

Separators and Adjustment Sets in Causal Graphs: Complete Criteria and an Algorithmic Framework *

Benito van der Zander

Institute for Theoretical Computer Science, Universität zu Lübeck, Germany

Maciej Liśkiewicz

Institute for Theoretical Computer Science, Universität zu Lübeck, Germany

Johannes Textor

Institute for Computing and Information Sciences, Radboud University Nijmegen, Nijmegen, The Netherlands

Abstract:

Principled reasoning about the identifiability of causal effects from non-experimental data is an important application of graphical causal models. We present an algorithmic framework for efficiently testing, constructing, and enumerating m -separators in ancestral graphs (AGs), a class of graphical causal models that can represent uncertainty about the presence of latent confounders. Furthermore, we prove a reduction from causal effect identification by covariate adjustment to m -separation in a subgraph for directed acyclic graphs (DAGs) and maximal ancestral graphs (MAGs). Jointly, these results yield constructive criteria that characterize all adjustment sets as well as all minimal and minimum adjustment sets for identification of a desired causal effect with multivariate exposures and outcomes in the presence of latent confounding. Our results extend several existing solutions for special cases of these problems. Our efficient algorithms allowed us to empirically quantify the identifiability gap between covariate adjustment and the do-calculus in random DAGs, covering a wide range of scenarios. Implementations of our algorithms are provided in the R package DAGITTY.

Keywords: Causal inference, Covariate adjustment, Ancestral graphs, d-separation, m -separation, Complexity, Bayesian network, Knowledge representation

Contents

1	Introduction	2
2	Preliminaries	4
2.1	General backgrounds	5
2.2	Identification via covariate adjustment	6
3	Algorithms for m-separation in ancestral graphs	7
3.1	Linear-time algorithms for testing and finding m -separators	8
3.2	Polynomial time algorithms for minimal and minimum m -separators	10
3.3	The hardness of strong-minimality	14

*This is a revised and extended version of preliminary work presented at the 27th [1] and 30th [2] conferences on Uncertainty in Artificial Intelligence (UAI).

3.4	Algorithms for minimal m -separators in sparse graphs	15
3.5	Algorithms for enumerating all m -separators	16
4	Empirical analysis of DAG consistency testing	17
4.1	Simulation setup	18
4.2	Empirical results	18
5	Adjustment in DAGs	19
5.1	Constructive back-door criterion	19
5.2	CBC vs Pearl's back-door criterion	21
6	Algorithms for testing and computing adjustment sets in DAGs	22
7	Extending the CBC	24
7.1	Identification by plain formulas	25
7.2	Identification by generalized parent adjustment	25
7.3	Identification when \mathbf{X} and \mathbf{Y} partition \mathbf{V}	26
8	Empirical analysis of identifiability by adjustment	27
8.1	Instance generation	28
8.2	Algorithms	28
8.3	Results	29
9	Adjustment in MAGs	37
9.1	Adjustment amenability	38
9.2	Adjustment criterion for MAGs	38
9.3	Adjustment set construction	41
10	Discussion	41
11	Acknowledgment	42
12	References	42
13	Appendix	44
13.1	Auxiliary lemmas and proofs	44
13.2	Further experimental results	46
13.3	Do-calculus	46

1 Introduction

Graphical causal models are popular tools for reasoning about assumed causal relationships between random variables and their implications [3, 4]. Such models represent the mechanisms that are assumed to generate an observed joint probability distribution. By analyzing the model structure, one can not only determine conditionally independence relationships, but also deduce which variables introduce potential bias (confounding) when the causal effect of certain exposure variables (\mathbf{X}) on outcome variables (\mathbf{Y}) is to be determined. Confounding is a notorious issue when performing causal inference from observational rather than experimental data, which is often necessary in population research within Epidemiology [5], the Social Sciences [4], or Econometrics.

A traditional approach to address confounding is to simply adjust (e.g., stratify) for as many covariates as possible that temporally precede the exposure variable (pre-exposure variables) [6, 7]. This practice has roots in the analysis of data from randomized trials, in which confounding can only occur by chance. For observational data, however,

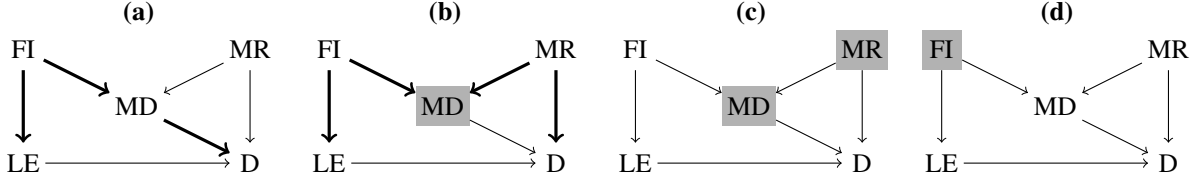


Figure 1: Causal diagram [5, Chapter 12] describing the effect of low education (LE) on diabetes risk (D) with the covariates family income (FI), mother’s genetic risk to develop diabetes (MR), and mother’s diabetes (MD). The unadjusted estimate (a) is biased due to the common ancestor FI – bias “flows” via the biasing path $LE \leftarrow FI \rightarrow MD \rightarrow D$ (bold edges). Adjustment for MD (b) blocks this biasing path, but opens a new one because FI and MR become correlated. The minimal adjustments {MD, MR} (c) and {FI} (d) close all biasing paths. Note that, if both FI and MR were unmeasured, it would be impossible to know whether adjustment for only MD would increase or reduce bias. This shows that, unlike for experimental data, conditioning on pre-exposure covariates can be both beneficial and detrimental in observational data.

it can be easily shown using graphical causal models that conditioning on one variable can turn a previously non-confounding variable into a confounder (Figure 1). If that new variable has not been or cannot be measured (e.g., if we could not observe variable MR in Figure 1), then conditioning on a pre-exposure variable could have a detrimental effect. Such examples show that it is impossible to decide which covariate sets one should adjust for to address confounding without at least some knowledge of the causal relationships between the involved covariates. To our knowledge, graphical causal models are currently the only causal inference framework in which it is possible to state a set of causal assumptions and derive from those assumptions a conclusive answer to the question whether, and how, a causal effect of interest is identifiable via covariate adjustment.

Covariate adjustment is not complete for identification; other methods like the front-door criterion or do-calculus [3] can permit identification even if covariate adjustment is impossible. In practice, adjustment is however often preferred to such alternatives because its statistical properties are well understood, giving access to useful methodology like robust estimators and confidence intervals. In contrast, knowledge about the statistical properties of e.g. front-door estimation is still lacking [8, 9]¹.

A famous result that characterizes valid sets of variables for covariate adjustment in a given graphical model is the back-door criterion by Pearl [3]. We call such sets of variables *adjustment sets* or simply *adjustments*. Although the back-door criterion is *sound*, meaning that sets fulfilling the criterion are indeed valid adjustments, it is not *complete* – not all valid adjustments fulfill the criterion. It is even possible that none of the valid adjustments in a causal model fulfill the back-door criterion, which would lead a user to incorrectly decide that a different method is required to identify a causal effect of interest. The first sound and complete criterion that characterizes adjustments was given by Shpitser and colleagues [10] for DAGs. This criterion achieves completeness by weakening the simple but overly restrictive condition in the back-door criterion that adjustment sets must not contain any descendants of the exposure variable; such variables can be allowed provided they do not block causal paths or create new biasing paths. However, a characterization of adjustment sets in terms of blocked paths does not yield a practical algorithm for adjustment set construction: Although enumerating all subsets of covariate variables and all paths would work, such a brute-force approach is only feasible in very small graphs. Thus, the practical relevance of covariate adjustment calls for algorithmically efficient methods that exhaustively characterize all available options. Such methods would ensure that the researcher can devise the best possible strategy to identify a given causal effect.

This paper provides efficient algorithms that provide exhaustive answers to the question: Given a causal graph \mathcal{G} , which covariates \mathbf{Z} can we adjust for to estimate the causal effect of the exposures \mathbf{X} on the outcomes \mathbf{Y} ? We address many variants of this question, such as requiring \mathbf{Z} to be minimal or minimum as well as imposing the constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ for given sets \mathbf{I}, \mathbf{R} . All our algorithms handle sets of multiple, possibly interrelated exposures \mathbf{X} and outcomes \mathbf{Y} , which is important in applications such as case-control studies that screen several putative causes of

¹ Quoting [8], “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.”

a disease [11]. The basis for our algorithms are theorems that reduce adjustment to d -separation or m -separation in a subgraph, combined with a generic, efficient and flexible algorithmic framework to find, verify, and enumerate separating sets. Our algorithms are guaranteed to find all valid adjustment sets for a given model with polynomial delay. Besides adjustment, the framework is potentially useful for other applications, as we illustrate by applying it to model checking. We also perform an extensive empirical evaluation of the power of covariate adjustment compared to other identification techniques in random DAGs.

