-separated from Y in $\mathcal{G}_c$. In such cases, it can help to remove more edges from $\mathcal{G}$ than just $X \to Y$ before searching for d-separated variables. Now one cannot remove arbitrary edges at random, but under specific conditions, one can remove a set of edges $\mathbf{X} \to Y$.

[Bri04; Bri10] and [BP02] have studied these conditions and introduced the concept of *instrumental sets*. Given variables Y and $\mathbf{X} = \{X_1, \ldots, X_k\}$, an instrumental set $\mathbf{Z}$ identifies the parameters $c_1, \ldots, c_k$ of the edges $X_1 \overset{c_1}{\to} Y, \ldots, X_k \overset{c_k}{\to} Y$ simultaneously. For this, the instrumental set $\mathbf{Z}$ needs to be d-separated from Y in the graph $\overline{\mathcal{G}} = \mathcal{G} \setminus (\mathbf{X} \to Y)$. The formal definition of instrumental sets is given in the following subsections of Section 6.4 with a three-layer hierarchy of simple and more general instrumental sets.

instrumental sets

graph $\overline{\mathcal{G}}$

Until our work, it was not clear whether such instrumental sets can be found efficiently. Moreover, no results have been known demonstrating that searching for instrumental sets is hard either. In Section 6.5, we analyze which instrumental sets are hard to find and which can be found in polynomial time.

6.4.1 Simple Instrumental Sets

First we need to introduce the concept of incompatible paths. Let $(X_1, \ldots, X_k)$ and $(Z_1, \ldots, Z_k)$ be tuples of k variables, and $\pi_1, \ldots, \pi_k$ be unblocked paths connecting variable X_i to Z_i. The paths $\pi_1, \ldots, \pi_k$ are *incompatible* if for all i and j with $1 \leq i < j \leq k$, variable Z_j does not appear in path π_i; and, if paths π_i and π_j have a common variable V, then both $\pi_i[V \sim X_i]$ and $\pi_j[Z_j \sim V]$ point to V. This definition implies that it is not possible to rearrange the edges of incompatible paths to create new paths between the same nodes.

incompatible

[Bri10] defines simple instrumental sets as:

Definition 6.26 ([Bri10])**.** *The set $\mathbf{Z}$ is said to be a* simple instrumental set *relative to $\mathbf{X}$ and Y in $\mathcal{G}$ if, for a permutation $Z_1, \ldots, Z_k$ of $\mathbf{Z}$ and a permutation $X_1, \ldots, X_k$ of $\mathbf{X}$, it is true:*

simple instrumental set

(a) There exist unblocked paths $\pi_1, \ldots, \pi_k$ connecting $Z_1, \ldots, Z_k$ to $X_1, \ldots, X_k$, resp., such that the paths are incompatible, and

(b) the variables Z_i are d-separated from Y in $\overline{\mathcal{G}}$.

It is easy to see that variables Z_1 and Z_2 of the model in Figure 6.5 satisfy the conditions of Definition 6.26 relative to $\{X_1, X_2\}$ and Y with the paths $\pi_1 = Z_1 \to X_1$ and $\pi_2 = Z_2 \to X_2$.

No conditional IV relative to $X_i \to Y$ exists in Figure 6.5, because blocking one of the paths through $\mathbf{X} \to Y$ opens $\mathbf{X} \leftrightarrow Y$, so Z_1 (or Z_2) and Y are not d-separable in $\mathcal{G}_c$.

The causal effects $c_1, \ldots, c_k$ are in general identified by a simple instrumental set as follows: Since Z_i and Y are d-separated in $\overline{\mathcal{G}}$, every path between Z_i and Y has the form $Z_i \overset{*}{\leftsquigarrow} X_j \to Y$. According to Wright's method of path coefficients, this means the covariance $\rho_{Z_i,Y}$ is the sum of the products of the path coefficients of all those paths. The first parts $Z_i \overset{*}{\leftsquigarrow} X_j$ correspond to all paths between Z_i and X_j, so the sum of these coefficients becomes the covariance ρ_{Z_i,X_j}. The coefficient of edge $X_j \to Y$ is just c_j, so $\rho_{Z_i,Y} = \sum_j \rho_{Z_i,X_j} c_j$. With the abbreviation $a_{ij} = \rho_{Z_i,X_j}$, we have $\rho_{Z_i,Y} = \sum_j a_{ij} c_j$ [Bri10].

Since this holds for each variable of the instrumental set, it yields a system of linear equations:

$$\begin{aligned}
\rho_{Z_1,Y} &= a_{11}c_1 + \ldots + a_{1k}c_k \\
&\ldots \\
\rho_{Z_k,Y} &= a_{k1}c_1 + \ldots + a_{kk}c_k.
\end{aligned}$$

This system of equations can be solved for $c_1, \ldots, c_k$. [Bri10] argues that a unique solution exists generically because the determinant of the matrix of coefficients $(a_{ij})_{ij}$ is not 0. One of the terms in the determinant is the product of the path coefficients π_i, and since those paths are incompatible, there exist no other paths that could cancel this term. This identification is generical since the determinant can still be zero for some coefficients despite being a non-zero polynomial in the coefficients, i.e., when the coefficients are the roots of the determinant. But a set of roots has Lebesgue measure of zero.

For example, the model of Figure 6.5 yields these equations

$$\begin{aligned}
\rho_{Z_1,Y} &= a_{11}c_1 + a_{12}c_2 \\
\rho_{Z_2,Y} &= a_{21}c_1 + a_{22}c_2,
\end{aligned}$$

which can be solved using some linear algebra as

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}^{-1} \begin{pmatrix} \rho_{Z_1,Y} \\ \rho_{Z_2,Y} \end{pmatrix}.$$

6.4.2 Generalized Instrumental Sets

Like the concept of conditional instrumental variables generalizes instrumental variables, Brito and Pearl have generalized the simple instrumental sets through the use of conditioning.

generalized instrumental set

Definition 6.27 ([BP02; Bri04])**.** *The set* $\mathbf{Z}$ *is said to be a* generalized instrumental set *relative to* $\mathbf{X}$ *and* Y *in* $\mathcal{G}$ *if, for a permutation* $Z_1, \ldots, Z_k$ *of* $\mathbf{Z}$ *and a permutation* $X_1, \ldots, X_k$ *of* $\mathbf{X}$*, there exist triples* $(Z_1, \mathbf{W}_1, \pi_1), \ldots, (Z_k, \mathbf{W}_k, \pi_k)$*, with* $\mathbf{W}_i \subseteq \mathbf{V}$ *such that:*

(a) Every π_i *is an unblocked path between* Z_i *and* Y *including edge* $X_i \to Y$*, and for* $i = 1, \ldots, k$*,* Z_i *and the elements of* $\mathbf{W}_i$ *are non-descendents of* Y*,*

(b) every set $\mathbf{W}_i$ *d-separates* Z_i *from* Y *in* $\overline{\mathcal{G}}$*; but* $\mathbf{W}_i$ *does not block path* π_i*, and*

(c) paths $\pi_1, \ldots, \pi_k$ *are incompatible.*

Brito and Pearl [Bri04; BP02] show $\rho_{Z_i,Y,\mathbf{W}_i} = \frac{\phi_i(Z_i,Y,\mathbf{W}_i)}{\psi_i(Z_i,\mathbf{W}_i)\psi_i(Y,\mathbf{W}_i)}$, where ϕ_i is a linear function on the covariances $\rho_{V,Y}$ for $V \in Z_i \cup \mathbf{W}_i$, and ψ_i is a function on the covariances of the variables given as arguments. They further give a recursive definition of these functions. Since the functions are defined in terms of covariances, they can be evaluated from observed data. Covariance $\rho_{Z_i,Y,\mathbf{W}_i}$ is zero if and only if $\phi_i(Z_i, Y, \mathbf{W}_i)$ is zero, so one could think of ϕ_i as the actual covariance between Z_i and Y and of ψ_i as a normalization factor.

Because $\phi_i(Z_i, Y, \mathbf{W}_i)$ is linear in the covariance of $Z_i \cup \mathbf{W}_i$ with Y, it can be written as $\phi_i(Z_i, Y, \mathbf{W}_i) = b_{i_0}\rho_{Z_iY} + \sum_{j:W_j \in \mathbf{W}_i} b_{i_j}\rho_{W_jY}$ for some values b_{i_j} which can be obtained from the recursive definition.

Since all paths between $V_j \in Z_i \cup \mathbf{W}_i$ and Y contain an edge into Y as a factor, the covariances can also be written as a linear combination $\rho_{V_jY} = \sum_l \alpha_{i_jl}c_l$. Only the coefficients c_l of the edges $X_i \to Y$ occur as non-zero terms in the linear combination (i.e., no coefficient of any edge $(Pa(Y) \setminus \mathbf{X}) \to Y$ occurs) because Z_i and Y are d-separated in $\overline{\mathcal{G}}$.

With the abbreviation $q_{ij} = \sum_k b_{i_k}a_{i_k,j}$, it follows that $\phi_i(Z_i, Y, \mathbf{W}_i) = \sum_j q_{ij}c_j$, which is a linear equation system with non-zero determinant that can be solved for c_j.

6.4.3 Simple Conditional Instrumental Sets

We introduce a natural intermediate level between the simple and the generalized instrumental sets by restricting Definition 6.27 such that the sets $\mathbf{W}_1 = \mathbf{W}_2 = \ldots = \mathbf{W}_k$ have to be equal.

simple conditional instrumental set

Definition 6.28. *The set* $\mathbf{Z}$ *is said to be a* simple conditional instrumental set *relative to* $\mathbf{X}$ *and* Y *in* $\mathcal{G}$ *if, for a permutation* $Z_1, \ldots, Z_k$ *of* $\mathbf{Z}$ *and a permutation* $X_1, \ldots, X_k$ *of* $\mathbf{X}$*, there exist a set* $\mathbf{W} \subseteq \mathbf{V}$ *and pairs* $(Z_1, \pi_1), \ldots, (Z_k, \pi_k)$ *such that:*

(a) Every π_i *is an unblocked path between* Z_i *and* Y *including edge* $X_i \to Y$*, and all* Z_i *and all elements of* $\mathbf{W}$ *are non-descendents of* Y*,*

(b) $\mathbf{W}$ *d-separates every* Z_i *from* Y *in* $\overline{\mathcal{G}}$*; but* $\mathbf{W}$ *does not block any path* π_i*, and*

(c) paths $\pi_1, \ldots, \pi_k$ *are incompatible.*

Since a simple conditional instrument set is also a generalized instrumental set, it can be used to compute the causal effect of $X_i \to Y$ in the same way.

6.4.4 Singleton Sets as Instrumental Variables

In the restricted case $k = 1$, the instrumental sets only contain a single instrumental variable, and the definitions of instrumental sets become similar to the definitions of single instrumental variables of Section 6.2. Specifically, the graph $\overline{\mathcal{G}}$ becomes equal to $\mathcal{G}_c$ and the restriction to incompatible paths can be ignored. So Definition 6.26 of simple instrumental sets is not just similar, but identical to Definition 6.1 of instrumental variables if $k = 1$.

It is also obvious that Definition 6.27 of generalized instrumental sets becomes equivalent to Definition 6.28 of simple conditional instrumental sets for singletons. Less obvious is that both definitions become equivalent to Definition 6.8 of an active conditional instrumental variable:

Lemma 6.29. *A set* $\{Z\}$ *is a generalized instrumental set relative to* $\{X\}$ *and* Y *if and only if* Z *is an active conditional instrument relative to* $X \to Y$*.*

	Complexity	Algorithm/Reference
Simple Instrumental Sets (Definition 6.26)		
Testing	$\mathcal{O}(n+km)$	TEST-SIMPLE-IVS
Finding	$\mathcal{O}(n+km)$	FIND-SIMPLE-IVS
Simple Conditional Instrumental Sets (Definition 6.28)		
Testing	$\mathcal{O}(n+km)$	TEST-SIMPLE-COND-IVS
Finding	$\mathcal{O}(n^{k+3})$	FIND-SIMPLE-COND-IVS
	NP-complete	Kumor et al. [KCB19]
Generalized Instrumental Set (Definition 6.27)		
Testing	$\mathcal{O}(k(k!)^2 n^{3k+1})$	TEST-GENERAL-IVS
	NP-complete	Theorem 6.36
Finding	$\mathcal{O}(k(k!)^2 n^{4k+1})$	FIND-GENERAL-IVS

Table 6.3: This table summarizes our algorithms and the complexities to test whether a given set $\mathbf{Z}$ is an instrumental set for given $\mathbf{X}, Y$, and to find an instrumental set $\mathbf{Z}$ for given $\mathbf{X}, Y$ for the three layers of instrumental sets. $k = |\mathbf{X}|$ is the size of $\mathbf{X}$. The NP-complete problems can be solved with a runtime exponential in k, which is polynomial time if k is bounded by a constant.

Proof. Let the triple $(Z, \mathbf{W}, \pi)$ satisfy the conditions of Definition 6.27. The path π is of the form $\pi' \to Y$, where π' is a path between Z and X. It is easy to see that $\mathbf{W}$ and π' satisfy the conditions of Definition 6.8.

Below we show that the opposite implication is true, too. So, let the set $\mathbf{W}$ and path π satisfy the conditions of Definition 6.8. Since $\mathbf{W}$ does not block the path π in $\mathcal{G}_c$, we have $X \notin \mathbf{W}$. We show that the triple $(Z, \mathbf{W}, \pi \to Y)$ satisfies the conditions of a generalized instrumental set.

Set $\mathbf{W}$ consists of non-descendants of Y due to condition 1 of Definition 6.8. Node Z is a non-descendant of Y, otherwise the directed path between Y and Z could not be blocked by $\mathbf{W}$ using only non-descendants. The path π ends with X and does not contain Y, since $\mathbf{W}$ does not block π in $\mathcal{G}_c$, but d-separates Z and Y in $\mathcal{G}_c$ and we have that $Y \notin \mathbf{W}$. So, $\pi \to Y$ is an unblocked path between Z and Y including the edge $X \to Y$.

$\mathbf{W}$ d-separates Z from Y in $\mathcal{G}_c = \overline{\mathcal{G}}$ due to condition 2 of Definition 6.8. Set $\mathbf{W}$ does not block the path $\pi \to Y$, since it does not block π in $\mathcal{G}_c$ due to 4. Finally, note that the last condition of Definition 6.27 is always true for a singleton set. □

6.5 Algorithmics of Instrumental Sets

In this section, we analyze the complexity of finding and testing instrumental sets, i.e., for given $\mathbf{X}$ and Y, find an instrumental set $\mathbf{Z}$ or test whether an additionally given $\mathbf{Z}$ is actually an instrumental set. The results are shown in Table 6.3.