In real-life applications, not all covariates are equally well suited for adjustment. Measuring some variables may be difficult or expensive, or could be affected by substantial measurement error. For instance, in Figure 1, measuring the variable MR (the genetic risk inherited from the mother) would require knowledge of the genetic variants that lead to a higher diabetes risk and availability of the genomes of the mothers of all study participants, but under these circumstances, it could be measured rather reliably. The family income, on the other hand, could be determined rather easily using a questionnaire, which however would likely suffer from substantial bias (e.g., people with high incomes could be less willing to report them). In such cases algorithms that return adjustment sets in which some (measurement or reliability) cost of all variables in the adjustment is minimized are potentially useful. For this reason, we provide variants to list only those sets that minimize a user-supplied cost function.

A model represented as a DAG assumes causal sufficiency, i.e. that every relevant variable is included in the model. If this is not the case and there are unknown latent variables, one can use a maximal ancestral graph (MAG) model that just specifies the ancestral relations between the variables [12]. An adjustment in a MAG remains valid when the MAG is extended by any number of additional latent variables as long as the ancestral relations and m -separating sets are preserved. This is an important feature because the assumption that all relevant confounders have been measured and included in a model is often unrealistic. We show that our entire algorithmic framework for computing adjustment sets can be generalized to MAGs, allowing practitioners to substantially relax the assumption of causal sufficiency that DAG-based analyses rely on.

This paper is structured as follows. Section 2 introduces notation and basic concepts. In Section 3, we present algorithms for verifying, constructing, and listing m -separating sets \mathbf{Z} in ancestral graphs. All algorithms handle the case of unconstrained separators, minimal and minimum separators as well as the additional constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ for given sets \mathbf{I}, \mathbf{R} . They subsume a number of earlier solutions for special cases of these problems, e.g. the Bayes-Ball algorithm for verification of d -separating sets [13] or the use of network flow calculations to find minimal d -separating sets in DAGs [14, 15]. Section 4 presents DAG consistency testing as a possible application of these algorithms beyond adjustment sets. In Section 5, we present a constructive back-door criterion (CBC) that reduces the construction of adjustment sets to the finding of separating sets in DAGs, explore possible variations of it and compare it to Pearl’s back-door criterion [3]. Section 6 explains how to apply the algorithms of Section 3 to the graphs resulting from the criterion of Section 5, which yields efficient polynomial time algorithms for verifying, constructing, and listing adjustment sets in DAGs. This combination leads to the first efficient implementation of the sound and complete adjustment criterion by Shpitser and colleagues [10]. Section 7 extends our criterion by addressing some cases where, even though covariate adjustment is not possible, it is still not necessary to invoke the do-calculus since the causal effect can be identified in another simple manner. In Section 8 we compare the results and performance of our adjustment criterion (with and without the extensions derived in Section 7) with Pearl’s back-door criterion as well as the IDC algorithm, which finds all identifiable causal effects, on randomly generated DAGs. Finally, in Section 9, we extend our constructive back-door criterion of Section 5 to MAGs, which generalizes the sound but incomplete adjustment criterion for MAGs without constraints by Maathuis and Colombo [16]. The proof techniques introduced in Sections 5 and 9 have been successfully applied to obtain complete and constructive adjustment criteria for several other classes of graphical causal models since the publication of our preliminary version [2], as summarized in Table 1.

2 Preliminaries

We denote sets by bold upper case letters (\mathbf{S}), and sometimes abbreviate singleton sets $\{S\}$ as S . Graphs are written calligraphically (\mathcal{G}), and variables in upper-case (X).

Graph class	First sound and complete criterion	First sound and complete constructive criterion
DAGs	Shpitser et al. [10]	this paper (conference version [2])
MAGs	this paper (conference version [2])	this paper (conference version [2])
CPDAGs	Perković et al. [17]	van der Zander and Liškiewicz [18]
PAGs	Perković et al. [17]	Perković et al. [19]
CGs	van der Zander and Liškiewicz [18]	van der Zander and Liškiewicz [18]
maximal PDAGs	Perković et al. [20]	Perković et al. [20]

Table 1: An overview of our results 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). A criterion is sound if it is only satisfied by adjustment sets. It is complete if every adjustment set satisfies it. It is constructive if it leads to an efficient algorithm for constructing an adjustment set. Paper [2] is the preliminary conference version of this work, and all subsequent results are based on the techniques developed in this paper.

2.1 General backgrounds

Mixed graphs and paths We consider mixed graphs $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ with nodes (vertices, variables) \mathbf{V} and directed ($A \rightarrow B$), undirected ($A - B$), and bidirected ($A \leftrightarrow B$) edges \mathbf{E} . Nodes linked by an edge are *adjacent* and *neighbors* of each other. A *walk* of length n is a node sequence V_1, \dots, V_{n+1} such that there exists an edge sequence E_1, E_2, \dots, E_n for which every edge E_i connects V_i, V_{i+1} . Then V_1 is called the *start node* and V_{n+1} the *end node* of the walk. A *path* is a walk in which no node occurs more than once. 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 path is *proper*. Given a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ and a node set \mathbf{V}' , the *induced subgraph* $\mathcal{G}_{\mathbf{V}'} = (\mathbf{V}', \mathbf{E}')$ contains the edges $\mathbf{E}' = \mathbf{E} \cap (\mathbf{V}' \times \mathbf{V}')$ from \mathcal{G} that are adjacent only to nodes in \mathbf{V}' . 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. Throughout, n stands for the number of nodes and m for the number of edges of a graph.

Ancestry A walk of the form $V_1 \rightarrow \dots \rightarrow V_n$ is *directed*, or *causal*. A non-causal walk is *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 . A graph is *acyclic* if no directed walk from a node to itself is longer than 0. All directed walks in an acyclic graph are paths. A walk is *anterior* if it were directed after replacing all edges $U - V$ by $U \rightarrow V$. If there is an anterior path from U to V , then U is called an *anterior* of V . All ancestors of V are anteriors of V . Every node is its own ancestor, descendant, and anterior. For a node set \mathbf{X} , the set of all of its ancestors is written as $An(\mathbf{X})$. The descendant and anterior sets $De(\mathbf{X}), Ant(\mathbf{X})$ are analogously defined. Also, we denote by $Pa(\mathbf{X}), Ch(\mathbf{X}), Ne(\mathbf{X})$, the set of parents (children, neighbors) of \mathbf{X} .

m -Separation A node V on a walk w is called a *collider* if two arrowheads of w meet at V , e.g., one possible collider is $U \leftrightarrow V \leftarrow Q$. There can be no collider if w is shorter than 2. Two nodes U, V are called *collider connected* if there is a path between them on which all nodes except U and V are colliders. Adjacent vertices are collider connected. Two nodes U, V are called *m -connected* by a set \mathbf{Z} if there is a path π between them on which every node that is a collider is in $An(\mathbf{Z})$ and every node that is not a collider is not in \mathbf{Z} . Then π is called an *m -connecting path*. The same definition can be stated simpler using walks: U, V are called *m -connected* by \mathbf{Z} if there is a walk between them on which all colliders and only colliders are in \mathbf{Z} . If U, V are *m -connected* by the empty set, we simply say they are *m -connected*. If U, V are not *m -connected* by \mathbf{Z} , we say that \mathbf{Z} *m -separates* them or *blocks* all paths between them. Two node sets \mathbf{X}, \mathbf{Y} are *m -separated* by \mathbf{Z} if all their nodes are pairwise *m -separated* by \mathbf{Z} . In DAGs, *m -separation* is equivalent to the well-known *d -separation* criterion [3].

Ancestral graphs and DAGs A mixed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is called an *ancestral graph* (AG) if the following two conditions hold: (1) For each edge $A \leftarrow B$ or $A \leftrightarrow B$, A is not an ancestor of B . (2) For each edge $A - B$, there are no edges $A \leftarrow C$, $A \leftrightarrow C$, $B \leftarrow C$ or $B \leftrightarrow C$. There can be at most one edge between two nodes in an AG [12]. Syntactically, all DAGs are AGs and all AGs containing only directed edges are DAGs. 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 turned into a MAG \mathcal{M} by adding bidirected edges between node pairs that cannot be m -separated, which preserves all m -separation relationships in the graph [12].

Graph transformations A DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is represented by a MAG $\mathcal{M} = \mathcal{G}[\mathbf{L}]_{\mathbf{S}}^{\mathbf{L}}$ 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 $V \notin \text{An}(U \cup \mathbf{S})$. The *canonical DAG* $C(\mathcal{M})$ 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 $C(\mathcal{M})[\mathbf{L}]_{\mathbf{S}}^{\mathbf{L}} = \mathcal{M}$ (see [12]).

The *augmented graph* $(\mathcal{G})^a$ of a certain AG \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} . Two node sets \mathbf{X} and \mathbf{Y} are m -separated by a node set \mathbf{Z} in \mathcal{G} if and only if \mathbf{Z} is an \mathbf{X} - \mathbf{Y} node cut in $(\mathcal{G}_{\text{An}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})})^a$ [12]. For DAGs the augmented graph is also called the moralized graph, so the construction of the augmented graph is termed *moralization*.

For any subset \mathbf{A} and \mathbf{B} of \mathbf{V} , by $\mathbf{A} \rightarrow \mathbf{B}$ we denote the set of all edges $A \rightarrow B$ 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 $\mathcal{G}_{\overline{\mathbf{X}}} = (\mathbf{V}, \mathbf{E} \setminus ((\mathbf{V} \rightarrow \mathbf{X}) \cup (\mathbf{V} \leftrightarrow \mathbf{X})))$. The removal of all edges leaving \mathbf{X} is written as $\mathcal{G}_{\underline{\mathbf{X}}} = (\mathbf{V}, \mathbf{E} \setminus ((\mathbf{X} \rightarrow \mathbf{V}) \cup (\mathbf{X} - \mathbf{V})))$. 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 subscript at the corresponding function without specifying the graph, e.g. $De_{\overline{\mathbf{X}}}$ or $An_{\underline{\mathbf{X}}}$. This notation is used for DAGs and MAGs. Note that for DAGs in each case only the first kind of edges needs to be removed as there are no \leftrightarrow or $-$ edges in DAGs.

2.2 Identification via covariate adjustment

Do-operator and identifiability A DAG \mathcal{G} encodes the factorization of a joint distribution P for the set of variables $\mathbf{V} = \{X_1, \dots, X_n\}$ as $P(\mathbf{v}) = \prod_{j=1}^n P(x_j | pa_j)$, where pa_j denotes a particular realization of the parent variables of X_j in \mathcal{G} . When interpreted causally, an edge $X_i \rightarrow X_j$ is taken to represent a direct causal effect of X_i on X_j . For disjoint $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$, the *(total) causal effect* of \mathbf{X} on \mathbf{Y} is $P(\mathbf{y} | do(\mathbf{x}))$ where $do(\mathbf{x})$ represents an intervention that sets $\mathbf{X} = \mathbf{x}$. In a DAG, this intervention corresponds to removing all edges into \mathbf{X} , disconnecting \mathbf{X} from its parents, i.e. constructing the graph $\mathcal{G}_{\overline{\mathbf{X}}}$. Given DAG \mathcal{G} and a joint probability density P for \mathbf{V} the post-intervention distribution can be expressed in a truncated factorization formula:

$$P(\mathbf{v} | do(\mathbf{x})) = \begin{cases} \prod_{X_j \in \mathbf{V} \setminus \mathbf{X}} P(x_j | pa_j) & \text{for } \mathbf{V} \text{ consistent with } \mathbf{x}, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

When all variables in \mathbf{V} are observed (we model this by setting $\mathbf{R} = \mathbf{V}$), the causal effect $P(\mathbf{y} | do(\mathbf{x}))$ of \mathbf{X} on \mathbf{Y} in a given graph \mathcal{G} can be determined uniquely from the pre-intervention distribution P using the factorization above. However, when some variables are unobserved, the question whether $P(\mathbf{y} | do(\mathbf{x}))$ is *identifiable*, i.e., if it can be computed from the pre-intervention distribution independent of the unknown quantities for the unobserved variables $\mathbf{V} \setminus \mathbf{R}$, becomes much more complex. For a formal definition of identifiability see [3, Section 3.2]. Figure 2 shows an example DAG \mathcal{G}_1 for which the causal effect of X on Y is not identifiable, and two DAGs \mathcal{G}_2 and \mathcal{G}_3 for which the effect is identifiable.

Adjustment sets The identification via covariate adjustment, studied in this paper, is defined for DAGs below and can be extended to MAGs in the usual way [16] (we will give a formal definition Section 9).

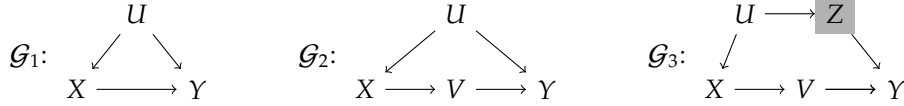


Figure 2: Identification of $P(y \mid do(x))$ in graphs with the unobserved nodes $\mathbf{V} \setminus \mathbf{R} = \{U\}$: In \mathcal{G}_1 the effect is not identifiable. In \mathcal{G}_2 , it is identifiable (using the front-door method [3]) but for this DAG no adjustment set \mathbf{Z} exists, with $\mathbf{Z} \subseteq \mathbf{V} \setminus \{U\}$. In \mathcal{G}_3 the effect can be identified via covariate adjustment $\mathbf{Z} = \{Z\}$.

Definition 1 (Adjustment [3]). *Given a DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ and pairwise disjoint $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$, \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 consistent with \mathcal{G} we have*

$$P(\mathbf{y} \mid do(\mathbf{x})) = \begin{cases} P(\mathbf{y} \mid \mathbf{x}) & \text{if } \mathbf{Z} = \emptyset, \\ \sum_{\mathbf{z}} P(\mathbf{y} \mid \mathbf{x}, \mathbf{z}) P(\mathbf{z}) & \text{otherwise.} \end{cases} \quad (2)$$

Identification via covariate adjustment is not complete in the sense that there exist graphs for which $P(\mathbf{y} \mid do(\mathbf{x}))$ is identifiable but for which no covariate adjustment set \mathbf{Z} , with $\mathbf{Z} \subseteq \mathbf{R}$, exists. For an example of such a DAG see Figure 2. In [21] (see also [3, Section 3.4]) Pearl introduced the *do-calculus* of intervention that was proven to be complete for identifying causal effects [22, 23]. Based on the rules of the do-calculus, the IDC algorithm proposed by Shpitser and Pearl [24] computes a formula for $P(\mathbf{y} \mid do(\mathbf{x}))$ if (and only if) the effect is identifiable. A drawback of the algorithm is that it can generate very complex formulas, even when a causal effect is also identifiable via covariate adjustment (Expression (2)) or in another simple manner. Also the runtime of the IDC algorithm is higher compared to our algorithms to determine adjustment sets, as we will show later.

Minimality In this paper we present methods for computing an adjustment set that satisfies the condition of Definition 1 for a given instance. An important feature of our approach is that it allows also to find adjustment sets satisfying some additional desirable constraints, e.g., minimality. Below we define this notion formally both for adjustment and separation sets.

For pairwise disjoint $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$, and a subset \mathbf{M} of \mathbf{V} , an m -separator (resp. adjustment set) \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 an m -separator (adjustment set) relative to (\mathbf{X}, \mathbf{Y}) . An m -separator (adjustment) \mathbf{Z} is \mathbf{M} -minimum according to a cost function $w : \mathbf{V} \rightarrow \mathbb{R}^+$ if $\mathbf{M} \subseteq \mathbf{Z}$ and no m -separator (adjustment) \mathbf{Z}' with $\mathbf{M} \subseteq \mathbf{Z}'$ and $\sum_{Z \in \mathbf{Z}'} w(Z) < \sum_{Z \in \mathbf{Z}} w(Z)$ exists. If \mathbf{M} is an empty set, then the \emptyset -minimality coincides with the standard notion for minimal/minimum m -separator, resp. adjustment set. We will call the \emptyset -minimality also *strong-minimality*. A more detailed explanation of these concepts along with some examples is provided in Section 3.2.

In this paper we consider m -separators (adjustments) \mathbf{Z} that are supersets of a constraining set \mathbf{I} . Thus, beside strongly-minimal separators and adjustments \mathbf{Z} , with $\mathbf{I} \subseteq \mathbf{Z}$, it is reasonable to investigate \mathbf{I} -minimal and \mathbf{I} -minimum sets. For a given constraining set \mathbf{I} we will abbreviate the \mathbf{I} -minimal/minimum $\mathbf{Z} \supseteq \mathbf{I}$ as *weakly-minimal/minimum* or just as *minimal/minimum*. Note a subtle, but important difference between weak and strong minimality. For example, the existence of a weakly-minimal m -separator does not necessarily imply that a strongly-minimal separator exists. E.g., in the DAG $X \rightarrow I \leftarrow V \rightarrow Y$, set $\mathbf{Z} = \{I, V\}$ is an \mathbf{I} -minimal d -separator relative to (X, Y) but in the graph there exists no strongly-minimal d -separator \mathbf{Z}' , with $I \in \mathbf{Z}'$. On the other hand, it is easy to see that every strongly-minimal m -separator \mathbf{Z} , with $\mathbf{I} \subseteq \mathbf{Z}$, is also an \mathbf{I} -minimal one and the same holds for the minimum sets.