We begin in Subsection 6.5.1 with an efficient algorithm to find incompatible paths. This algorithm combined with nearest separators yields algorithms to test and find simple conditional instrumental sets in Section 6.5.2. In Section 6.4.1, we modify those algorithms to find simple instrumental sets. In Section 6.5.4, we prove that it is an NP-complete problem to test whether a given set is a generalized instrumental set. In Section 6.5.5, we give an (inefficient) algorithm to find incompatible paths, which can be used to find and test generalized instrumental sets.

6.5.1 Finding Incompatible Paths via Flows

A network flow is the standard algorithm to find disjoint paths. However, flow algorithms cannot be applied directly to a causal graph, since these algorithms are not designed to handle d-separation. In Section 3.3.8, we have transformed the causal graph to an undirected moral graph before applying the flow algorithm. But the moral graph cannot be used to find active paths – without colliders – for instrumental sets, because paths in the moral graph can skip colliders. We will thus introduce another transformation of a causal DAG $\mathcal{G}$ to a DAG $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$ in which active d-paths from $\mathbf{Z}$ to $\mathbf{X}$ become directed paths from $\mathbf{Z}$ to $\mathbf{X}$.

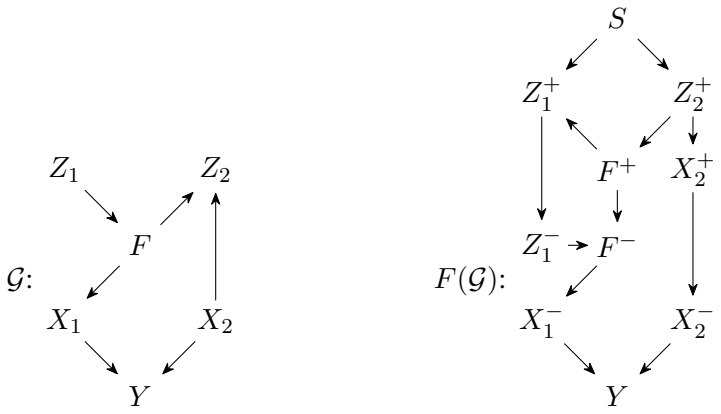


Figure 6.6: A DAG $\mathcal{G}$ and its flow graph $F(\mathcal{G}, \{Z_1, Z_2\}, Y, \emptyset)$.

Nodes of $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$ consist of two sets, which we denote as $\mathbf{V}^+$ and $\mathbf{V}^-$, as well as Y and a new start node S (the node Y as a child of all $\mathbf{X}$ does not need to exist in $\mathcal{G}$, but in the case of instrumental sets, it does exist). The first set, also called $(+)$-layer, is the induced subgraph of all ancestors of $\mathbf{Z}$ with inverted edges. The second set, called $(-)$-layer, is the induced subgraph of all ancestors of $\mathbf{X}$[1]. If the same node exists in both layers, the two versions of it are distinct but connected by an edge from the $(+)$-layer to the $(-)$-layer. Thus, a d-path $Z \overset{*}{\leftarrow} F \overset{*}{\rightarrow} X$ becomes a directed path $Z^+ \overset{*}{\rightarrow} F^+ \rightarrow F^- \overset{*}{\rightarrow} X^-$, and any directed path in F corresponds to a d-path in $\mathcal{G}$ that is active given $\mathbf{W}$ iff it contains no node of $\mathbf{W}$.

A similar construction was used by Brito for constructing a so-called *auxiliary set* of variables for model identification [Bri04].

The flow graph $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W}) = (\mathbf{V}_F, \mathbf{E}_F)$ with respect to $\mathbf{Z}$, $\mathbf{X}$, $\mathbf{W}$ is formally defined as follows. Let $\mathbf{V}^+ = \{V^+ \mid V \in An(\mathbf{Z})\}$ and $\mathbf{V}^- = \{V^- | V \in An(\mathbf{X})\}$. Then the vertices and edges are:

$$
\begin{aligned}
\mathbf{V}_F = \;& \mathbf{V}^+ \cup \mathbf{V}^- \cup \{S, Y\} \\
\mathbf{E}_F = \;& \{V^+ \rightarrow W^+ \mid V, W \in \mathbf{V}^+; V \leftarrow W \in \mathbf{E}\} \\
& \cup \{V^+ \rightarrow V^- \mid V \in \mathbf{V}^+ \cap \mathbf{V}^-\} \\
& \cup \{V^- \rightarrow W^- \mid V, W \in \mathbf{V}^-; V \rightarrow W \in \mathbf{E}\} \\
& \cup \{S \rightarrow Z^+ \mid Z \in \mathbf{Z}\} \\
& \cup \{X^- \rightarrow Y \mid X^- \in \mathbf{X}\}.
\end{aligned}
$$

We assign capacities to the nodes as follows: S and Y have infinite capacity, the $\pm$ nodes

[1] Signs $+$ and $-$ can be seen as the arrow head of the edge leaving a node of this layer.

resulting from nodes in $\mathbf{W}$ have zero capacity, and all other nodes have unit capacity. All edges have infinite capacities.

Lemma 6.30. *Given $\mathbf{Z}, \mathbf{X}, \mathbf{W}$, there exists an $|\mathbf{X}|$-flow from $\mathbf{Z}$ to $\mathbf{X}$ in $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$ if and only if there exist $|\mathbf{X}|$ incompatible paths from $\mathbf{Z}$ to $\mathbf{X}$ in $\mathcal{G}$ that are active given $\emptyset$ and $\mathbf{W}$.*

Proof. ($\Leftarrow$): Let $k = |\mathbf{X}|$ and $\pi_1, \ldots, \pi_k$ be incompatible paths from $\mathbf{Z}$ to $\mathbf{X}$ in $\mathcal{G}$. Each π_i can be split into two (possibly empty) arcs π_i^+ and π_i^-. The first is directed towards $\mathbf{Z}$ and the second is directed towards Y. π_i^+ corresponds to a directed path of plus nodes in $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$ and π_i^- to a directed path of minus nodes. A possibly existing fork F_i on π_i corresponds to a subpath $F_i^+ \rightarrow F_i^-$. So, for each path π_i, there exists a path π_i' in $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$ because π_i^+ only contains ancestors of $\mathbf{Z}$ and π_i^- only ancestors of $\mathbf{X}$.

These paths π_i' do not intersect. If, for $i < j$, there were two paths π_i' and π_j' containing a common node V^+, the corresponding paths π_i and π_j in $\mathcal{G}$ would have the form $Z_i \overset{+}{\leftarrow} V$, $Z_j \overset{+}{\leftarrow} V$, or $V \in \{Z_i, Z_j\}$, but $\pi_j[Z_j \sim V]$ would need to point towards V in incompatible paths. The same argument holds for common nodes in V^-.

Since each of these paths π_i' can carry a unit flow, an $|\mathbf{X}|$-flow exists.

($\Rightarrow$): The $|\mathbf{X}|$-flow can be represented as $k = |\mathbf{X}|$ paths $\pi_1', \ldots, \pi_k'$ from S to Y that are disjoint (except for the end nodes) and not blocked by $\mathbf{W}$. We can assume that the paths are chosen such that their total length is minimal[2]. We can project these paths to d-connected paths $\pi_1, \ldots, \pi_k$ in $\mathcal{G}$, by dropping the $\pm$ markers from nodes, the first edge and replacing $\rightarrow V^+ \rightarrow V^- \rightarrow$ with a single fork $\leftarrow V \rightarrow$.

If path π_i intersects path π_j at a non-end node, there is a common node V, which corresponds to V^+ and V^- in π_i' and π_j'. We will order the paths such that V^- is in π_j' and $V \rightarrow$ occurs in π_j. So each such intersection gives a partial ordering constraint $\pi_i \prec \pi_j$.

If these partial orderings cannot be combined to a valid total ordering, there is a cycle of $k' \geq 2$ paths $\pi_{i_1} \prec \ldots \prec \pi_{i_{k'}} \prec \pi_{i_1}$ intersecting at nodes $V_{i_1}, \ldots, V_{i_{k'}}$ with $V_{i_j} \leftarrow$ in π_{i_j} and $V_{i_j} \rightarrow$ in $\pi_{i_{j+1}}$ (we set $i_{k'+1} = i_1$). Since the construction of $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$ can neither lead to colliders in the projected paths nor intersections at forks because a path containing a fork contains the $+$ and $-$ variant of the node, the paths $\pi_{i_{j+1}}$ have the form $Z_{i_{j+1}} \overset{*}{\leftarrow} V_{i_{j+1}} \leftarrow\!\!\overset{*}{\sim}\!\!\rightarrow V_{i_j} \overset{*}{\rightarrow} X_{i_{j+1}}$. So from every node V_{i_j}, there exist directed paths to nodes in $\mathbf{X}$ and $\mathbf{Z}$. Thus, we can replace each path π_{i_j} with a path $Z_{i_j} \overset{*}{\leftarrow} V_{i_j} \overset{*}{\rightarrow} X_{i_{j+1}}$. These new paths are shorter than the original paths, violating the initial assumption (this also holds if $V_{i_j} = Z_{i_j}$, $V_{i_j} = X_{i_j}$, or the new paths intersect themselves).

If two paths π_i and π_j with $i < j$ intersect in a node V, π_i' contains V^+, and π_j' contains V^-. Since π_j' starts in Z_j^+, V cannot be Z_j, and Z_j does not occur in π_i. Because V cannot be a fork, π_i contains $V \leftarrow$, and π_j contains $\rightarrow V \rightarrow$, which makes the paths incompatible. □

6.5.2 Testing and Finding Simple Conditional Instruments

Testing Simple Conditional Instruments

Let Y be a node, $\mathbf{X} \subseteq Pa(Y)$, and $\mathbf{Z}$ a set in a DAG $\mathcal{G}$. In order to test whether $\mathbf{Z}$ is a simple conditional instrumental set, we need to simultaneously find (or verify the existence of) a set $\mathbf{W}$, permutations of $\mathbf{X}$ and $\mathbf{Z}$, and incompatible paths connecting Z_i and X_i satisfying Definition 6.28. It turns out we can again search for a nearest separator between Y and $\mathbf{Z}$ in $\overline{\mathcal{G}}$ to find $\mathbf{W}$ because the nearest separator will not block any unnecessary paths.

[2]Such paths can be found efficiently by a min-cost-max-flow algorithm.

Lemma 6.31. *Let X, Y, Z be nodes in a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, $\mathbf{X} \subseteq Pa(Y)$ a set with $X \in \mathbf{X}$, $\pi : Z \overset{+}{\sim} X \to Y$ an active path in $\mathcal{G}$, $\overline{\mathcal{G}} = (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \to Y))$ a subgraph, and $\mathbf{W}$ a nearest separator relative to (Y, Z) in $\overline{\mathcal{G}}$.*

If there exists a set $\mathbf{W}'$ that d-separates Y and Z in $\overline{\mathcal{G}}$ and does not contain a node of π, then $\mathbf{W}$ also does not contain a node of π.

Proof. Since Y can only occur once in π, π contains only one edge $X \to Y$ and the subpath $\pi[Z \overset{*}{\sim} X]$ also exists in $\overline{\mathcal{G}}$.

Due to Theorem 6.11, $\mathbf{W} \cup \mathbf{W}'$ does not block $\pi[Z \overset{*}{\sim} X]$ and does not contain X, so $\mathbf{W}'$ contains no non-collider of π. As an active path π contains no colliders. □

Since $\mathbf{Z}$ is a set and not a single node, we add a helper node Z' and connect it as $Z' \leftarrow Z$ to all $Z \in \mathbf{Z}$ before calling algorithm FIND-NEAREST-SEPARATOR on (Y, Z').

Knowing $\mathbf{W}$, it is then possible to find the paths and permutations as a max-flow. This idea yields the following algorithm:

function TEST-SIMPLE-COND-IVS($\mathcal{G}$, $\mathbf{X}$, Y, $\mathbf{Z}$)
  $\mathcal{G}' := (\mathbf{V} \cup Z', (\mathbf{E} \setminus (\mathbf{X} \to Y)) \cup (Z' \leftarrow \mathbf{Z}))$
  $\mathbf{W} :=$ FIND-NEAREST-SEPARATOR($\mathcal{G}', Y, Z'$)
  if $\mathbf{W} = \bot \vee \mathbf{W} \cap De(Y) \neq \emptyset \vee \mathbf{Z} \cap \mathbf{W} \neq \emptyset$ **then**
    return FALSE
  $\mathcal{G}_F := F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$
  if an $|\mathbf{X}|$-flow from $\mathbf{Z}$ to $\mathbf{X}$ exists in $\mathcal{G}_F$ **then**
    return TRUE
  else
    return FALSE

Analysis of the Algorithm. First we show that the algorithm returns TRUE when called on a simple conditional instrumental set $\mathbf{Z}$. Let $\mathbf{W}'$ be a set and (Z_i, π_i) pairs satisfying Definition 6.28. Because all paths from Z' pass through a node of $\mathbf{Z}$, $\mathbf{W}'$ d-separates Z' and Y in $\mathcal{G}'$. According to Proposition 6.13 and Corollary 6.14, algorithm FIND-NEAREST-SEPARATOR finds a nearest separator $\mathbf{W}$ with $(Y \perp\!\!\!\perp Z' \mid \mathbf{W})_{\mathcal{G}'}$, which does not contain descendants of Y. Due to Lemma 6.31, this set does not block any path π_i. Due to Lemma 6.30, these paths form an $|\mathbf{X}|$-flow in $F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \mathbf{W})$.

In the other direction: if the algorithm returns TRUE, it has found a set $\mathbf{W}$ and an $|\mathbf{X}|$-flow from Z' to Y. The set $\mathbf{W}$ d-separates every Z_i from Y in $\overline{\mathcal{G}}$ since it d-separates Z' from Y in $\mathcal{G}'$.

The first two conditions of Definition 6.28 are satisfied by the construction, and because no element of $\mathbf{Z}$ can be a descendant of Y unless there is a path in $\overline{\mathcal{G}}$ that can only be blocked by a descendant of Y.