3 Algorithms for m -separation in ancestral graphs

In this section, we compile an algorithmic framework for solving a host of problems related to verification, construction, and enumeration of m -separating sets in an ancestral graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ with n nodes in \mathbf{V} and m edges in \mathbf{E} . The problems are defined in Table 2, which also shows the asymptotic runtimes of our algorithms. The goal is to test or to output either arbitrary m -separating sets \mathbf{Z} without further constraints or m -separating sets that have a minimal

		Runtime	Reference
Verification:	For given $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and constraint \mathbf{I} decide if ...		
TESTSEP	\mathbf{Z} m -separates \mathbf{X}, \mathbf{Y}	$O(n + m)$	Proposition 1
TESTMINSEP	$\mathbf{Z} \supseteq \mathbf{I}$ m -separates \mathbf{X}, \mathbf{Y} and \mathbf{Z} is ...		
	\mathbf{I} -minimal	$O(n^2)$	Proposition 3
	strongly-minimal	$O(n^2)$	Proposition 3
Construction:	For given \mathbf{X}, \mathbf{Y} and constraints \mathbf{I}, \mathbf{R} , output an ...		
FINDSEP	m -separator \mathbf{Z} with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$O(n + m)$	Proposition 2
FINDMINSEP	m -separator \mathbf{Z} with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...		
	\mathbf{I} -minimal	$O(n^2)$	Proposition 4
	strongly-minimal	NP-hard	Proposition 7
FINDMINCOSTSEP	m -separator \mathbf{Z} with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...		
	\mathbf{I} -minimum	$O(n^3)$	Proposition 5
	strongly-minimum	$O(n^3)$	Proposition 6
Enumeration:	For given $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ enumerate all ...	Delay	
LISTSEP	m -separators \mathbf{Z} with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$O(n(n + m))$	Proposition 9
LISTMINSEP	\mathbf{I} -minimal m -separators \mathbf{Z} with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$O(n^3)$	Proposition 10

Table 2: Definitions of algorithmic tasks related to m -separation in an ancestral graph \mathcal{G} of n nodes and m edges and the time complexities of algorithms given in this section which solve 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. A minimum m -separator minimizes the sum $\sum_{Z \in \mathcal{Z}} w(Z)$ 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 [26]).

size, so that no node can be removed from the set without turning it into a non-separating set. A further constraint is that \mathbf{Z} should be bounded by arbitrary given sets $\mathbf{I} \subseteq \mathbf{R}$ as $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. The variables in \mathbf{I} will always be included in the m -separating set, even if the set remains a separator without these variables. This allows to deal with cases where study design (or reviewers) mandate that certain variables be adjusted for. The set $\mathbf{V} \setminus \mathbf{R}$ corresponds to a set of *unobserved* variables, which are known to exist, but have not been (or cannot be) measured. This constraint can also be used to model correlated errors between variables, i.e., rather than connecting such variables with a bidirected edge like in a semi-Markovian model, the variables are connected to another variable not in \mathbf{R} . Both constraints together are also an important technical tool for the design of efficient enumeration algorithms and the adjustment algorithms in the later sections.

The rest of this section describes our algorithms solving these problems, which are mostly generalizations of existing algorithms from [25, 13, 14, 1]. For each problem, we present the algorithm as a function of the same name as the problem, so that the association between problem and algorithm is easy to follow and the algorithms can be built upon each other.

In contrast to the preliminary conference version of this work [2], we state the algorithms separately for sparse and dense graphs, as well as introducing and analyzing a new notion of minimality.

3.1 Linear-time algorithms for testing and finding m -separators

The problems TESTSEP and FINDSEP can be solved immediately in the ancestral graph. TESTSEP just requires us to verify the m -separation definition for given sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$. In DAGs d -separation is usually tested with the Bayes-Ball algorithm [13], which can be visualized as sending balls through the graph along the edges, tracking which nodes have been visited from either their parents or children, until all reachable nodes have been visited. In other words, Bayes-Ball is basically a breadth-first-search with some rules that determine if an edge pair lets the ball pass, bounces it back on the same edge or blocks it completely. These rules can be adapted to m -separation as shown in Figure 3,

which leads to the following algorithm for testing if a given \mathbf{Z} m -separates \mathbf{X} and \mathbf{Y} in an AG \mathcal{G} :

```

function TESTSEP( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ )
   $\mathbf{P} \leftarrow \{(\leftarrow, X) \mid X \in \mathbf{X}\}$  ▷ list of pairs (type of edge, node) whose visit is pending
   $\mathbf{Q} \leftarrow \mathbf{P}$  ▷ all pending and ever visited nodes
  while  $\mathbf{P}$  not empty do
    Let  $(e, T)$  be a (type of edge, node) pair of  $\mathbf{P}$ 
    Remove  $(e, T)$  from  $\mathbf{P}$ 
    for all neighbors  $N$  of  $T$  do
      Let  $T$  and  $N$  be connected by edge  $f$ 
      if  $e, T, f$  is an  $m$ -connecting path segment and  $(f, N) \notin \mathbf{Q}$  then
        Add  $(f, N)$  to  $\mathbf{P}$  and  $\mathbf{Q}$ 
      if  $M \in \mathbf{Y}$  then return false
  return true

```

Analysis of the Algorithm. From the rules in Figure 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 m -connected walk. The correctness follows from the fact that the algorithm searches over all walks starting in \mathbf{X} . The runtime is linear as it is a breadth-first-search and each node is visited at most four times. The rules for entering a node through edges \rightarrow and \leftrightarrow as well as through edges \leftarrow and $-$ are the same, so an implementation could merge these cases in \mathbf{Q} , so each node is visited at most twice. \square

Proposition 1. *Using the algorithm above, the task TESTSEP can be solved in time $O(n + m)$.*

The next problem FINDSEP asks for a set \mathbf{Z} m -separating given \mathbf{X} and \mathbf{Y} with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. A canonical solution for this problem is given in the following lemma.

Lemma 1. *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 an m -separator \mathbf{Z}_0 relative to (\mathbf{X}, \mathbf{Y}) with $\mathbf{I} \subseteq \mathbf{Z}_0 \subseteq \mathbf{R}$, then $\mathbf{Z} = \text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \cap \mathbf{R}$ is an m -separator.*

Proof. Let us consider a proper walk $w = X, V_1, \dots, V_n, Y$ with $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$. If w does not contain a collider, all nodes V_i are in $\text{Ant}(\mathbf{X} \cup \mathbf{Y})$ and the walk is blocked by \mathbf{Z} , unless $\{V_1, \dots, V_n\} \cap \mathbf{R} = \emptyset$ in which case the walk is not blocked by \mathbf{Z}_0 either. If the walk contains colliders \mathbf{C} , it is blocked, unless $\mathbf{C} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. Then all nodes V_i are in $\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$ and the walk is blocked, unless $\{V_1, \dots, V_n\} \cap \mathbf{R} = \mathbf{C}$. Since $\mathbf{C} \subseteq \mathbf{Z}$ is a set of anteriors, there exists a shortest (possible of length zero) path $\pi_j = V_j \rightarrow \dots \rightarrow W_j$ for each $V_j \in \mathbf{C}$ with $W_j \in \mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}$ (it cannot contain an undirected edge, since there is an arrow pointing to V_j). Let $\pi'_j = V_j \rightarrow \dots \rightarrow W'_j$ be the shortest subpath of π_j that is not blocked by \mathbf{Z}_0 . Let w' be the walk w after replacing each V_j by the walk $V_j \rightarrow \dots \rightarrow W'_j \leftarrow \dots \leftarrow V_j$. If any of the W'_j is in $\mathbf{X} \cup \mathbf{Y}$ we truncate the walk, such that we get the shortest walk between nodes of \mathbf{X} and \mathbf{Y} . Since π'_j is not blocked, w' contains no colliders except w'_j and all other nodes of w' are not in \mathbf{R} , w' is not blocked and \mathbf{Z}_0 is not a separator. \square

This yields the following algorithm to find an m -separator relative to \mathbf{X}, \mathbf{Y} :

```

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

```

Analysis of the Algorithm. The algorithm finds an m -separator due to Lemma 1 and runs in linear time, since the set $\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \cap \mathbf{R}$ can be calculated in linear time and the algorithm TESTSEP runs in linear time as well. \square

Proposition 2. *Using the algorithm above, the task FINDSEP can be solved in time $O(n + m)$.*

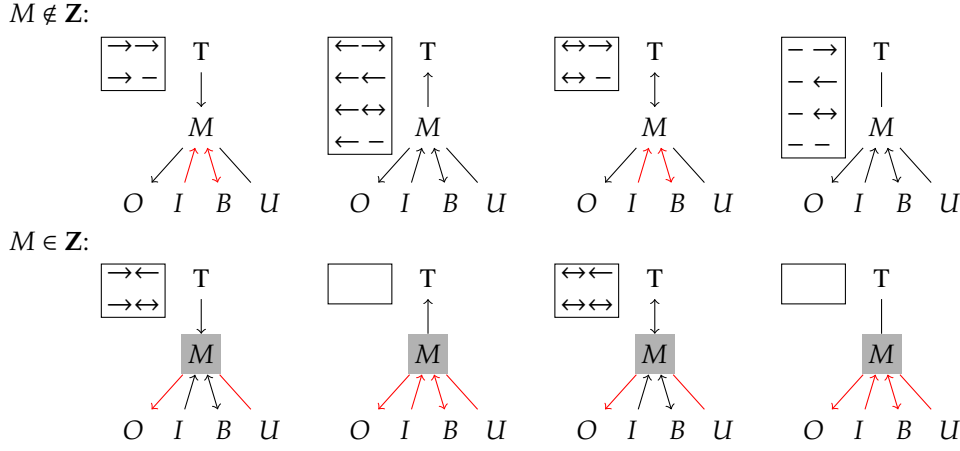


Figure 3: Expanded rules for Bayes-Ball in AGs, listing (in boxes) all combinations of edge pairs through which the ball is allowed to pass. The Bayes ball starts at the top node T and passes through the middle node M to one of the bottom nodes $\{O, I, B, U\}$. Forbidden passes are marked in red. Here, by a pair of edges we mean an edge between T (Top node) and M (Middle node) and $M \rightarrow O$ (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: $M \notin Z$ and $M \in Z$ (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 literature.

Set R is required to be disjoint with X and Y , because m -separation of a set Z relative to X, Y is not defined when Z contains a node of X or Y , so Z is always a subset of $R \subseteq V \setminus (X \cup Y)$. However, our algorithms still remove $X \cup Y$ from 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 Polynomial time algorithms for minimal and minimum m -separators

In this section we present algorithms for testing minimal m -separators for both variants of the minimality. We give also algorithms for constructing weakly- and strongly-minimum m -separators Z satisfying constraints $I \subseteq Z \subseteq R$ and for finding weakly-minimal separators. Computing strongly-minimal separators appears to be a hard problem – we discuss its complexity in detail in the next section. This is a very surprising result since finding objects of minimum costs or sizes is, typically, harder than constructing minimal objects.

Our algorithms are easily implementable and have runtimes $O(n^2)$, resp. $O(n^3)$. These time complexities are reasonable particularly in case of dense ancestral graphs, i.e. graphs with a large number of edges $n \ll m \leq n^2$ and $O(n + m) = O(n^2)$. In Section 3.4 we propose algorithms which are faster than those presented below, when the instance graphs are sparse.

The algorithms for minimal m -separators consist of two phases. First we convert the ancestral graph to a special augmented graph. Then, a minimal separator is searched as a vertex cut in the augmented graph. This approach generalizes d -separation algorithms of [14] and [1], particularly the handling of the undirected graph after the moralization step is the same as in [14].

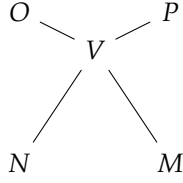
Constructing the augmented graph according to its definition and searching a collider connected path for all pairs of nodes would lead to a suboptimal algorithm. Therefore, we first give an asymptotically optimal (linear time in output size) moralization algorithm for AGs:

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

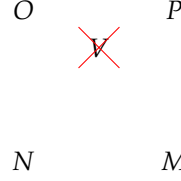
Proof. The algorithm proceeds in four steps.

1. Start by setting $(\mathcal{G})^a$ to the skeleton of \mathcal{G} .

V and its neighbors $Ne(V)$ in \mathcal{G}^a :



Case: $V \in \mathbf{I}$



Case: $V \notin \mathbf{R}$

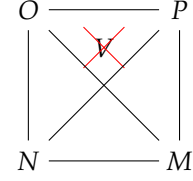


Figure 4: This figure explains the removal of nodes in \mathbf{I} and outside of \mathbf{R} from the augmented graph $(\mathcal{G}_{Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^a$. Shown is an exemplary node V with all its neighbors in the augmented graph. In the case $V \in \mathbf{I}$ the node 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. The time needed to insert the new edges is $O(|\mathbf{V} \setminus \mathbf{R}| \cdot |Ne(\mathbf{V} \setminus \mathbf{R})|^2) = O(n^3)$, so this removal of nodes outside \mathbf{R} is only used for algorithm LISTMINSEP in Section 3.5 and the other algorithms handle such nodes with different approaches.

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 $U - V$ 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 $O(n^2)$. \square

Lemma 1 gave us a closed form solution to find an m -separator, which yields a constraint for minimal m -separators:

Corollary 1 (Ancestry of \mathbf{M} -minimal separators). *Given an AG \mathcal{G} and three sets $\mathbf{X}, \mathbf{Y}, \mathbf{M}$, every \mathbf{M} -minimal m -separator \mathbf{Z} is a subset of $Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$.*

Proof. Assume there is an \mathbf{M} -minimal separator \mathbf{Z} with $\mathbf{Z} \not\subseteq Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$. Setting $\mathbf{I} = \mathbf{M}$ and $\mathbf{R} = \mathbf{Z}$, Lemma 1 shows that $\mathbf{Z}' = Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) \cap \mathbf{Z}$ is an m -separator with $\mathbf{M} \subseteq \mathbf{Z}'$. But $\mathbf{Z}' \subseteq Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})$ and $\mathbf{Z}' \subseteq \mathbf{Z}$, so $\mathbf{Z} \neq \mathbf{Z}'$ and \mathbf{Z} is not an \mathbf{M} -minimal separator. \square

Corollary 2 (Ancestry of minimal separators). *Given an AG \mathcal{G} and three sets $\mathbf{X}, \mathbf{Y}, \mathbf{I}$, every \mathbf{I} -minimal or \emptyset -minimal m -separator is a subset of $Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$.*

Proof. This follows from Corollary 1 with $\mathbf{M} = \mathbf{I}$ or $\mathbf{M} = \emptyset$. In both cases we have $Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) \subseteq Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$. \square

Corollary 2 applies to minimum separators as well because every minimum separator must be minimal. For all strongly-minimal (i.e., recall, \emptyset -minimal) or \mathbf{I} -minimal m -separator \mathbf{Z} satisfying $\mathbf{I} \subseteq \mathbf{Z}$ the corollary implies $Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}) = Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$ and thus $(\mathcal{G}_{Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})})^a = (\mathcal{G}_{Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^a$, so the augmented graph $(\mathcal{G}_{Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})})^a$ can be constructed without knowing the actual \mathbf{Z} . We will use m_a to denote the number of edges in $(\mathcal{G}_{Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^a$. Since nodes in \mathbf{I} as well as nodes outside \mathbf{R} are known to either be in \mathbf{Z} or to be excluded from \mathbf{Z} , they do not need to be considered by the algorithms and can be removed from the augmented graph as shown in Figure 4, similarly to the approach in [25].