From Lemma 6.30, we know there are $|\mathbf{X}|$-incompatible paths satisfying condition (c).

The runtime is dominated by the runtime of the maximum flow algorithm. Because the capacities in the flow network are integers, and the value of any flow is bounded by k, the maximum flow can be constructed in time $\mathcal{O}(n + km)$ (see [KT06, Chapter 7.1]). Thus, the maximum flow here is slightly faster than the flow discussed in Section 3.3.8 without a bound on the size of the flow. □

This proves:

Proposition 6.32. *Algorithm* TEST-SIMPLE-COND-IVS *tests, for a given node Y and node sets $\mathbf{X}$ and $\mathbf{Z}$ in a causal DAG $\mathcal{G}$, whether $\mathbf{Z}$ is a simple conditional instrumental set relative to $(\mathbf{X}, Y)$. The runtime of this algorithm is $\mathcal{O}(n + km)$.*

Finding Simple Conditional Instruments

Using the above algorithm, it is trivial to find a simple conditional instrumental set by enumerating all possible sets of size $k = |\mathbf{X}|$ and testing if any of those is a simple conditional instrumental set:

function FIND-SIMPLE-COND-IVS($\mathcal{G}$, $\mathbf{X}$, Y)
  for every subset $\mathbf{Z} \subseteq \mathbf{O}$ with $|\mathbf{Z}| = |\mathbf{X}|$ **do**
    if TEST-SIMPLE-COND-IVS($\mathcal{G}, \mathbf{X}, Y, \mathbf{Z}$) **then**
      return $\mathbf{Z}$
  return $\perp$

Unfortunately, there are $\mathcal{O}(n^k)$ possible sets $\mathbf{Z}$, so the runtime of the algorithm is $\mathcal{O}(n^k(n + km)) = \mathcal{O}(n^{k+3})$. One might attempt to find a faster algorithm, however, Kumor et al. have shown recently that finding $\mathbf{Z}$ is NP-complete [KCB19], so it is unlikely that a significantly faster algorithm exists.

Proposition 6.33. *Algorithm* FIND-SIMPLE-COND-IVS *finds, for a given node Y and a node set $\mathbf{X}$ in a causal DAG $\mathcal{G}$, a simple conditional instrumental set $\mathbf{Z}$ relative to $(\mathbf{X}, Y)$ if such a set exists; otherwise, it returns $\perp$. The runtime of this algorithm is $\mathcal{O}(n^{k+3})$.*

6.5.3 Testing and Finding Simple Instruments

Testing Simple Instruments

Testing whether a given $\mathbf{Z}$ fulfills the conditions of simple instruments can be done in time $\mathcal{O}(n + km)$ using the algorithm TEST-SIMPLE-COND-IVS. To this aim, we modify the algorithm by replacing the calculation of $\mathbf{W}$ as a nearest separator with a fixed $\mathbf{W} = \emptyset$.

function TEST-SIMPLE-IVS($\mathcal{G}, \mathbf{X}, Y, \mathbf{Z}$)
  $\overline{\mathcal{G}} := (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \to Y))$
  if $\mathbf{Z}$ and Y are not d-separated in $\overline{\mathcal{G}}$ **then**
    return FALSE
  $\mathcal{G}_F := F(\mathcal{G}, \mathbf{Z}, \mathbf{X}, \emptyset)$
  if an $|\mathbf{X}|$-flow from $\mathbf{Z}$ to $\mathbf{X}$ exists in $\mathcal{G}_F$ **then**
    return TRUE
  else
    return FALSE

Analysis of the Algorithm. Algorithm TEST-SIMPLE-IVS is identical to TEST-SIMPLE-COND-IVS with $\mathbf{W}$ replaced by $\emptyset$. Since a simple conditional instrumental set with empty $\mathbf{W}$ is a simple instrumental set, TEST-SIMPLE-IVS tests whether $\mathbf{Z}$ is a simple instrumental set. □

Proposition 6.34. *Algorithm* TEST-SIMPLE-IVS *tests, for a given node Y and node sets $\mathbf{X}$ and $\mathbf{Z}$ in a causal DAG $\mathcal{G}$, whether $\mathbf{Z}$ is a simple instrumental set relative to $(\mathbf{X}, Y)$. The runtime of this algorithm is $\mathcal{O}(n + km)$.*

Finding Simple Instruments

Now we present an algorithm to find a simple instrumental set $\mathbf{Z}$ for given $\mathbf{X}$ and Y. Since simple instrumental sets are similar to simple conditional instrumental sets, one might think finding them is an NP-complete problem, too. Fortunately, simple instrumental sets can be found efficiently by searching a maximal flow from S to Y through every node that might be in $\mathbf{Z}$. Since there must be an active path between $\mathbf{X}$ and each node in $\mathbf{Z}$, only nodes in $De(An(\mathbf{X}))$ might be in $\mathbf{Z}$.

function FIND-SIMPLE-IVS($\mathcal{G} = (\mathbf{O} \subseteq \mathbf{V}, \mathbf{E})$, $\mathbf{X}$, Y)
  Let $\overline{\mathcal{G}} = (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \to Y))$
  Let $\mathbf{B}$ be all nodes d-separated from Y in $\overline{\mathcal{G}}$
  $\mathbf{D} := De(An(\mathbf{X})) \cap \mathbf{O} \cap \mathbf{B}$
  $\mathcal{G}_F := F(\mathcal{G}, \mathbf{D}, \mathbf{X}, Pa(Y) \setminus \mathbf{X})$
  if an $|\mathbf{X}|$-flow in $\mathcal{G}_F$ exists **then**
    return $\mathbf{Z}$ = "children of the source S on the flow's paths"
  else
    return $\perp$

Analysis of the Algorithm. Assume there exists a simple instrumental set $\mathbf{Z}$. Then all nodes of $\mathbf{Z}$ are d-separated from Y in $\overline{\mathcal{G}}$, observed, and not d-separated from Y in $\mathcal{G}$, so $\mathbf{Z} \subseteq De(An(\mathbf{X})) \cap \mathbf{O} \cap \mathbf{B}$. There exist $|\mathbf{X}|$-many incompatible, unblocked paths between $\mathbf{Z}$ and $\mathbf{X}$ not containing parents of Y, so due to Lemma 6.30, there exists an $|\mathbf{X}|$-flow in $\mathcal{F}(\mathcal{G}, \mathbf{D}, \mathbf{X}, Pa(Y) \setminus \mathbf{X})$. Thus, FIND-SIMPLE-IVS returns a set.

Assume FIND-SIMPLE-IVS returns a set $\mathbf{Z}$. Then it is d-separated from Y. The flow found by FIND-SIMPLE-IVS can be represented as k disjoint paths $\pi_1, \ldots, \pi_k$ starting at S and then visiting a node of $\mathbf{D}$, from which we form set $\mathbf{Z}$. All nodes in $\mathbf{Z}$ are observed and d-separated from Y in $\overline{\mathcal{G}}$. Each path $\pi_i = V_1^+ \overset{*}{\sim} V_{l_i}^-$ consists of a (possibly empty) subpath of plus nodes followed by a subpath of minus nodes. If the path contains a plus node, it contains a subpath $V_j^+ \to V_{j+1}^-$ at which it switches from the plus layer to the minus layer. All preceding nodes on the path are (+)-ancestors of V_j^+ in the flow graph, so they are descendants of V_j in $\mathcal{G}$. Hence, every node on these paths is in $De(An(\mathbf{X}))$ of $\mathcal{G}$. Thus and due to Lemma 6.30, there exist unblocked, incompatible paths in $\mathcal{G}$ between $\mathbf{Z}$ and $\mathbf{X}$. So $\mathbf{Z}$ is a simple instrumental set. □

If no $|\mathbf{X}|$-flow exists, but a k'-flow with $k' < |\mathbf{X}|$ exists, the children of the source form a simple instrumental set $\mathbf{Z}$ of size k'. So this algorithm can be modified to find a simple instrumental set of maximal size.

That FIND-SIMPLE-IVS has polynomial runtime unlike algorithm FIND-SIMPLE-COND-IVS, shows that the complexity of finding simple conditional instrumental sets results from the difficulty of finding a separator $\mathbf{W}$. Although we can use a nearest separator as $\mathbf{W}$, such a nearest separator cannot be calculated without knowing $\mathbf{Z}$ (cf. our remarks at the end of Section 6.3.2).

Proposition 6.35. *Algorithm* FIND-SIMPLE-IVS *finds, for a given node Y and a node set $\mathbf{X}$ in a causal DAG $\mathcal{G}$, a simple instrumental set $\mathbf{Z}$ relative to $(\mathbf{X}, Y)$ if such a set exists; otherwise, it returns $\perp$. The runtime of this algorithm is $\mathcal{O}(n^{k+3})$.*

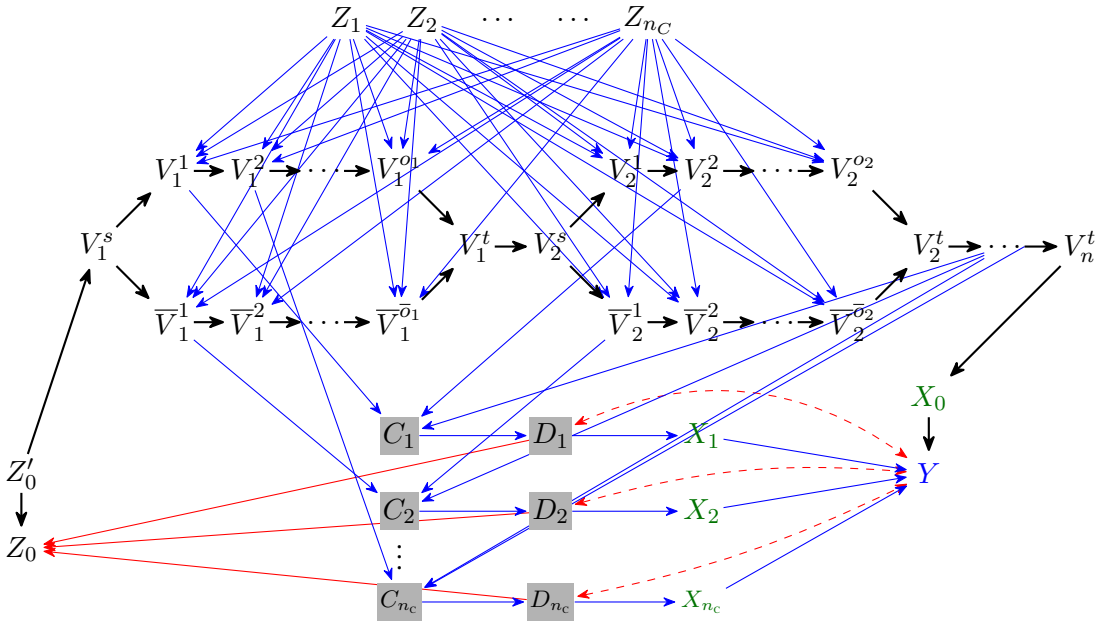


Figure 6.7: A graph $\mathcal{G}$ with a generalized instrumental set $\mathbf{Z}$ constructed from a 3-SAT instance.

6.5.4 Hardness of Testing Generalized Instrumental Sets

Now we prove that it is an NP-complete problem to test whether a given set is a generalized instrumental set:

Theorem 6.36. *Given a DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, a node Y, and sets $\mathbf{X}, \mathbf{Z} \subset \mathbf{V}$, determining whether $\mathbf{Z}$ is a generalized instrumental set relative to $\mathbf{X}$ and Y is an NP-complete problem.*

Proof. The conditions of Definition 6.27 are easily verified after guessing the tuples $(Z_1, \mathbf{W}_1, \pi_1)$, $\ldots$, $(Z_k, \mathbf{W}_k, \pi_k)$. Thus, the problem is in NP.

To prove the NP-hardness, we show a polynomial time reduction from 3-SAT to the problem. Let $\mathbf{V}$ be a set of n_V variables and let $\mathcal{C} = (V_{1,1} \vee V_{1,2} \vee V_{1,3}) \wedge (V_{2,1} \vee V_{2,2} \vee V_{2,3}) \wedge \ldots (V_{n_C,1} \vee V_{n_C,2} \vee V_{n_C,3})$ with $V_{i,j} \in \mathbf{V} \cup \{\overline{V} \mid V \in \mathbf{V}\}$ be a 3-SAT instance with n_C clauses. The variables $\mathbf{V} = \{V_i \mid 1 \leq i \leq n_V\}$ and clauses of $\mathcal{C} = \{C_i \mid 1 \leq i \leq n_C\}$ are arbitrarily indexed. Let $o_i = |\{C \in \mathcal{C} \mid V_i \in C\}|$, resp. $\overline{o}_i$, denote the number of occurrences of literal V_i, resp. $\overline{V}_i$, in $\mathcal{C}$. W.l.o.g. we assume $o_i > 0$ and $\overline{o}_i > 0$.

We adapt the proof given by [EIS76] for multi-commodity flows to instrumental sets. So we construct a DAG $\mathcal{G}$ as shown in Figure 6.7.

$\mathcal{G}$ has the following nodes:

$$\begin{aligned}\mathbf{V}_{\mathcal{G}} = \quad & \{Y, Z_0', Z_0, \ldots, Z_{n_C}, X_0, \ldots, X_{n_C}\} \\ & \cup \{C_1, \ldots, C_{n_C}, D_1, \ldots, D_{n_C}\} \\ & \cup \{V_i^s, V_i^t \mid 1 \leq i \leq n_V\} \\ & \cup \{V_i^j \mid 1 \leq i \leq n_V \wedge 1 \leq j \leq o_i\} \\ & \cup \{\overline{V}_i^j \mid 1 \leq i \leq n_V \wedge 1 \leq j \leq \overline{o}_i\}\end{aligned}$$

and edges:

$$
\begin{aligned}
\mathbf{E} = \quad & \{Z_0 \leftarrow Z_0' \rightarrow V_1^s\} \\
& \cup \{V_i^s \rightarrow V_i^1 \rightarrow \ldots \rightarrow V_i^{o_i} \rightarrow V_i^t \mid 1 \leq i \leq n_V\} \\
& \cup \{V_i^s \rightarrow \overline{V}_i^1 \rightarrow \ldots \rightarrow \overline{V}_i^{\overline{o}_i} \rightarrow V_i^t \mid 1 \leq i \leq n_V\} \\
& \cup \{V_i^t \rightarrow V_{i+1}^s \mid 1 \leq i \leq n_V - 1\} \\
& \cup \{V_{n_V}^t \rightarrow X_0 \rightarrow Y\} \\
& \cup \{Z_i \rightarrow V_j^k \mid 1 \leq i \leq n_C \wedge 1 \leq j \leq n_V \wedge 1 \leq k \leq o_j\} \\
& \cup \{Z_i \rightarrow \overline{V}_j^k \mid 1 \leq i \leq n_C \wedge 1 \leq j \leq n_V \wedge 1 \leq k \leq \overline{o}_j\} \\
& \cup \{V_j^k \rightarrow C_i \mid \text{ the } k\text{-th occurrence of } V_j \text{ is in } C_i\} \\
& \cup \{\overline{V}_j^k \rightarrow C_i \mid \text{ the } k\text{-th occurrence of } \overline{V}_j \text{ is in } C_i\} \\
& \cup \{C_i \rightarrow D_i \rightarrow X_i \rightarrow Y \mid 1 \leq i \leq n_V\} \\
& \cup \{Y \leftrightarrow D_i \rightarrow Z_0 \mid 1 \leq i \leq n_V\}
\end{aligned}
$$

We use indices 0 to n_C for X_i instead of 1 to $n_C + 1$ to simplify the notation. We claim that there exists an assignment to $V_1, \ldots, V_{n_V}$ that satisfies $\mathcal{C} = \bigwedge_i C_i$ iff $\mathbf{Z} = \{Z_0, \ldots, Z_{n_C}\}$ is a generalized instrumental set relative to $\mathbf{X} = \{X_0, \ldots, X_{n_C}\}$ and Y in $\mathcal{G}$.

($\Leftarrow$): Assume $\mathbf{Z}$ is a generalized instrumental set. Then there exist tuples $(Z_{i_0}, \mathbf{W}_0, \pi_0)$, $(Z_{i_1}, \mathbf{W}_1, \pi_1), \ldots, (Z_{i_{n_C}}, \mathbf{W}_{n_C}, \pi_{n_C})$ satisfying Def. 6.27. First we show that the path from Z_0 actually ends at $X_0 \rightarrow Y$. There are active paths $Y \leftrightarrow D_i \rightarrow Z_0$ for all D_i, which need to be blocked. Thus, nodes D_i are in the $\mathbf{W}_j$ associated with the path starting at Z_0, so the path cannot contain D_i. Since $X_1, \ldots, X_{n_C}$ can only be reached by traversing $D_1, \ldots, D_{n_C}$, the path has to end at X_0.

Since the nodes $Z_1, \ldots, Z_{n_C}$ are all connected to exactly the same nodes, we can assume w.l.o.g. that path π_i starts at Z_i.

Every node V_i^j is only visited by a directed subpath $\rightarrow V_i^j \rightarrow$ because every path can only enter it through a $\rightarrow$ edge. So none of these nodes is visited by two paths; otherwise, condition *(c)* of Definition 6.27 (that the subpath $\pi_{i'}[V_i^j \overset{*}{\sim} X_{i'}]$ has to point to V_i^j) would be violated.

Since path π_0 can neither visit node C_i nor Z_i for $i > 0$ through a collider, it visits V_i^s and then passes either through the upper path or the lower path to V_i^t. We assign the following values to the variables V_i

$$
V_i := \begin{cases} \text{TRUE} & \text{if } V_i^1 \notin \pi_0, \\ \text{FALSE} & \text{otherwise.} \end{cases}
$$

This assignment satisfies the formula: Assume there is clause C_i that is not satisfied. We know that path π_i has the form $Z_i \rightarrow W_k^j \rightarrow \ldots W_k^{j'} \rightarrow C_i \rightarrow D_i \rightarrow X_i \rightarrow Y$ for W's corresponding to one variable V_k or its negation $\overline{V}_k$, since π_i cannot cross through V_k^t to another lobe; otherwise, it would intersect π_0 at V_k^t. Also $W_k^t \notin \pi_0$. If W_k^j corresponds to V_k, then V_k is TRUE and clause C_i contains variable V_k. If W_k^j corresponds to $\overline{V}_k$, then V_k is FALSE and C_i contains the negation. So C_i is satisfied.

($\Rightarrow$): Let $V_i \in \{\text{TRUE}, \text{FALSE}\}$ be a satisfying assignment for the variables V_i. Assume C_i is satisfied by a literal $W \in C_i$ which is the k-th occurrence of a variable V_j in $\mathcal{C}$. Let

$v(C_i) \in \{V_j^k, \overline{V}_j^k\}$ be the node corresponding to W. Let $p(i) = V_i^1 \to \ldots \to V_i^{o_i}$ if $V_i = \text{FALSE}$; otherwise, let $p(i) = \overline{V}_i^1 \to \ldots \to \overline{V}_i^{o_i}$. We choose the following tuples

- $(Z_0, \{C_i, D_i \mid 1 \leq i \leq n_C\}, Z_0 \leftarrow Z_0' \to V_1^s \to p(1) \to V_1^t \to V_2^s \to p(2) \to V_2^t \to V_3^s \to \ldots \to p(n_V) \to V_{n_V}^t \to X_0 \to Y)$,
- $(Z_1, \emptyset, Z_1 \to v(C_1) \to C_1 \to D_1 \to X_1 \to Y)$,
- ...,
- $(Z_{n_C}, \emptyset, Z_{n_C} \to v(C_{n_C}) \to C_{n_C} \to D_{n_C} \to X_{n_C} \to Y)$,

which satisfy the conditions of Definition 6.27:

(a) Y does not have any descendants, and each π_i is an unblocked path connecting Z_i with $X_i \to Y$.

(b) In $\overline{\mathcal{G}}$, all paths starting at Y begin with $Y \leftrightarrow D_i$. In the first tuple, the paths $Y \leftrightarrow D_i \to Z_0$ are blocked by D_i and the paths $Y \leftrightarrow D_i \leftarrow C_i \leftarrow$ are blocked by C_i. In all other tuples, the paths $Y \leftrightarrow D_i \to Z_0$ are irrelevant and the paths $Y \leftrightarrow D_i \leftarrow C_i \leftarrow$ are blocked by D_i. No path π_i is blocked by $\mathbf{W}_i$.

(c) No path $\pi_1, \ldots, \pi_{n_C}$ has a common node with π_0. Otherwise, a node V_k^j would correspond to a variable V_k that is FALSE, but literal V_k satisfies clause C_i; or a variable that is TRUE but literal $\overline{V}_k$ satisfies C_i. Paths $\pi_1, \ldots, \pi_{n_C}$ are vertex disjoint or the k-th occurrence of a variable would be in two different clauses. □

6.5.5 Testing and Finding Generalized Instrumental Sets with a Pebble Game

If the size of the set $\mathbf{Z}$ is bound by a constant, it is possible to test whether $\mathbf{Z}$ is a generalized instrumental set in polynomial time. The testing algorithm needs to find separators $\mathbf{W}_i$ and incompatible paths π_i. We will see that nearest separators can again be used as separators $\mathbf{W}_i$, but finding the paths π_i has become harder. Since each path π_i needs to be unblocked by a different $\mathbf{W}_i$, the end nodes of the paths are not interchangeable, so the paths cannot be found by a network flow.

Nevertheless, for a fixed k, finding k paths that are just node-disjoint is a well-researched problem (k-vertex disjoint paths problem, k-VDPP, or k-linkage), and known to be NP-complete in general directed graphs [GJ79a], but solvable in DAGs in polynomial time [FHW80].

Generalized Vertex Disjoint Paths Problem

The problem k-VDPP asks, given $2k$ not necessarily distinct nodes $(s_1, \ldots, s_k), (t_1, \ldots, t_k)$, whether there are k paths from each s_i to t_i that do not share a common node except for the end nodes.

We generalize k-VDPP to find directed paths that satisfy the following conditions:

Definition 6.37 (Generalized vertex disjoint paths problem (k-GVDPP)). *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG, let $(S_1, \ldots, S_k)$ and $(T_1, \ldots, T_k)$ be $2k$ not necessarily distinct nodes of $\mathcal{G}$, let $\mathbf{W}_1, \ldots, \mathbf{W}_k \subseteq \mathbf{V}$ be sets of nodes with $S_i, T_i \notin \mathbf{W}_i$, and let $\mathbf{C} \subseteq \{\{i, j\} \mid 1 \leq i, j \leq k\}$ be a set of pairs.*

The k-GVDPP asks whether there exist paths p_i such that

(a) *p_i is a directed path from S_i to T_i,*

(b) *p_i does not contain a node of $\mathbf{W}_i$, and*

(c) *p_i does not share a node with p_j, $i \neq j$, unless that node is S_i, T_i, S_j, T_j; or $\{i, j\} \in \mathbf{C}$.*

We generalize the pebbling game algorithm given by [SP78] for $k = 2$ and generalized by [FHW80] to arbitrary k.

Our pebble game is defined by the following rules, of which rule 2 and 3 may be applied arbitrarily often in any order. In the description below, the level of a node V is defined to be the length of the longest, directed path starting at V.

1. Initially: create k pebbles $p_1, \ldots, p_k$, and place p_i on S_i.

2. Pebble p_i may be moved along a directed edge $V \to W$ if W is not in $\mathbf{W}_i$, and
   - V has the largest level of any pebbled node, and
   - there is no pebble p_j on W unless $\{i, j\} \in \mathbf{C}$ or $W \in \{S_j, T_i\}$.

3. Pebble p_i may be removed once it reaches T_i.

4. The game is won if all pebbles are removed.

This game is equivalent to the k-GVDPP problem and it can be played efficiently:

Lemma 6.38. *The pebble game can be won iff there exists a solution to k-GVDPP.*

Proof. If there exists a solution, it is easy to see that the game can be won by moving pebble p_i along the path p_i whenever pebble p_i can be moved.

If the game can be won, the pebbles trace paths p_i through the graph that satisfy the conditions of k-GVDPP: Each p_i is a directed path from S_i to T_i and does not contain a node of $\mathbf{W}_i$. Two paths p_i and p_j with $\{i, j\} \notin \mathbf{C}$ cannot intersect each other in an internal node; otherwise, there would be a node X that is visited by pebble p_i and p_j. Assume w.l.o.g. p_i moves to node X first. Then the game cannot be won since pebble p_j on an ancestor of X must move before pebble p_i, but cannot move to node X as long as p_i is on that node. □

Lemma 6.39. *There exists an $\mathcal{O}(k(n+1)^{k+1})$ algorithm to solve k-GVDPP.*

Proof. There are only $\mathcal{O}((n+1)^k)$ different pebble placements and $\mathcal{O}((n+1)k)$ different transitions of one pebble moving to another node, so a reachability search on the graph of all placements can be done in $\mathcal{O}(k(n+1)^{k+1})$. □

Testing Generalized Instrumental Sets

We now reduce the problem of testing whether a set $\mathbf{Z}$ is a generalized instrumental set relative to $\mathbf{X}$ and Y to the k-GVDPP. Thereby, we use nearest separators as the sets $\mathbf{W}_i$. Because the paths π_i of the generalized instrumental set are active given $\emptyset$, they consist of one or two directed paths each, whose end nodes become the nodes $S_{i'}$ and $T_{i'}$ of k-GVDPP. The first end node is in $\mathbf{Z}$ and the last end node in $\mathbf{X}$. If the path π_i is split into two directed paths, the fork on the path becomes another end node for k-GVDPP. Since we cannot know a priori, which node is the fork, we repeat the construction and try every node of the graph as the fork. The condition that the paths need to be incompatible can be encoded in the constraints $\mathbf{C}$.

function TEST-GENERAL-IVS($\mathcal{G}$, $\mathbf{X}$, Y, $\mathbf{Z}$)
  $k := |\mathbf{X}|$
  for i in $1, \ldots, k$ **do**
    $\mathbf{W}_i :=$ a nearest separator for (Y, Z_i) in $\overline{\mathcal{G}}$
    if $\mathbf{W}_i = \perp \vee (\mathbf{W}_i \cap De(Y) \neq \emptyset)$ **then**
      return FALSE
  for all permutations π and π' of $\{1, \ldots, k\}$ **do**
    Let $(X_1, \ldots, X_k)$ corresponds to π
    Let $(Z_1, \ldots, Z_k)$, $(\mathbf{W}_1, \ldots, \mathbf{W}_k)$ correspond to π'
    if $\exists i : X_i \in \mathbf{W}_i$ **then continue**
    if $\exists i, j : i < j \wedge X_i = Z_j$ **then continue**
    for all $F_1, \ldots, F_k \in \mathbf{V} \setminus (Y \cup \mathbf{W}_1 \cup \ldots \cup \mathbf{W}_k)$ **do**
      if $\exists i \neq j : F_i \in \{F_j, X_j, Z_j\}$ **then**
        continue
      $\mathbf{F} := \{F_i \mid 1 \leq i \leq k\}$
      $k' := k + |\mathbf{F} \setminus (\mathbf{X} \cup \mathbf{Z})|$
      Construct a k'-GVDPP instance
        $S_1, \ldots, S_{k'}, T_1, \ldots, T_{k'}, \mathbf{W}'_1, \ldots, \mathbf{W}'_{k'}, \mathbf{C}$:
      $p := 1$
      $\mathbf{U} := \emptyset$
      $\mathbf{C} := \emptyset$
      for i in $1, \ldots, k$ **do**
        if $F_i = X_i$ **then**
          ADD-ARC $(i, X_i, Z_i, \mathbf{W}_i \cup \mathbf{F} \cup Y)$
        else if $F_i = Z_i$ **then**
          ADD-ARC $(i, Z_i, X_i, \mathbf{W}_i \cup \mathbf{F} \cup Y)$
        else
          ADD-ARC $(i, F_i, X_i, \mathbf{W}_i \cup \mathbf{F} \cup Y)$
          ADD-ARC $(i, F_i, Z_i, \mathbf{W}_i \cup \mathbf{F} \cup Y)$
      if $(S_1, \ldots, S_{k'}, T_1, \ldots, T_{k'}, \mathbf{W}'_1, \ldots, \mathbf{W}'_{k'}, \mathbf{C})$
        is a Yes-instance of k'-GVDPP
        then return TRUE
  return FALSE

Analysis of the Algorithm. The algorithm begins by creating a nearest separator relative to (Y, Z_i) in $\overline{\mathcal{G}}$ for each i to use it as set $\mathbf{W}_i$. Next it enumerates all permutations $Z_1, \ldots, Z_k$ of $\mathbf{Z}$ and $X_1, \ldots, X_k$ of $\mathbf{X}$ as well as all combinations for directed and fork paths for each π_i, i.e., π_i is considered either as directed from Z_i to X_i, directed from X_i to Z_i, or containing a fork F_i. Knowing the direction and/or fork of the path, we can treat it as one or two directed paths. From condition *(c)* of Definition 6.27, it follows that two of these directed paths can only intersect each other iff one path is directed towards a Z_i and the other path towards an X_j with $i < j$. These nodes and constraints directly correspond to a k-GVDPP instance with up to $2k$ nodes. We use an auxiliary procedure, called ADD-ARC and given below, to add the start and end nodes to the k-GVDPP instance and update the set $\mathbf{C}$ appropriately.