TESTMINSEP and FINDMINSEP can now be solved by a modified breadth-first-search in the graph $(\mathcal{G}_{Ant(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z})})^a$. We start with providing the function TESTMINSEP which tests if $\mathbf{Z} \subseteq \mathbf{R}$ is an \mathbf{M} -minimal m -separator relative to (\mathbf{X}, \mathbf{Y}) in an AC \mathcal{G} :

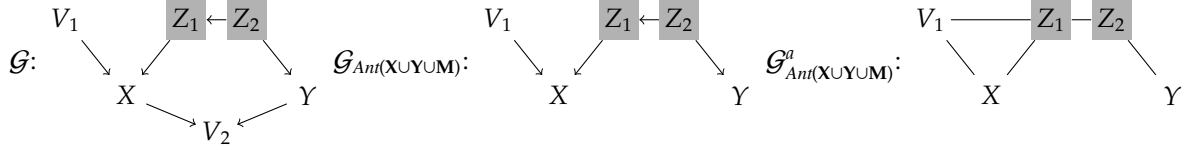


Figure 5: The transformation of a graph \mathcal{G} to $\mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})}$ to $(\mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})})^a$ with $\mathbf{M} = \emptyset$. The m -separating set $\{Z_1, Z_2\}$ is not minimal as no node is reachable from both \mathbf{X} and \mathbf{Y} , but each node alone $\{Z_1\}$ or $\{Z_2\}$ is a minimal m -separator.

```

function TESTMINSEP( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{M}, \mathbf{R}$ )
  if  $\mathbf{Z} \setminus \text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M}) \neq \emptyset$  or  $\mathbf{Z} \not\subseteq \mathbf{R}$  then return false
  if not TESTSEP( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ ) then return false
   $\mathcal{G}'^a \leftarrow (\mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{M})})^a$ 
  Remove from  $\mathcal{G}'^a$  all nodes of  $\mathbf{M}$ .
   $R_x \leftarrow \{Z \in \mathbf{Z} \mid \exists \text{ path from } X \text{ to } Z \text{ in } \mathcal{G}'^a \text{ not intersecting } \mathbf{Z} \setminus \{Z\}\}$ 
  if  $\mathbf{Z} \neq R_x$  then return false
   $R_y \leftarrow \{Z \in \mathbf{Z} \mid \exists \text{ path from } Y \text{ to } Z \text{ in } \mathcal{G}'^a \text{ not intersecting } \mathbf{Z} \setminus \{Z\}\}$ 
  if  $\mathbf{Z} \neq R_y$  then return false
  return true

```

Analysis of the Algorithm. TESTMINSEP, runs in $O(m_a)$ because R_x and R_y can be computed with an ordinary search that aborts when a node in \mathbf{Z} is reached. If each node in \mathbf{Z} is reachable from both \mathbf{X} and \mathbf{Y} , the set is \mathbf{M} -minimal as no node can be removed without opening a connecting path as shown in the example of Figure 5.

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. \square

Proposition 3. Using the algorithm above, the task TESTMINSEP, both for testing \mathbf{I} -minimality and strong-minimality, can be solved in time $O(m_a) = O(n^2)$.

The difference between \mathbf{I} -minimal and strongly-minimal separation sets $\mathbf{Z} \supseteq \mathbf{I}$ is illustrated in Figure 5. There are exactly two strongly-minimal separating sets: $\{Z_1\}$ and $\{Z_2\}$. No other m -separator will be strongly-minimal regardless of which nodes are added to a constraining set \mathbf{I} . Therefore, for $\mathbf{I} = \emptyset$, both m -separators satisfy the constraint, for $\mathbf{I} = \{Z_1\}$ or $\mathbf{I} = \{Z_2\}$, only one of them does and, for a $\mathbf{I} = \{Z_1, Z_2\}$ or $V_1 \in \mathbf{I}$, no strongly-minimal m -separator satisfies it. The constraint \mathbf{I} just chooses some m -separators of a fixed set of strongly-minimal m -separators.

On the other hand, when computing an \mathbf{I} -minimal m -separator, we treat the nodes of \mathbf{I} as fixed and search for a minimal m -separator among all supersets of \mathbf{I} . In Figure 5, if \mathbf{I} is either $\{Z_1\}$, $\{Z_2\}$ or $\{Z_1, Z_2\}$, then \mathbf{I} itself is an \mathbf{I} -minimal m -separator and no other \mathbf{I} -minimal m -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 m -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 5, for $\mathbf{M} = \{V_1\}$, $\mathbf{I} = \{Z_1, V_1\}$ would only accept $\{Z_1, V_1\}$ as m -separator. $\mathbf{M} = \{Z_1\}$, $\mathbf{I} = \{Z_1, Z_2\}$ would not allow any. Every m -separator $\mathbf{Z} \subseteq \mathbf{R}$ is \mathbf{Z} -minimal, \mathbf{R} -minimal and \mathbf{V} -minimal.

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

```

function FINDMINSEP( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ )
   $\mathcal{G}' \leftarrow \mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})}$ 
   $\mathcal{G}'^a \leftarrow (\mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^a$ 
  Remove from  $\mathcal{G}'^a$  all nodes of  $\mathbf{I}$ .
   $\mathbf{Z}' \leftarrow \mathbf{R} \cap \text{Ant}(\mathbf{X} \cup \mathbf{Y}) \setminus (\mathbf{X} \cup \mathbf{Y})$ 
   $\mathbf{Z}'' \leftarrow \{Z \in \mathbf{Z}' \mid \exists \text{ a path from } \mathbf{X} \text{ to } Z \text{ in } \mathcal{G}'^a \text{ not intersecting } \mathbf{Z}' \setminus \{Z\}\}$ 
   $\mathbf{Z} \leftarrow \{Z \in \mathbf{Z}'' \mid \exists \text{ a path from } \mathbf{Y} \text{ to } Z \text{ in } \mathcal{G}'^a \text{ not intersecting } \mathbf{Z}'' \setminus \{Z\}\}$ 
  if not TESTSEP( $\mathcal{G}', \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ ) then return  $\perp$ 
  return  $\mathbf{Z} \cup \mathbf{I}$ 

```

Analysis of the Algorithm. Algorithm FINDMINSEP begins with the separating set $\mathbf{R} \cap \text{Ant}(\mathbf{X} \cup \mathbf{Y}) \setminus (\mathbf{X} \cup \mathbf{Y})$ and finds a subset satisfying the conditions tested by algorithm TESTMINSEP. As TESTMINSEP it can be implemented in $O(m_a)$ using a breadth-first-search starting from \mathbf{X} (\mathbf{Y}) that aborts when a node in \mathbf{Z}' (\mathbf{Z}'') is reached. \square

Algorithm FINDMINSEP finds an \mathbf{I} -minimal m -separator. Note that setting the argument $\mathbf{I} = \emptyset$ does not lead to a strongly-minimal separator, because then the returned set \mathbf{Z} may no longer satisfy the constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$.

Proposition 4. *The algorithm above finds an \mathbf{I} -minimal m -separator \mathbf{Z} , with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$, in time $O(m_a) = O(n^2)$.*