If one of these k-GVDPP instances has a solution, $\mathbf{Z}$ is a generalized instrumental set, and the algorithm returns TRUE. □

Below, we present the pseudocode for ADD-ARC:

function ADD-ARC($i, S, T, \mathbf{W}$)
  $S_p := S$; $T_p := T$;
  if $T = Z_i$ **then**
    ▷ Path towards the instrument
    $\mathbf{W}'_p := \mathbf{W} \cup \mathbf{Z} \cup \{X_j \mid i \geq j \vee F_j = X_j\}$
    $\mathbf{U} := \mathbf{U} \cup \{p\}$
  else
    ▷ Path towards Y
    $\mathbf{W}'_p := \mathbf{W} \cup \{Z_j \mid i \leq j \vee F_j = Z_j\} \cup \mathbf{X}$
    $\mathbf{C} := \mathbf{C} \cup \{\{u, p\} \mid u \in \mathbf{U}\}$
  $\mathbf{W}'_p := \mathbf{W}'_p \setminus \{S, T\}$
  $p := p + 1$

Lemma 6.40. *There exists an algorithm which, for a given Y and sets of k nodes $\mathbf{X}$ and $\mathbf{Z}$, using a solver for GVDPP, tests whether $\mathbf{Z}$ is a generalized instrumental set relative to $\mathbf{X}$ and Y calling the solver $\mathcal{O}((k!)^2 n^k)$ times for k'-GVDPP instances, where $k' \in \{k, \ldots, 2k\}$.*

Proof. If $\mathbf{Z}$ is a generalized instrumental set relative to $\mathbf{X}$ and Y, algorithm TEST-GENERAL-IVS will return TRUE: Let $(Z_1, \mathbf{W}_1, \pi_1), \ldots, (Z_k, \mathbf{W}_k, \pi_k)$ be the triples of Definition 6.27. We consider the iteration in which the algorithm tests the same permutation of $\mathbf{X}, \mathbf{Z}$. These triples remain valid if we replace the $\mathbf{W}_i$ with the nearest separating sets used by the algorithm, because a nearest separator does not contain descendants of Y and neither blocks path π_i nor contains X_i if such a set exists due to Proposition 6.13, Corollary 6.14 and Lemma 6.31. There is no $i < j$ with $X_i = Z_j$, since Z_j must not appear in path π_i.

Now consider the iteration in which each F_i is the fork on path π_i or, if π_i does not have a fork and is a directed path from Z_i to Y, is Z_i. Since the paths end at Y, this fork cannot be Y. It cannot be in $\mathbf{W}_i$ or π_i was blocked. A path π_i cannot intersect π_j at a fork, since no arrow at a fork points to that fork violating condition three of Def. 6.27. So the condition $\exists i \neq j : F_i \in \{F_j, X_j, Z_j\}$ is false, and the algorithm does not abort there. The set $\mathbf{F}$ is the set of all forks, including Z_i for paths π_i without a fork.

The algorithm now creates a $(k' = k + |\mathbf{F} \setminus (\mathbf{X} \cup \mathbf{Z})|)$-GVDPP instance, whose solution is the paths π_i, whereby paths containing a fork are split into two directed paths $\pi_{i,\leftarrow}, \pi_{i,\rightarrow}$ and the tail $\rightarrow Y$ of every path is removed. It does this by calling ADD-ARC for all endpoint node pairs of the split paths in the innermost loop.

The sets $\mathbf{W}'_i$ created by ADD-ARC do not contain an internal node of the respective $\pi_{i,\leftarrow}$ or $\pi_{i,\rightarrow}$: This is true for the initial $\mathbf{W}_i \cup \mathbf{F} \cup Y$ as argued above. An arc $\pi_{i,\leftarrow}$ towards Z_i cannot contain another Z_j. For $j > i$, this follows directly from Def. 6.27 *(c)*, for $j < i$, $\pi_{i,\leftarrow}[Z_i \overset{*}{\leftarrow} Z_j]$ should point towards Z_j, but this path is directed towards Z_i. For the same reason, it cannot contain a node X_j with $j < i$. It also cannot contain X_j if X_j is a fork on π_j. An arc $\pi_{i,\rightarrow}$ cannot contain a Z_j with $j > i$ due to condition *(c)* of Definition 6.27. If π_j is a directed path from Z_j to Y, π_i cannot contain Z_j with $j < i$ as $\pi_j[Z_j \overset{*}{\rightarrow} Y]$ should point towards Z_j. π_i cannot contain a X_j with $j < i$ because then $\pi_j[X_j \overset{*}{\rightarrow} Y]$ should point towards X_j, but is an edge $X_j \rightarrow Y$. For $j > i$, $\pi_i[X_j \overset{*}{\rightarrow} Y]$ should point towards X_j, but all edges on $\pi_{i,\rightarrow}$ point towards Y. Since π_i is a path, X_i as well as Z_i can only occur in one of the arcs $\pi_{i,\leftarrow}, \pi_{i,\rightarrow}$.

Also all intersecting arcs are included in the set $\mathbf{C}$: The set $\mathbf{U}$ contains the indices of all upwards directed arcs, i.e., when ADD-ARC for $\pi_{i,\rightarrow}$ is called, $\mathbf{U}$ contains the indices of all

$\pi_{j,\leftarrow}$ with $j < i$. Arcs directed in the same direction, i.e., $\pi_{i,\rightarrow}$ and $\pi_{j,\rightarrow}$ or $\pi_{i,\leftarrow}$ and $\pi_{j,\leftarrow}$, cannot intersect each other, since the edges at an intersection have to point in opposite directions. So only intersections between an upward $\pi_{i,\leftarrow}$ and a downward directed arc $\pi_{j,\rightarrow}$ occur, with $i < j$ due to condition *(c)* of Definition 6.27 . These are precisely the arcs contained in $\mathbf{C}$.

Thus, the generated k'-GVDPP instance is solvable and the algorithm returns TRUE.

In the other direction, if the algorithm returns TRUE, $\mathbf{Z}$ is a general instrumental set: Then the algorithm has fixed permutations $X_1, \ldots, X_k$, $Z_1, \ldots, Z_k$, sets $\mathbf{W}_1, \ldots, \mathbf{W}_k$, and paths $\pi_1 = \pi_{1,\leftarrow}\pi_{1,\rightarrow} \rightarrow Y, \ldots, \pi_k = \pi_{k,\leftarrow}\pi_{k,\rightarrow} \rightarrow Y$, where $\pi_{i,\leftarrow}, \pi_{i,\rightarrow}$ denote the paths in the solution of k'-GVDPP for the nodes added by the call of ADD-ARC for i in the respective direction. $\mathbf{W}_i$ does not contain descendants of Y, d-separates Z_i from Y in $\overline{\mathcal{G}}$, and does not block π_i.

For $1 \leq i < j \leq k$, the node Z_j does not occur as an internal node in π_i, since it was added to the forbidden nodes in $\mathbf{W}'_i$. If paths π_i and π_j have a common node, this node is either an endpoint node (of an arc) or the indices of the corresponding arcs were added to $\mathbf{C}$.

If two endpoints nodes are the same, this node cannot be a fork in π_i or π_j, since there is no F_i with $F_i \in \{F_j, X_j, Z_j\}$ for $i \neq j$. There is also no $Z_i = Z_j$ or $X_i = X_j$; thus, the intersecting node has to be $Z_i = X_j$ with $i \leq j$ due to the test after the $\mathbf{X}$ permutation. If $i = j$, it is not an intersection. Because $F_i \neq Z_i$ and $F_j \neq X_j$, both $\pi_i[Z_i \overset{*}{\sim} Y]$ and $\pi_j[Z_j \overset{*}{\sim} X_j]$ point towards $Z_i = X_j$ satisfying 6.27 *(c)*.

If it is only an endpoint node in one arc, it can only be a node of $\mathbf{X} \cup \mathbf{Z}$, because all other endpoint nodes are in $\mathbf{F}$ and thus forbidden. Within $\pi_{i,\leftarrow}$, only a node X_j with $i < j$ and $F_j \neq X_j$ can occur, leading to subpaths in π_i and π_j of $\leftarrow X_j \leftarrow$ and $\rightarrow X_j \rightarrow$. Within $\pi_{i,\rightarrow}$, only Z_j with $i > j$ and $F_j \neq Z_j$ occurs, leading to subpaths $\rightarrow Z_j \rightarrow$ and $Z_j \leftarrow$. Both intersections are allowed by Definition 6.27 *(c)*.

If the paths intersect at an internal node V, it only occurs in arcs with indices in $\mathbf{C}$, i.e., $\pi_{i,\leftarrow}$ and $\pi_{j,\rightarrow}$ with $i < j$. Thus, the subpaths are $\leftarrow V \leftarrow$ and $\rightarrow V \rightarrow$, which is also allowed by Definition 6.27 *(c)*.

Hence, all conditions of Definition 6.27 are satisfied, and $\mathbf{Z}$ is a generalized instrumental set. □

Proposition 6.41. *Algorithm* TEST-GENERAL-IVS *tests, for a given node Y and node sets $\mathbf{X}$ and $\mathbf{Z}$ in a causal DAG $\mathcal{G}$, whether $\mathbf{Z}$ is a simple instrumental set relative to $(\mathbf{X}, Y)$. The runtime of this algorithm is $\mathcal{O}(k(k!)^2 n^{3k+1})$.*

Finding Generalized Instrumental Sets

Like in the case of simple conditional instrumental sets, we only know how to find generalized instrumental sets with an exhaustive search:

```
function FIND-GENERAL-IVS(G, X, Y)
    for every subset Z ⊆ O with |Z| = |X| do
        if TEST-GENERAL-IVS(G, X, Y, Z) then
            return Z
```

Proposition 6.42. *Algorithm* FIND-GENERAL-IVS *finds, for a given node Y and a node set $\mathbf{X}$ in a causal DAG $\mathcal{G}$, a simple instrumental set $\mathbf{Z}$ relative to $(\mathbf{X}, Y)$ if such a set exists; otherwise, it returns $\perp$. The runtime of this algorithm is $\mathcal{O}(k(k!)^2 n^{4k+1})$.*

6.6 Discussion

We have analyzed the complexity of conditional instrumental variables and instrumental sets. In both cases, it is NP-complete to test whether a given IV (or set of IVs) is actually a conditional IV (or generalized instrumental set). We have defined a less general variant that can be tested and found in polynomial time. Table 6.2 and Table 6.3 present all results.

Our ancestral instrumental variables, as a special case of conditional IVs, can be tested in linear-time and can be found as well as enumerated in $\mathcal{O}(nm)$. Furthermore, whenever a conditional IV exists, an ancestral IV also exists. Hence, we can always find a conditional IV in $\mathcal{O}(nm)$ when one exists. From a computational complexity perspective, the result that instrumentalization is hard whereas finding a conditional IV is easy is rather intriguing. This can be explained by noting that the solution space of the IV problem decomposes into some instances that are easy to find (ancestral IVs) and others that are hard to find (non-ancestral IVs). One can also consider testing as a constrained version of finding with the given IV as an additional constraint. It is not unusual that an easy problem becomes hard when constraints are added. Nevertheless, in this case, adding as a second constraint that the conditioning set should be ancestral changes the problem to testing an ancestral IV and makes the problem easy again.

A simple conditional instrumental set of size k, as a special case of generalized instrumental sets, can be tested in $\mathcal{O}(n + km)$. Finding it or enumerating all such sets requires $\mathcal{O}(n^{k+3})$ time, which is only polynomial if the size k is bounded.

Here adding the constraint that all conditioning sets $\mathbf{W}_i$ should be equal has changed a hard problem to an easy problem. However, comparing the network flow to the pebble game shows that generalized instrumental sets are hard because the connections between $\mathbf{X}$ and $\mathbf{Z}$ are not arbitrary. The network flow does not distinguish the nodes of $\mathbf{X}$ ($\mathbf{Z}$) from each other and might connect any pair of them by a path, while the pebble game only connects X_i to Z_i. So the constraint on the conditioning sets actually removes constraints of the paths.

One surprising result is that it is possible to find a nearest separator that does not block the path between X and Z without knowing this path and without considering X. Even more surprising is that it can be found in linear-time and is related to the minimal separators. In this chapter, we have used minimal separators as nearest separators, whereby algorithm FINDMINSEP finds the minimal separator with two calls to REACHABLE. It is easy to see that the first call to REACHABLE already returns a nearest separator. Thus, one could inversely implement the nearest separator algorithm FIND-NEAREST-SEPARATOR as a single call of REACHABLE and then define minimal separators as two calls to FIND-NEAREST-SEPARATOR [ZL19].

An interesting problem for future research is also to find an efficiently testable subclass of generalized instruments which is larger than the simple conditional instrumental sets provided in this thesis.

7 Discussion

We have developed efficient algorithms for separating sets, adjustment sets, and instrumental variables. Most of the algorithms have been implemented in DAGitty [Tex+16].

They are the first sound and complete algorithms to identify the causal effect by covariate adjustment in DAGs, RCGs, and MAGs. Whenever an adjustment set exists, they can find one or even a minimal one in linear-time. We can also find a minimum adjustment set or enumerate all adjustment sets with polynomial delay. Although the causal effect in some models cannot be identified by adjustment and requires more powerful methods like the do-calculus, identification by adjustment is preferred in practice due to its statistical properties [SVR10]. We have empirically evaluated methods beyond adjustment in random graphs.

For linear SEMs, we give the first sound and complete algorithms to identify the direct causal effect with a conditional instrumental variable. Although testing whether a certain variable is a conditional IV is NP-complete, we can find some conditional IVs in $\mathcal{O}(nm)$ time whenever at least one exists. We have also studied instrumental sets, and although testing and finding them is NP-complete in the most general case, simple conditional sets can be tested and – if of bounded size – found in polynomial time.

Some open questions remain for further research:

- Can a minimum separator and adjustment set be found in time $\mathcal{O}(nm)$?
- Can the delay complexity of enumerating minimal separators and adjustment sets be improved to $\mathcal{O}(nm)$?
- Can our algorithms be generalized to further classes like PAGs or directed cyclic graphs?
- Are nearest separators useful for other classes than DAGs or SEMs?
- Is it possible to find ancestral instruments in linear-time?
- What is the complexity of identifying causal effects in SEMs?
- What is the largest subclass of SEMs in which the causal effect can be identified in polynomial time?
- Can instruments relative to the total effect be used for anything?
- Which causal effects can be identified in non-linear parametric models?

We hope our algorithms will be useful to epidemiologists, econometrists, and climatologists who investigate the four important questions asked at the beginning of the introduction.

8 Bibliography

[ADC96] Silvia Acid and Luis M De Campos. "An algorithm for finding minimum d-separating sets in belief networks". In: *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 1996, pp. 3–10.

[AIR96] Joshua D. Angrist, Guido W. Imbens, and Donald B. Rubin. "Identification of Causal Effects Using Instrumental Variables". In: *Journal of the American Statistical Association* 91.434 (1996), pp. 444–455.

[AMP97] Steen A Andersson, David Madigan, and Michael D Perlman. "On the Markov equivalence of chain graphs, undirected graphs, and acyclic digraphs". In: *Scandinavian Journal of Statistics* 24.1 (1997), pp. 81–102.

[AMP+97] Steen A Andersson, David Madigan, Michael D Perlman, et al. "A characterization of Markov equivalence classes for acyclic digraphs". In: *The Annals of Statistics* 25.2 (1997), pp. 505–541.

[Ang98] Joshua D. Angrist. "Estimating the Labor Market Impact of Voluntary Military Service Using Social Security Data on Military Applicants". In: *Econometrica* 66.2 (1998), pp. 249–288.

[AP08] Joshua D. Angrist and Jörn-Steffen Pischke. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press, 2008.

[Bol89] Kenneth A Bollen. *Structural equations with latent variables*. John Wiley & Sons, 1989.

[Bon+18] Stephan Bongers, Jonas Peters, Bernhard Schölkopf, and Joris M. Mooij. "Theoretical Aspects of Cyclic Structural Causal Models". In: arXiv:1611.06221v2 [stat.ME] (2018).

[BP02] Carlos Brito and Judea Pearl. "Generalized Instrumental Variables". In: *UAI*. 2002, pp. 85–93.

[Bri04] Carlos Brito. "Graphical Methods for Identification in Structural Equation Models". PhD Thesis. Dept. of Comp. Sc., University of California, Los Angeles, 2004.

[Bri10] Carlos Brito. "Instrumental sets". In: *Heuristics, Probability and Causality. A Tribute to Judea Pearl* (2010), pp. 295–307.

[BS95] Remco R Bouckaert and Milan Studenỳ. "Chain graphs: semantics and expressiveness". In: *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*. Springer, 1995, pp. 69–76.

[BT84] R.J. Bowden and D.A. Turkington. *Instrumental variables*. Cambridge University Press, 1984.

[BTP14] Elias Barenboim, Jin Tian, and Judea Pearl. "Recovering from Selection Bias in Causal and Statistical Inference". In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 2014, pp. 2410–2416.

[Chi02] David Maxwell Chickering. "Learning equivalence classes of Bayesian-network structures". In: *The Journal of Machine Learning Research* 2 (2002), pp. 445–498.

[Chi95] David Maxwell Chickering. "A transformational characterization of equivalent Bayesian network structures". In: *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 87–98.

[Dun75] Otis Dudley Duncan. *Introduction to structural equation models*. Academic Press, 1975.

[EIS76] S. Even, A. Itai, and A. Shamir. "On the Complexity of Timetable and Multicommodity Flow Problems". In: *SIAM Journal on Computing* 5.4 (1976), pp. 691–703.

[Elw13] Felix Elwert. "Graphical Causal Models". In: *Handbook of Causal Analysis for Social Research*. Handbooks of Sociology and Social Research. Springer, 2013, pp. 245–273.

[Eve79] Shimon Even. *Graph Algorithms*. Computer Science Press, 1979.

[FDD12] Rina Foygel, Jan Draisma, and Mathias Drton. "Half-trek criterion for generic identifiability of linear structural equation models". In: *The Annals of Statistics* 40.3 (June 2012), pp. 1682–1713.

[FHW80] Steven Fortune, John Hopcroft, and James Wyllie. "The directed subgraph homeomorphism problem". In: *Theoretical Computer Science* 10.2 (1980), pp. 111 –121. ISSN: 0304-3975.

[Fis66] Franklin M. Fisher. *The identification problem in econometrics*. McGraw-Hill, 1966.

[FM17] Patrick Forré and Joris M. Mooij. "Markov Properties for Graphical Models with Cycles and Latent Variables". In: arXiv:1710.08775 [math.ST] (2017).

[Fry90] Morten Frydenberg. "The chain graph Markov property". In: *Scandinavian Journal of Statistics* 17 (1990), pp. 333–353.

[GJ79a] Michael Garey and David Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman and Company, 1979.

[GJ79b] Michael Garey and David Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. 1979.

[GK17] Adam Glynn and Konstantin Kashin. "Front-door Versus Back-door Adjustment with Unmeasured Confounding: Bias Formulas for Front-door and Hybrid Adjustments with Application to a Job Training Program". In: *Journal of the American Statistical Association* (2017).

[GP98] David Galles and Judea Pearl. "An axiomatic characterization of causal counterfactuals". In: *Foundations of Science* 3.1 (1998), pp. 151–182.

[GPSS10] Luis D. García-Puente, Sarah Spielvogel, and Seth Sullivant. “Identifying Causal Effects with Computer Algebra”. In: *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2010, pp. 193–200.

[HPM19] Leonard Henckel, Emilija Perković, and Marloes H. Maathuis. *Graphical Criteria for Efficient Total Effect Estimation via Adjustment in Causal Linear Models*. 2019. arXiv: 1907.02435.

[HV06] Yimin Huang and Marco Valtorta. “Pearl’s calculus of intervention is complete”. In: *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2006, pp. 217–224.

[Imb14] Guido Imbens. “Instrumental Variables: An Econometrician’s Perspective”. In: *Statistical Science* 29.3 (2014), pp. 323–358.

[Kal+12] Markus Kalisch, Martin Mächler, Diego Colombo, Marloes Maathuis, and Peter Bühlmann. “Causal Inference Using Graphical Models with the R Package pcalg”. In: *Journal of Statistical Software* 47.11 (2012), . ISSN: 1548-7660.

[KCB19] Daniel Kumor, Bryant Chen, and Elias Bareinboim. “Efficient Identification in Linear Structural Causal Models with Instrumental Cutsets”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 12477–12486.

[Kos02] Jan T.A. Koster. “Marginalizing and conditioning in graphical models”. In: *Bernoulli* 8.6 (Dec. 2002), pp. 817–840.

[KT06] Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.

[Lau+90] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H.-G. Leimer. “Independence properties of directed Markov fields”. In: *Networks* 20.5 (1990), pp. 491–505.

[LG14] François Le Gall. “Powers of tensors and fast matrix multiplication”. In: *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*. ACM. 2014, pp. 296–303.

[LS88] S. L. Lauritzen and D. J. Spiegelhalter. “Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems”. In: *Journal of the Royal Statistical Society. Series B* 50.2 (1988), pp. 157–224.

[LW89] Steffen Lauritzen and Nanny Wermuth. “Graphical models for associations between variables, some of which are qualitative and some quantitative”. In: *The Annals of Statistics* 17 (1989), pp. 31–57.

[MC15] Marloes H. Maathuis and Diego Colombo. “A generalized backdoor criterion”. In: *Annals of Statistics* 43.3 (June 2015), pp. 1060–1088.

[Mee95] Christopher Meek. “Causal inference and causal explanation with background knowledge”. In: *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 403–410.

[Oei] *The On-Line Encyclopedia of Integer Sequences*. OEIS Foundation Inc., 2019.

[Orl13] James B. Orlin. “Max flows in O(nm) time, or better”. In: *Proceedings of the 45th annual ACM symposium on Theory of computing*. ACM. 2013, pp. 765–774.

[Pap93] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1993. ISBN: 0201530821.

[Pea01] Judea Pearl. *Parameter identification: A new perspective*. Tech. rep. R-276. UCLA, 2001.

[Pea09] Judea Pearl. *Causality*. Cambridge University Press, 2009, p. 464. ISBN: 0-521-77362-8.

[Pea93] Judea Pearl. "Comment: Graphical models, causality and intervention". In: *Statistical Science* 8 (1993), pp. 266–269.

[Per+15] Emilija Perković, Johannes Textor, Markus Kalisch, and Marloes Maathuis. "A Complete Generalized Adjustment Criterion". In: *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2015, pp. 682–691.

[Per+16] Emilija Perković, Johannes Textor, Markus Kalisch, and Marloes Maathuis. *Complete graphical characterization and construction of adjustment sets in Markov equivalence classes of ancestral graphs*. Tech. rep. 1606.06904. preprint, `arxiv.org/abs/1606.06903`. arXiv: 2016.

[Per+18] Emilija Perković, Johannes Textor, Markus Kalisch, and Marloes Maathuis. "Complete graphical characterization and construction of adjustment sets in Markov equivalence classes of ancestral graphs". In: *Journal of Machine Learning Research* 18.220 (2018), pp. 1–62.

[PKM17] Emilija Perković, Markus Kalisch, and Marloes Maathuis. "Interpreting and using CPDAGs with background knowledge". In: *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*. Available at `arxiv.org/abs/1707.02171`. 2017, .

[RGL08] Kenneth J. Rothman, Sander Greenland, and Timothy L. Lash. *Modern Epidemiology*. Wolters Kluwer, 2008. ISBN: 0781755646.

[RS02] Thomas Richardson and Peter Spirtes. "Ancestral Graph Markov Models". In: *Annals of Statistics* 30 (2002), pp. 927–1223.

[RTL76] Donald J. Rose, R. Endre Tarjan, and George S. Lueker. "Algorithmic aspects of vertex elimination on graphs". In: *SIAM J. Comput.* 5.2 (1976), pp. 266–283.

[SGS01] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. 2nd. 2001. ISBN: 9780262194402.

[Sha98] Ross D. Shachter. "Bayes-Ball: The Rational Pastime". In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1998, pp. 480–487.

[SP06a] Ilya Shpitser and Judea Pearl. "Identification of Conditional Interventional Distributions". In: *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2006, pp. 437–444.

[SP06b] Ilya Shpitser and Judea Pearl. "Identification of joint interventional distributions in recursive semi-Markovian causal models". In: *Proceedings of the 21st National Conference on Artificial Intelligence*. Vol. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2006, pp. 1219–1226.

[SP78] Y. Shiloach and Y. Perl. "Finding two disjoint paths between two pairs of vertices in a graph". In: *Journal of the ACM (JACM)* 25.1 (1978), pp. 1–9.

[SVR10] Ilya Shpitser, Tyler VanderWeele, and James Robins. “On the Validity of Covariate Adjustment for Estimating Causal Effects”. In: *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2010, pp. 527–536.

[Tak10] Ken Takata. “Space-Optimal, Backtracking Algorithms to List the Minimal Vertex Separators of a Graph”. In: *Discrete Applied Mathematics* 158 (2010), pp. 1660–1667.

[Tex+16] Johannes Textor, Benito van der Zander, Mark S Gilthorpe, Maciej Liśkiewicz, and George TH Ellison. “Robust causal inference using directed acyclic graphs: the R package ‘dagitty’”. In: *International Journal of Epidemiology* 45.6 (2016), pp. 1887–1894.

[TK17] Santtu Tikka and Juha Karvanen. “Identifying Causal Effects with the R Package causaleffect”. In: *Journal of Statistical Software* 76.12 (2017), .

[TL11] Johannes Textor and Maciej Liśkiewicz. “Adjustment Criteria in Causal Diagrams: An Algorithmic Perspective”. In: *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2011, pp. 681–688.

[TPP98] Jin Tian, Azaria Paz, and Judea Pearl. *Finding Minimal D-separators*. Tech. rep. R-254. University of California, Los Angeles, 1998.

[Van09] Tyler J. VanderWeele. “On the relative nature of overadjustment and unnecessary adjustment”. In: *Epidemiology* 20.4 (July 2009), pp. 496–499.

[VP90] Thomas Verma and Judea Pearl. “Equivalence and synthesis of causal models”. In: *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence*. Elsevier, 1990, pp. 255–270.

[VP92] Thomas Verma and Judea Pearl. “An algorithm for deciding if a set of observed independencies has a causal explanation”. In: *Proceedings of the 8th Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1992, pp. 323–330.

[Wer11] Nanny Wermuth. “Probability distributions with summary graph structure”. In: *Bernoulli* 17.3 (Aug. 2011), pp. 845–879.

[Wri34] Sewall Wright. “The method of path coefficients”. In: *The Annals of Mathematical Statistics* 5.3 (1934), pp. 161–215.

[Zha08] Jiji Zhang. “Causal Reasoning with Ancestral Graphs”. In: *Journal of Machine Learning Research* 9 (2008), pp. 1437–1474.

[ZL16a] Benito van der Zander and Maciej Liśkiewicz. “On Searching for Generalized Instrumental Variables.” In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2016, pp. 1214–1222.

[ZL16b] Benito van der Zander and Maciej Liśkiewicz. “Separators and Adjustment Sets in Markov Equivalent DAGs”. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. 2016, pp. 3315–3321.

[ZL19] Benito van der Zander and Maciej Liśkiewicz. “Finding Minimal d-separators in Linear Time and Applications”. In: *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2019.

[ZLT14] Benito van der Zander, Maciej Liśkiewicz, and Johannes Textor. "Constructing Separators and Adjustment Sets in Ancestral Graphs". In: *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*. (**IBM Best Student Paper Award**). AUAI Press, 2014, pp. 907–916.

[ZLT19] Benito van der Zander, Maciej Liśkiewicz, and Johannes Textor. "Separators and Adjustment Sets in Causal Graphs: Complete Criteria and an Algorithmic Framework". In: *Artificial Intelligence* 270 (2019), pp. 1–40.