In the problem FINDMINCOSTSEP, each node V is associated with a cost $w(V)$ given by a cost function $w : \mathbf{V} \rightarrow \mathbb{R}^+$ and the task is to find a set \mathbf{Z} m -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 an m -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 an m -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 afterwards solve the problem with any weighted min-cut algorithm.

```

function FINDMINCOSTSEP( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}, w$ )
   $\mathcal{G}' \leftarrow \mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})}$ 
   $\mathcal{G}'^a \leftarrow (\mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^a$ 
  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 $O(n^3)$ due to the well-known min-cut-max-flow theorem [27, Chapter 6]. \square

Proposition 5. *The algorithm above solves in time $O(n^3)$ the task FINDMINCOSTSEP in case of \mathbf{I} -minimality.*

The runtime primarily depends on the used min-cut/max-flow algorithm. Using a state-of-the-art max-flow algorithm recently presented by Orlin improves the runtime to $O(nm_a)$ [28], although this is no necessarily an improvement in our setting, because the augmented graph can have $m_a = O(n^2)$ edges and then $O(nm_a)$ and $O(n^3)$ 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 $O(m_a \sqrt{n})$ algorithm is known [27], 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 $\mathbf{R}' = (\mathbf{R} \setminus (\mathbf{X} \cup \mathbf{Y})) \cap \text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$ by removing the nodes of $\mathbf{V} \setminus \mathbf{R}$ iteratively in $O((n - |\mathbf{R}|)n^2)$ as shown in Figure 4 or by creating a new graph only containing those nodes in $O(|\mathbf{R}'|m_a)$ as described in [14], resulting in a total runtime of $O(\min((n - |\mathbf{R}|)n^2, |\mathbf{R}'|m_a) + m_a \sqrt{|\mathbf{R}'|}) = O(|\mathbf{R}'|m_a)$ for a unit cost function.

The set \mathbf{Z} returned by algorithm `FINDMINCOSTSEP` is also 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 if $\sum_{Z \in \mathbf{Z}} w(Z) = \sum_{Z \in \mathbf{Z}'} w(Z)$. 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 sum. Thus, we get the following:

Proposition 6. *Finding an m -separator \mathbf{Z} , with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$, which is strongly-minimum can be done in time $O(n^3)$*

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 m -separator satisfying the constraints exist.

3.3 The hardness of strong-minimality

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

In this section we show that it is hard to find a strongly-minimal m -separator even for singleton X, Y in DAGs, in which m -separation and d -separation are equivalent:

Proposition 7. *It is an NP-complete problem to decide if in a given DAG there exists a strongly-minimal d -separating set \mathbf{Z} containing \mathbf{I} .*

Proof. The problem is in NP, since given \mathbf{Z} verifying $\mathbf{I} \subseteq \mathbf{Z}$ is trivial 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 [29], 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, \dots, V_k , and ℓ clauses $C_i = (W_{i,1} \vee W_{i,2} \vee W_{i,3})$ of literals $W_{i,j} \in \{V_1, \dots, V_k, \overline{V}_1, \dots, \overline{V}_k\}$. It is NP-hard to decide if there exists a Boolean assignment $\phi : \{V_1, \dots, V_k\} \rightarrow \{\text{true}, \text{false}\}$ that satisfies the formula $C_1 \wedge \dots \wedge C_\ell$.

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

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

The resulting graph is shown in Figure 6. 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

$$\mathbf{Z} = \mathbf{I} \cup \{V_i, V_{i,r} \mid i \in \{1, \dots, k\}, \phi(V_i) = \text{false}\} \cup \{\overline{V}_i, V_{i,l} \mid i \in \{1, \dots, k\}, \phi(V_i) = \text{true}\}.$$

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. The parents $V_{i,l}$ are reachable, but

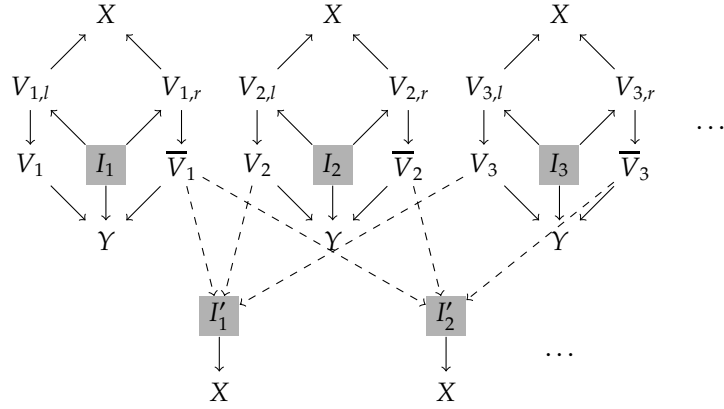


Figure 6: Here we show the graph used in the proof of Proposition 7. It represents the first three variables V_1, V_2, V_3 , and two clauses $(\bar{V}_1 \vee V_2 \vee V_3)$ and $(\bar{V}_1 \vee \bar{V}_2 \vee \bar{V}_3)$. All shown X (Y) nodes can be considered to be a single node X (Y), but we display them separately to reduce the number of overlapping edges in the figure.

they are either in \mathbf{Z} or only $V_i, I_i \in \mathbf{Z}$ are reachable and the path stops there. Likewise no path through $V_{i,r}$ can reach Y . Hence \mathbf{Z} is a d -separator.

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

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

“ \Rightarrow ”: Now we show that a strongly-minimal d -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 $X \leftarrow V_{i,r} \rightarrow \bar{V}_i \rightarrow Y$ need to be blocked by a node of $\{V_{i,l}, V_i\}$ and a node of $\{V_{i,r}, \bar{V}_i\}$. If neither V_i nor \bar{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 d -separator and \mathbf{Z} is not strongly-minimal. Therefore V_i or \bar{V}_i is in \mathbf{Z} and the following Boolean assignment ϕ to the variables is well-defined:

$$\phi(V_i) = \begin{cases} \text{true} & V_i \notin \mathbf{Z}, \\ \text{false} & \bar{V}_i \notin \mathbf{Z}, \\ \text{false} & \text{otherwise.} \end{cases}$$

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 $I'_i \leftarrow \bar{V}_j \rightarrow Y$) and V_j (or \bar{V}_j) is not in \mathbf{Z} for some j . This V_j (or \bar{V}_j) satisfies clause C_i according to the definition of ϕ . Hence every clause and the formula is satisfiable. \square

The moralization of the constructed graph implies that it is also NP-hard to find strongly-minimal vertex separators in undirected graphs and from the characterizations of adjustment sets in the next sections it will follow that it is also NP-hard to find strongly-minimal adjustments in DAGs or ancestral graphs.

3.4 Algorithms for minimal m -separators in sparse graphs

Subsection 3.2 gave $O(n^2)$ algorithms for the problems TESTMINSEP and FINDMINSEP in case of \mathbf{I} -minimality, which is optimal for dense graphs. The runtime of these algorithms is limited by the time required for construction of the augmented graph, which might contain nodes and edges that have no influence on the minimal m -separator

and are thus included unnecessarily. This leads to the question, can we test or find the minimal m -separator without constructing the augmented graph? The obvious way is to test the definition of minimality directly by removing each of the nodes of \mathbf{Z} and testing if the smaller set is still an m -separator, which yields $O(|\mathbf{Z}|(n+m)) = O(|\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})|(n+m)) = O(n(n+m))$ algorithms. This is generally slower than the previous runtimes, however in sparse graphs that have a low number of edges $m \approx n$ and a low number of ancestors $|\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})| \ll n$, it might be faster. This alternative algorithm for TESTMINSEP in sparse graphs, in case of \mathbf{I} -minimality, is as follows

```

function TESTMINSEPPARSE( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{I}, \mathbf{R}$ )
  if  $\mathbf{Z} \setminus \text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \neq \emptyset$  or  $\mathbf{Z} \not\subseteq \mathbf{R}$  then return false
  if not TESTSEP( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ ) then return false
   $\mathcal{G}' \leftarrow \mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})}$ 
  for all  $U \leftarrow \mathbf{Z} \setminus \mathbf{I}$  do
    if TESTSEP( $\mathcal{G}', \mathbf{X}, \mathbf{Y}, \mathbf{Z} \setminus \{U\}$ ) then return false
  return true

```

An \mathbf{I} -minimal m -separator relative to (\mathbf{X}, \mathbf{Y}) can be computed using the algorithm

```

function FINDMINSEPPARSE( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ )
   $\mathcal{G}' \leftarrow \mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})}$ 
   $\mathbf{Z} \leftarrow \mathbf{R} \cap \text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \setminus (\mathbf{X} \cup \mathbf{Y})$ 
  if not TESTSEP( $\mathcal{G}', \mathbf{X}, \mathbf{Y}, \mathbf{Z}$ ) then return  $\perp$ 
  for all  $U$  in  $\mathbf{Z} \setminus \mathbf{I}$  do
    if TESTSEP( $\mathcal{G}', \mathbf{X}, \mathbf{Y}, \mathbf{Z} \setminus \{U\}$ ) then
       $\mathbf{Z} \leftarrow \mathbf{Z} \setminus \{U\}$ 
  return  $\mathbf{Z}$ 

```

Proposition 8. *The tasks TESTMINSEP and FINDMINSEP for \mathbf{I} -minimal separators can be solved in time $O((n+m) \cdot |\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})|)$.*

The correctness of these algorithms depends on the non-obvious fact that m -separators are monotone: if $\mathbf{Z} \setminus V$ is not an m -separator, no $\mathbf{Z} \setminus \mathbf{V}$ with $V \in \mathbf{V}$ is one either. This monotonicity was proven for d -separation by [14] and we will state their results in terms of m -separation:

Lemma 3. *Let \mathbf{X}, \mathbf{Y} be two sets and let \mathbf{Z} be an m -separator relative to (\mathbf{X}, \mathbf{Y}) . If the set $\mathbf{Z} \cup \{Z_1, \dots, Z_n\}$ is also an m -separator, where Z_1, \dots, Z_n are single nodes which are not in \mathbf{Z} then either $\mathbf{Z} \cup \{Z_1\}$, or $\mathbf{Z} \cup \{Z_2\}$, ... or $\mathbf{Z} \cup \{Z_n\}$ must be another m -separator between \mathbf{X} and \mathbf{Y} .*

Corollary 3. *If \mathbf{Z} and $\mathbf{Z} \cup \mathbf{Z}_n$ are m -separators, where $\mathbf{Z}_n = \{Z_1, \dots, Z_n\}$, then there exist a series of $n-1$ m -separators: $\mathbf{Z} \cup \mathbf{Z}_i$, $i = 1, \dots, n-1$, with $\mathbf{Z}_1 \subset \dots \subset \mathbf{Z}_{n-1} \subset \mathbf{Z}_n$ such that each \mathbf{Z}_i contains exactly i nodes.*

Corollary 4. *If no single node can be removed from a m -separator \mathbf{Z} without destroying m -separability, then \mathbf{Z} is minimal.*

To generalize the proofs of [14] to m -separation, it is sufficient to replace d -separation with m -separation, DAGs with AGs and ancestors with antérieurs throughout their arguments. Therefore, we do not repeat the proofs here.

3.5 Algorithms for enumerating all m -separators

Lastly, we consider the problem of listing *all* m -separators and *all* minimal m -separators between \mathbf{X} and \mathbf{Y} in \mathcal{G} . Since there might be an exponential number of (minimal) m -separators, it is not possible to list them all in polynomial time. For example in a path $X \leftarrow V \leftarrow V' \leftarrow Y$ either V or V' must be in a minimal m -separator between X and Y , so a graph containing k such paths will have at least 2^k different m -separators. Therefore we will describe algorithms running with $O(n^3)$ delay, which means that at most $O(n^3)$ time will pass between the start or the output of an m -separator and the output of the next m -separator or the end of the algorithm. They are based on the enumeration algorithm for minimal vertex separators of [26].