[ZTL15] Benito van der Zander, Johannes Textor, and Maciej Liśkiewicz. "Efficiently Finding Conditional Instruments for Causal Inference". In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. 2015, pp. 3243–3249.

[ZTL19] Benito van der Zander, Johannes Textor, and Maciej Liśkiewicz. "Graphical Methods for Finding Instrumental Variables". In: *Proceedings of the 17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*. University of Twente, 2019, pp. 135–138.

A Further experimental results

Tables A.1 and A.2 in this section show the results of versions of the experiments presented in Section 5.2 in Tables 5.1 and 5.2 in which the parameter controlling the number of unobserved variables is set to 0.25 or 0.5.

		$l=2$			$l=5$			$l=10$			$l=20$		
n	k	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**
10	1	8235	8235	9901	4772	4772	8938	2452	2452	7443	2373	2373	7469
10	2	4307	4840	8035	528	1120	3545	0	349	1977	0	379	2004
10	3	1603	2353	5173	36	313	1168	0	96	591	0	107	594
10	5	184	823	1700	0	99	204	0	44	76	0	36	82
25	1	9306	9306	9978	7019	7019	9507	3549	3549	8141	1645	1645	6686
25	2	7312	7466	9489	2132	2490	5757	310	485	2815	24	126	1862
25	3	4863	5250	8101	416	718	2668	10	65	925	0	14	523
25	5	1466	2060	4255	10	65	449	0	2	99	0	0	38
50	1	9668	9668	9993	8075	8075	9763	4408	4408	8517	1927	1927	7024
50	2	8555	8600	9814	4013	4222	7274	639	777	3614	92	145	2145
50	5	3727	4167	6835	86	174	1034	1	1	158	0	0	81
50	7	1449	1946	3888	4	22	233	0	0	14	0	0	7
100	1	9818	9818	9997	8886	8886	9879	5158	5158	8833	2173	2173	7271
100	2	9341	9354	9951	5742	5871	8400	1013	1084	4462	167	209	2564
100	5	6215	6404	8637	443	595	2122	3	11	340	0	0	96
100	10	1453	1813	3490	0	1	78	0	0	1	0	0	1
250	1	9917	9917	10000	9559	9559	9964	6033	6033	9208	2558	2558	7691
250	2	9712	9712	9990	7840	7888	9362	1717	1764	5611	216	236	3038
250	5	8293	8343	9669	2015	2192	4569	9	18	598	0	0	158
250	15	1728	2014	3676	0	1	27	0	0	1	0	0	0
250	25	85	164	361	0	0	0	0	0	0	0	0	0
500	1	9968	9968	10000	9765	9765	9986	6684	6684	9455	2667	2667	7888
500	2	9864	9866	9997	8920	8933	9762	2304	2329	6352	303	314	3414
500	5	9162	9178	9910	4207	4343	6662	46	50	955	0	0	186
500	22	1533	1774	3148	0	0	7	0	0	0	0	0	0
500	50	0	3	10	0	0	0	0	0	0	0	0	0
1000	1	9984	9984	10000	9903	9903	9997	7266	7266	9615	2831	2831	8086
1000	2	9926	9926	10000	9490	9491	9902	3261	3278	7194	348	350	3757
1000	5	9599	9602	9984	6613	6703	8370	75	81	1486	0	0	261
1000	32	1413	1588	2801	0	0	1	0	0	0	0	0	0
1000	100	0	0	0	0	0	0	0	0	0	0	0	0
2000	1	9994	9994	10000	9945	9945	10000	7924	7924	9773	3117	3117	8322
2000	2	9963	9963	10000	9809	9812	9985	4140	4150	7842	452	456	4191
2000	5	9800	9802	9992	8273	8316	9403	210	217	2140	0	0	356
2000	45	1541	1728	2840	0	0	0	0	0	0	0	0	0
2000	200	0	0	0	0	0	0	0	0	0	0	0	0

Table A.1: Numbers of instances for $P(\textit{unobserved}) = 0.25$ that are identifiable by use of BC, CBC, CBC$^+$ (as defined in Section 5.2.2). We did not run the IDC algorithm on these data due to its high time complexity. Gray cells highlight where the CBC was able to identify at least 400 more graphs than the BC.

		$l=2$			$l=5$			$l=10$			$l=20$		
n	k	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**	**BC**	**CBC**	**CBC$^+$**
10	1	7418	7418	9799	3102	3102	8085	1500	1500	6515	1520	1520	6537
10	2	3289	3795	7602	303	649	3029	0	223	1910	0	251	1878
10	3	1000	1575	4512	13	138	1038	0	57	549	0	43	539
10	5	165	822	1684	0	87	220	0	39	85	0	47	76
25	1	8912	8912	9960	5107	5107	9106	1920	1920	7224	898	898	5913
25	2	6413	6555	9258	1115	1346	5056	154	249	2592	9	77	1803
25	3	3665	4060	7595	182	339	2326	6	24	880	0	7	521
25	5	820	1241	3591	1	20	419	0	2	86	0	0	40
50	1	9429	9429	9983	6206	6206	9487	2454	2454	7831	1004	1004	6414
50	2	7928	7991	9738	2141	2272	6516	260	334	3353	43	76	2127
50	5	2484	2835	6082	21	52	921	0	1	198	0	0	64
50	7	735	1041	3164	0	1	205	0	0	20	0	0	6
100	1	9725	9725	9995	7032	7032	9658	2952	2952	8281	1154	1154	6626
100	2	8858	8882	9930	3198	3285	7674	403	444	4069	74	85	2422
100	5	4828	5052	8211	89	129	1791	1	1	277	0	0	83
100	10	619	828	2793	0	0	90	0	0	2	0	0	0
250	1	9876	9876	9999	8259	8259	9908	3539	3539	8767	1314	1314	7069
250	2	9542	9546	9979	5055	5097	8865	575	591	5085	83	93	2925
250	5	7423	7498	9502	422	469	3608	1	1	613	0	0	155
250	15	711	922	2833	0	0	37	0	0	0	0	0	0
250	25	12	25	243	0	0	0	0	0	0	0	0	0
500	1	9937	9937	10000	8992	8992	9957	4038	4038	9062	1336	1336	7354
500	2	9779	9781	9998	6569	6587	9369	791	802	5748	98	100	3210
500	5	8625	8641	9852	1162	1231	5318	2	3	925	0	0	195
500	22	572	685	2245	0	0	3	0	0	0	0	0	0
500	50	0	1	3	0	0	0	0	0	0	0	0	0
1000	1	9972	9972	10000	9420	9420	9985	4487	4487	9314	1525	1525	7565
1000	2	9865	9865	9999	7872	7881	9746	1086	1094	6475	96	98	3660
1000	5	9328	9335	9957	2623	2683	7140	4	5	1461	0	0	253
1000	32	475	548	1910	0	0	0	0	0	0	0	0	0
1000	100	0	0	0	0	0	0	0	0	0	0	0	1
2000	1	9985	9985	10000	9715	9715	9992	5059	5059	9491	1693	1693	7799
2000	2	9949	9949	10000	8823	8828	9937	1500	1503	7201	122	122	3905
2000	5	9624	9626	9994	4614	4649	8529	19	19	2066	0	0	345
2000	45	467	524	1853	0	0	0	0	0	0	0	0	0
2000	200	0	0	0	0	0	0	0	0	0	0	0	0

Table A.2: Numbers of instances for $P(\textit{unobserved}) = 0.5$ that are identifiable by use of BC, CBC, CBC$^+$ (as defined in Section 5.2.2). We did not run the IDC algorithm on these data due to its high time complexity. Gray cells highlight where the CBC was able to identify at least 400 more graphs than the BC.

B Further Classes of Graphical Models

In this appendix, we mention some related classes of graphical models for sake of completeness that we have not considered further in this thesis.

Undirected graphs. An undirected graph only contains undirected edges. A probability distribution P is compatible with an undirected graph if P factorizes as

$$P(\mathbf{v}) = \frac{1}{N} \prod_{i=1} P_i(\mathbf{w}_i),$$

where $\{\mathbf{W}_i\}$ are all maximal cliques in the graph and N is a normalization constant. Variables in an undirected graph are conditionally independent given a set $\mathbf{Z}$ if all paths between them are blocked by $\mathbf{Z}$, i.e., contain a node of $\mathbf{Z}$. This kind of separation is simpler than d-separation, which can be seen as an advantage of undirected graphs.

Partial ancestral graphs (PAGs). Like a CPDAG encodes a Markov equivalence class of DAGs, a PAG encodes a Markov equivalence class of MAGs [Zha08]. PAGs can contain four different types of edges $\rightarrow$, $\leftrightarrow$, $\circ\!\!-\!\!\circ$ and $\circ\!\!\rightarrow$. Each edge is considered to be marked by either an arrowhead ($>$), tail ($-$), or circle ($\circ$) on both sides. An arrowhead (tail) means the corresponding edge in each graph of the Markov equivalence class has an arrowhead (tail) at this side, while a circle means both arrowheads and tails occur[1].

Maximal PDAGs. Maximal PDAGs represent a subset of a Markov equivalence class of DAGs using directed and undirected edges [PKM17]. Maximal PDAGs have the same purpose as (restricted) chain graphs, but are more general because they allow semi-directed cycles, so they can represent subsets that cannot be represented by chain graphs. However, semi-directed cycles make it impossible to decide conditional independence relations with d-separation, so [PKM17] define more complex separation rules, which lead to slower algorithms. Particularly, the fastest known method of finding non-causal definite status paths in a maximal PDAG is based on constructing a represented DAG in $\mathcal{O}(nm)$, while we can find such a path in RCGs in $\mathcal{O}(n+m)$ (see Chapter 4).

[1] With three possible marks on two sides, PAGs could have nine different types of edges. Nevertheless, $-\!\!-$, $\circ\!\!-$, and $-\!\!\circ$ do not occur because [Zha08] only considers MAGs without undirected edges. $\leftarrow$ and $\leftarrow\!\!\circ$ are not distinguished from their mirrored counterparts.

Summary graphs. Summary graphs allow $\rightarrow$, —, and -- edges [Wer11], whereby -- edges are equivalent to $\leftrightarrow$ edges of AGs, so separation in summary graphs behaves identical to m-separation of AGs [RS02]. Summary graphs do not allow directed cycles, but allow both a $\rightarrow$ and -- edge between the same pair of nodes.

MC graphs. Like AGs, MC graphs were developed to represent the marginalization (M) and conditioning (C) of a DAG [Kos02]. MC graphs allow edges $\rightarrow$, —, and $\leftrightarrow$. Between a pair of nodes any combination of these edges might occur (i.e., up to four edges between a pair), and a node can have an undirected edge to itself as a self-loop. Nevertheless, d-separation in MC graphs is identical to m-separation in AGs [RS02].

MC graphs are more general than summary graphs which are more general than AGs, i.e., AGs $\subset$ summary graphs $\subset$ MC graphs. However, this makes them them too large for their purpose of representing marginalization and conditioning [RS02]. There exist MC graphs that cannot be obtained by marginalizing and conditioning a directed (cyclic or acyclic) graph [Kos02; RS02], so their statistic interpretation is dubious.

Inducing path graphs (IPGs). Inducing path graphs extend DAGs with bidirected edges representing latent nodes [VP90; SGS01, chapter 2 and 6]. [Zha08] shows that IPGs can be characterized as a mixed graph with edges $\rightarrow$ and $\leftrightarrow$ which contain no directed cycle and are maximal, so IPGs are a slight generalization of MAGs permitting cycles $V_1 \leftrightarrow V_2 \overset{*}{\rightarrow} V_k \rightarrow V_1$.

Partially oriented inducing path graphs (POIPGs). Partially oriented inducing path graphs represent a Markov equivalent class of IPGs like PAGs represent a class of DAGs [SGS01, chapter 2 and 6]. They allow the four possible edge types of PAGs $\rightarrow$, $\leftrightarrow$, $\circ\!\!-\!\!\circ$, and $\circ\!\!\rightarrow$. Additionally, nodes can be marked as being non-colliders on every possible path in each represented IPG.

[Zha08] argues that MAGs and PAGs provide more qualitative causal information than IPGs and POIPGs since MAGs are more restricted than IPGs, so the Markov equivalence class contains fewer graphs and a PAG contains more edge marks than a POIPG.

Directed (cyclic) graphs. A serious limitation of DAG models is that they do not allow cycles or reciprocal interactions between variables [2]. When cycles are allowed in the directed graph, d-separation still corresponds to conditional independence relations in many models, but the probability distribution can no longer be factorized and there is no well-developed general theory of cyclic causal graphical models [SGS01, Section 12.1.2]. Nevertheless, such a theory is under active research [FM17; Bon+18].

[2] Even though cycles are so important in nature that the study of feedback loops has created an entire scientific field: cybernetics.

C Essential Paths and Nearest Separators

In [ZTL15], we have given a greedy $\mathcal{O}(n^3)$ algorithm to find nearest separators using the moral graph. Since that runtime is relatively slow, we have searched for faster approaches. The first idea was to remove the need of the moral graph, which yields a $\mathcal{O}(n(n+m))$ algorithm. Although that algorithm is obsolete after we have discovered our $\mathcal{O}(n+m)$ algorithm in Proposition 6.13 [ZL19], we describe it in this appendix since it provides new insight into nearest separators.

Formally, we start by analyzing the paths between Y and Z that need to be blocked in order to d-separate Y and Z, paths which we will call essential paths:

Definition C.1. *Given nodes Y and Z in $\mathbf{V}$, we call a tuple of paths $(\pi_i) = (\pi_1, \dots, \pi_k)$ with nodes $\mathbf{W} = (W_1, \dots, W_k)$ an* essential path bundle *relative to Y and Z if* *essential path bundle*

- *Each essential path π_i has Y as a start node and Z as an end node,*
- *W_i is the first non-collider on π_i in $\mathbf{O}$, and*
- *π_i is active given $\{W_1, \dots, W_{i-1}\}$.*

We call the $(W_1, \dots, W_k)$ the *nearest nodes* of (π_i). The essential path bundle can be empty, so $()$ with nearest nodes $\emptyset$ is an essential path bundle relative to all node pairs. However, the nearest nodes need to exist on the paths; thus, an essential path bundle must not contain paths that have no observed non-collider like $Y \to Z$, $Y \to \leftarrow Z$, or $Y \leftarrow U \to Z$ with $U \notin \mathbf{O}$. No path can occur more than once in the bundle because if there was a $\pi_i = \pi_{i'}$ with $i < i'$, then W_i would block $\pi_{i'}$. *nearest nodes*

Since we are interested in nearest separators, we will focus on the study of separating essential path bundles.

Definition C.2. *Given nodes Y and Z in $\mathbf{V}$, we call a tuple of paths $(\pi_i) = (\pi_1, \dots, \pi_k)$ with nearest nodes $\mathbf{W}$ a* separating essential path bundle *relative to Y and Z if* *separating essential path bundle*

- *(π_i) is an essential path bundle relative to Y and Z, and*
- *$\mathbf{W}$ d-separates Y and Z.*

Separating essential path bundles are also maximal in the sense that no path with observed non-colliders can be added to the tuple. Generally an essential path bundle is maximal if it is either separating or none of the paths active given the nearest nodes contains an observed

non-collider. It is not hard to see that any maximal essential path bundle relative to Y and Z is separating if and only if Y and Z are d-separable.

A well-known property of separators is that nodes that can be d-separated by any set can also be d-separated by subsets of their ancestors. A similar property holds for essential path bundles:

Lemma C.3. *Given an essential path bundle (π_i) relative to Y and Z, every node on a path π_i is in An$(Y \cup Z)$.*

Proof. All nodes on a path opened by a set $\mathbf{W}'$ are ancestors of the start/end nodes or of $\mathbf{W}'$ as shown in Lemma 3.13. π_1 is active given $\emptyset$, so it contains only ancestors of Y and Z. All other paths π_i only contains those ancestors and ancestors of an earlier path. □

Lemma C.4. *If there exists an essential path bundle relative to Y and Z that contains a descendant of Y other than Y itself, then*

- *Z is a descendant of Y, and*
- *Every set separating Y and Z contains a descendant of Y.*

Proof. Let V be the descendant in *De*$(Y) \setminus Y$. From Lemma C.3, we know $V \in$ *An*(Y, Z) and in a DAG a node cannot be a both descendant and ancestor, so V is an ancestor of Z. Hence, there is a causal path from Y to V and a causal path from V to Z. Thus, there exists a causal path from Y to Z that can only be blocked by a descendant of Y, and every separating set must contain a descendant of Y. □

The nearest nodes of a separating essential path bundle form a nearest separator as the following lemma shows:

Lemma C.5. *Let (π_i) be an essential path bundle relative to Y and Z, and let W_i be one of its nearest nodes $\mathbf{W} = (W_1, \ldots, W_k)$. If a set $\mathbf{W}' \subseteq \mathbf{O} \setminus \{W_i, Y, Z\}$ does not block every path π between W_i and Z, $\mathbf{W}'$ does not d-separate Y and Z.*

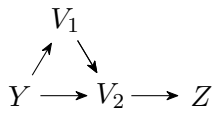
Proof. No node on $\pi_i[Y \overset{*}{\sim} W_i]$ besides colliders, Y, and W_i is observed, so $\mathbf{W}'$ contains no non-collider of $\pi_i[Y \overset{*}{\sim} W_i]$.

Every collider on $\pi_i[Y \overset{*}{\sim} W_i]$ as well as the node W_i is an ancestor of Y or Z due to Lemma C.3. Thus, according to Lemma 3.12, $\mathbf{W}'$ does not d-separate Y and Z.

□

On the other hand a set can be a nearest separator without consisting of nearest nodes of an essential path bundle. A simple example is the DAG $Y \to W \quad Z$, where Z is not adjacent to any node and $\mathbf{W} = \{W\}$ is a nearest separator, because Y and Z are always d-separated and X in condition (ii) can only be Z, which is always d-connected to itself. Yet, W cannot belong to any essential path bundle.

A separating essential path bundle does not yield a minimal nearest separator. For example, in a DAG



the paths $(Y \to V_1 \to V_2 \to Z, Y \to V_2 \to Z)$ form a separating essential path bundle with nearest nodes $\{V_1, V_2\}$. A minimal nearest separator can be obtained by the separating essential path bundle $(Y \to V_2 \to Z)$ with nearest nodes $\{V_2\}$.

The theory of essential path bundles yields directly the following algorithm to compute nearest separators:

function FIND-NEAREST-SEPARATOR-SLOW($\mathcal{G}, Y, Z$)
  $\mathbf{W} := \emptyset$
  while $\exists \pi :=$ a path from Y to Z active given $\mathbf{W}$ **do**
    if $\exists V \in \pi$: $V \in \mathbf{O} \setminus \{Y, Z\}$ and V is not a collider on π **then**
      $\mathbf{W} := \mathbf{W} \cup \{\text{first such } V \text{ on } \pi\}$
    else
      return $\perp$
  return $\mathbf{W}$

Proposition C.6. *The algorithm* FIND-NEAREST-SEPARATOR-SLOW *returns a nearest separator* $\mathbf{W} \subseteq An(Y \cup Z)$ *if* Y *and* Z *are separable in* $\mathcal{G}$*; otherwise it returns* $\perp$*. Moreover, if* Y *and* Z *can be separated in* $\mathcal{G}$ *by a set that does not contain a descendant of* Y*, then* $\mathbf{W} \subseteq An(Y \cup Z) \setminus De(Y)$*. The runtime of the algorithm is* $\mathcal{O}(n(n+m))$*.*

Proof. If the algorithm returns a set $\mathbf{W}$, it fulfills $(Z \perp\!\!\!\perp Y \mid \mathbf{W})_{\mathcal{G}}$ because it only returns a set if the while loop finds no d-path between Y and Z. All paths found by the algorithm form a separating essential path bundle, and the set $\mathbf{W}$ yields from their nearest nodes. So the algorithm returns a nearest separator. From Lemma C.3 and Lemma C.4, it follows that $\mathbf{W} \subseteq An(Y \cup Z)$ respectively $\mathbf{W} \subseteq An(Y \cup Z) \setminus De(Y)$.

If Y and Z are d-separable and the algorithm does not return a set, there exists a path π whose colliders have been opened by $\mathbf{W}$ and that does not contain an observed non-collider. Yet there is a separator $\mathbf{W}'$ that does not open all colliders on π. Let C be the last collider unopened by $\mathbf{W}'$ and $W_i \in \mathbf{W}$ the node opening it. Then the path $W_i \overset{*}{\leftarrow} \pi[C \overset{*}{\sim} Z]$ is open given $\mathbf{W}'$, so according to Lemma C.5, $\mathbf{W}'$ is not a separator.

The runtime is $\mathcal{O}(n(n+m))$ since a d-path can be found in $\mathcal{O}(m)$ steps and at most $\mathcal{O}(n)$ nodes can be added to $\mathbf{W}$. □

D Listings

List of Tables

List of Figures

List of Algorithms

Index

E Curriculum Vitae

Persönliche Daten

Benito van der Zander
geboren 17.06.1989 in Düsseldorf

Wissenschaftliche Ausbildung

Rheinisch-Westfälische Technische Hochschule Aachen

Master of Science in Informatik, Dezember, 2011

Note: 1.5
Thema der Arbeit: “Open Street SLAM: Combining Visual SLAM with Cadastral Maps”

Heinrich-Heine-Universität Düsseldorf

Bachelor of Science in Informatik, Oktober, 2009

Note: 1.5 mit Auszeichnung
Thema der Arbeit: “Eigenschaften eines Algorithmus zur Zuordnung von nicht eindeutigen Sende- und Empfangsereignissen in Ereignisprotokollen”

Studium erfolgte parallel zur Schule ab 2002 (mit einer Vorlesung je Semester)

Arbeitserfahrungen

Universität zu Lübeck

Wissenschaftlicher Mitarbeiter **seit 2016**
Forschung an DFG-Projekt “Kausalität”.

Wissenschaftliche Hilfskraft mit Abschluss **2013 – 2016**

Rheinisch-Westfälische Technische Hochschule Aachen

Wissenschaftliche Hilfskraft **Anfang 2012**

Wissenschaftliche Hilfskraft **2010 – 2011**

Heinrich-Heine-Universität Düsseldorf

Wissenschaftliche Hilfskraft **2008 – 2009**

Bundeswettbewerb Informatik **2008 – 2016**

Halbjährliche Bewertung von Einsendungen

AUSZEICHNUNGEN

Ausgezeichnete Publikationen

IBM Best Student Paper Award, zur Publikation "Constructing Separators and Adjustment Sets in Ancestral Graphs" (UAI 2014, [ZLT14]) **2014**

Balisage First Place Student Award zur Publikation "Extending XQuery with pattern matching over XML, HTML and JSON, and its usage for data mining" (Balisage 2014) **2014**

Stipendien

Lübecker Graduate School **2013 – 2016**

Studienstiftung des deutschen Volkes **2008 – 2011**

Central European Olympiad of Informatics **2006 – 2008**

2008: Silber Medaille
2007: Silber Medaille
2006: Bronze Medaille

Baltic Olympiad of Informatics **2006 – 2008**

2008: Bronze Medaille
2007: Silber Medaille
2006: Bronze Medaille

Bundeswettbewerb Informatik **2002 – 2006**

2006: Bundessieger und Sonderpreise: "beste Einzelidee" und "beste Gruppenleistung"
2005: Preisträger und Sonderpreis: "beste Einzelidee"
2004: Preisträger und Sonderpreise: "beste Einzelidee" und "bester jüngster Teilnehmer"

Jugend Forscht / Schüler Experimentieren **2001, 2008**

2008: Teilnahme mit "VideLibri"
2001: Sonderpreis Software-System-Technik für ein Spiel; Teilnahme mit Matheprogramm

PUBLIKATIONEN (INKLUSIVE KOAUTORSCHAFT)

Benito van der Zander and Maciej Liśkiewicz, "Finding Minimal d-separators in Linear Time and Applications" in *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI 2019), Tel Aviv, Israel, 2019*, AUAI Press, 2019

Benito van der Zander, Johannes Textor, and Maciej Liśkiewicz, "Graphical Methods for Finding Instrumental Variables" in *Proceedings of the 17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2019), Twente, Netherlands, 2019*, pp. 135–138, University of Twente, 2019

Benito van der Zander, Maciej Liśkiewicz, and Johannes Textor, “Separators and Adjustment Sets in Causal Graphs: Complete Criteria and an Algorithmic Framework” in *Artificial Intelligence 270 (2019)*, pp. 1–40

Johannes Textor, Benito van der Zander, Mark S Gilthorpe, Maciej Liśkiewicz, and George TH Ellison, “Robust causal inference using directed acyclic graphs: the R package ‘dagitty’” in *International Journal of Epidemiology 45.6 (2016)*, pp. 1887–1894

Benito van der Zander, Maciej Liśkiewicz, “On Searching for Generalized Instrumental Variables” in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016), Cadiz, Spain, May 9–11, 2016*, pp. 1214–1222, JMLR Proceedings, 2016

Benito van der Zander, Maciej Liśkiewicz, “Separators and Adjustment Sets in Markov Equivalent DAGs” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016), Phoenix, Arizona, USA, February 12–17, 2016*, pp. 3315–3321 , AAAI Press, 2016

Benito van der Zander, Johannes Textor, Maciej Liśkiewicz, “Efficiently Finding Conditional Instruments for Causal Inference” in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25–31, 2015*, pp. 3243–3249, AAAI Press / International Joint Conferences on Artificial Intelligence, 2015

Benito van der Zander, “Extending XQuery with pattern matching over XML, HTML and JSON, and its usage for data mining” in *Proceedings of Balisage: The Markup Conference , North Bethesda, USA, August 2-5*,2014 **(Balisage First Place Student Award)**

Benito van der Zander, Johannes Textor, Maciej Liśkiewicz, “Constructing Separators and Adjustment Sets in Ancestral Graphs” in *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014* **(IBM Best Student Paper Award)***), Quebec, Canada*, pp. 907–916, AUAI Press, 2014

Georgios Floros, Benito van der Zander, Bastian Leibe, “OpenStreetSLAM: Global Vehicle Localization Using OpenStreetMaps” in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2013), Karlsruhe, Germany, May 6–10, 2013*, pp. 1054–1059, IEEE, 2013

Dennis Mitzel, Georgios Floros, Patrick Sudowe, Benito van der Zander, Bastian Leibe, “Real Time Vision Based Multi-Person Tracking for Mobile Robotics and Intelligent Vehicles” in *Proceedings of the 4th International Conference on Intelligent Robotics and Applications (ICIRA (2) 2011), Aachen, Germany, Dec 6–9, 2011*, pp. 105–115, Springer, 2011

Benito van der Zander, Egon Wanke, Wolfgang Kieß, Björn Scheuermann, “Brief announcement: complexity and solution of the send-receive correlation problem” in *Proceedings of the Twenty-Ninth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2010), Zurich, Switzerland, July 25–28, 2010*, pp. 122–123, ACM

OPEN-SOURCE SOFTWARE PROJEKTE

DAGitty (http://dagitty.net) **seit 2015**
Implementierung der Algorithmen meiner Dissertation in DAGitty, einer Webseite und einem R-Paket.

Xidel (http://xidel.sourceforge.net) **seit 2012**
Ein XQuery-Interpreter für Data Mining von Webseiten, verfügbar als Kommandozeilenprogramm, Webservice und Library. Xidel implementiert den W3C XPath/XQuery 3.1-Standard, eine funktionale Programmiersprache mit fast 200 Standardfunktionen und 50 primitiven Typen. Xidel hat zusätzliche Erweiterungen für EXPath, JSONiq, CSS-Selektoren und XML/HTML-Patternmatching.

TeXstudio (http://www.texstudio.org) **seit 2009**
Ein LaTeX-Editor mit interaktiver Rechtschreib/Grammatikprüfung für Windows, Linux und macOS, entwickelt in C++ und Qt. Ursprünglich von mir als Fork von Texmaker gegründet, ist es nun ein unabhängiges Projekt mit über einer Millionen Downloads. Ausgezeichnet als **SourceForges "Project of the Month"** in Oktober 2015 und August 2013.

VideLibri (http://www.videlibri.de) **seit 2006**
VideLibri greift auf die Webkataloge von über 300 Bibliotheken zu, um die vom Benutzer ausgeliehenen Bücher anzuzeigen, die Leihfrist zu verlängern und im Katalog zu suchen. Es läuft auf Windows, Linux und Android. Zum Auslesen der Webseiten ohne verfügbare API wurde mittels XQuery Schnittstellen zu 20 zueinander inkompatiblen Katalogsystemen erstellt.