```

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 will output a separator, the tree height is at most n and the existence check needs $O(n + m)$, the delay time is $O(n(n + m))$. 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. \square

Proposition 9. *Using the algorithm above, the task LISTSEP can be solved with polynomial delay $O(n(n + m))$.*

To enumerate all minimal m -separators we can directly apply Takata’s enumeration algorithm [26] after transforming the ancestral graph to its augmented graph:

```

function LISTMINSEP( $\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ )
   $\mathcal{G}' \leftarrow \mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})}$ 
   $\mathcal{G}'^a \leftarrow (\mathcal{G}_{\text{Ant}(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^a$ 
  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 [26] to list all sets separating  $X^m$  and  $Y^m$ .

```

Analysis of the Algorithm. The correctness is shown by [1] for adjustment sets and generalizes directly to m -separators, because after moralization, both problems are equivalent to enumerating vertex cuts of an undirected graph. The handling of \mathbf{I} is shown by [25]. \square

Proposition 10. *The task LISTMINSEP can be solved with polynomial delay $O(n^3)$.*

4 Empirical analysis of DAG consistency testing

The concept of m -separation in ancestral graphs, or d -separation in DAGs, is of central importance in the field of Bayesian networks and graphical causal models because it couples the graphical model structure to conditional independencies in the corresponding probability distribution. Perhaps the most direct application of graphical separation is therefore *consistency testing*, where we check whether a graphical causal model is in fact consistent with the dataset it is intended to represent. If a graphical causal model fails to pass this test, then any downstream analyses, such as implied covariate adjustment sets, are potentially nullified. In this section, we illustrate how the algorithmic framework that we developed in Section 3 can be harnessed for graphical causal model checking by applying it to the problem of deriving *basis sets* for testing the consistency of a DAG to a given probability distribution.

A DAG \mathcal{G} is called *consistent* with a probability distribution P if for all pairwise disjoint subsets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$, where only \mathbf{Z} may be empty, \mathbf{X} and \mathbf{Y} are conditionally independent given \mathbf{Z} in P whenever \mathbf{Z} d -separates \mathbf{X} and \mathbf{Y} in \mathcal{G} . Therefore, to test consistency, we could in principle enumerate all d -separation relations in \mathcal{G} and then perform the corresponding conditional independence tests. However, that set of relations can be very large – there is already an exponential number of subsets \mathbf{X} and \mathbf{Y} to consider, each of which could be separated by an exponential number of sets \mathbf{Z} . A *basis set* of d -separation statements entails all other statements implied by \mathcal{G} when combining them using the *graphoid axioms* [3]. The *canonical* basis set has the form

$$\{\forall X \in \mathbf{V} : X \perp\!\!\!\perp \mathbf{V} \setminus \text{Pa}(X) \setminus \text{De}(X) \mid \text{Pa}(X)\},$$

or in words, every variable must be jointly independent of its non-descendants conditioned on its parents. This basis set only contains n elements but the independencies to be tested can be of very high order.

In practice, conditional independence is often tested using methods that do not distinguish between joint and pairwise independence, such as partial correlation or linear regression. In such cases, even simpler basis sets can be derived. For instance, Pearl and Meshkat [30] discuss basis sets for hierarchical linear regression models (also known as path models or structural equation models [31]). We here consider the vertices $\mathbf{V} = \{X_1, \dots, X_n\}$ to be indexed topologically, such that for $j > i$, we have that $X_j \notin \text{An}(X_i)$. Then we consider basis sets having the form

$$\{\forall j > i, X_i \rightarrow X_j \notin \mathbf{E} : X_i \perp\!\!\!\perp X_j \mid \mathbf{Z}_{ij}\},$$

or in words, we include one separation statement for every nonadjacent pair of variables. These are pairwise rather than joint independencies, which is equivalent when we measure dependence only by partial correlation [30]. An obvious choice is $\mathbf{Z}_{ij} = \text{Pa}(X_j)$, in which case we simply obtain a pairwise version of the canonical basis set above. We call that basis set the *parental basis set*. However, Pearl and Meshkat show that another valid choice is any separating set \mathbf{Z}_{ij} that contains only nodes whose distance to X_j is smaller than the distance between X_i and X_j (where the distance is defined length of a shortest path between two nodes). We call any such set a *sparse basis set*. Note that every parental basis set is also sparse, but the separators in sparse basis sets can be substantially smaller than in parental basis sets.

4.1 Simulation setup

We evaluate the difference between parental and sparse basis sets on random DAGs. These are generated with various numbers of variables

$$n \in \{10, 25, 50, 100\}.$$

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

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

For small n the probabilities are capped at 1. For example, for $n = 10$ a parameter $P(\text{edge}) = \min\{\frac{10}{10-1}, 1\} = 1$ 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

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

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

Our algorithmic framework makes it straightforward to compute sparse basis sets with minimal separating sets by simply setting $\mathbf{R} = \{X_k \in \mathbf{V} \mid d(X_k, X_j) < d(X_i, X_j)\}$, where we use $d(X_a, X_b)$ to denote the distance between two nodes.

4.2 Empirical results

We empirically evaluate the sizes of parental and sparse basis sets in these random DAGs (Figure 7). The results show that the benefit of using sparse rather than parental basis sets does not depend much on the size of the DAG but rather on the amount of edges. For instance, when nodes had on average 5 neighbours, sparse basis sets had between 20% and 40% fewer conditioning variables than the canonical parental basis sets.

These results provide a first example of how our framework can be applied in the context of graphical causal modelling. In the rest of this paper, we focus on covariate adjustment as our focus application area. However, given that separators play a central role in many aspects of graphical causal models, we expect that there should be many more applications in addition to those shown in this paper.

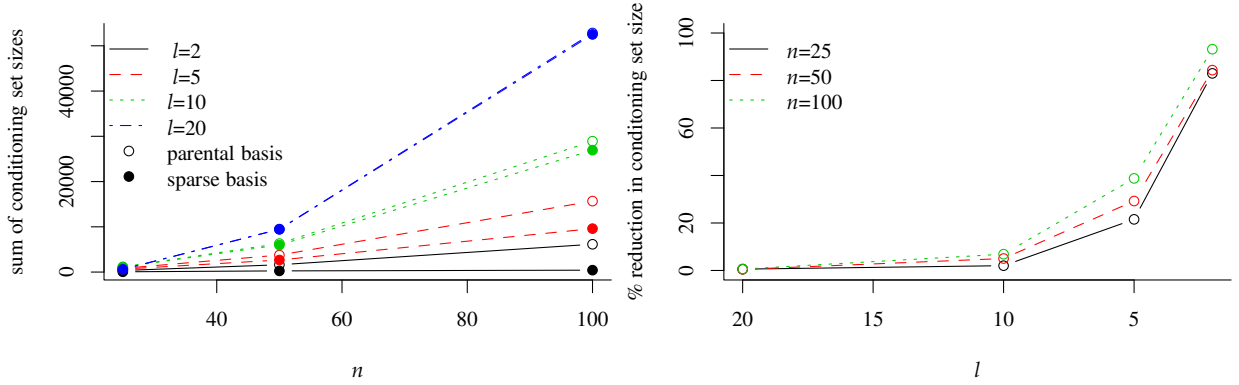


Figure 7: Model checking with parental or sparse basis sets for random DAGs. Like above, n denotes the number of nodes in the DAG, and l denotes the expected number of neighbours of each node. The left panel shows the total size of the conditioning sets as a function of n , showing the expected quadratic increase. The right panel emphasizes that the benefit of using a sparse basis set is greatest if the graph is also sparse, in which case the total number of variables that need to be conditioned on can be reduced by up to 90%.

5 Adjustment in DAGs

We now leverage the algorithmic framework of Section 3 together with a constructive, sound and complete criterion for covariate adjustment in DAGs to solve all problems listed in Table 2 for adjustment sets rather than m -separators in the same asymptotic time.

First, note that the formal definition of adjustments (Definition 1) cannot be used to actually find an adjustment set as there are infinitely many probability distributions that are consistent to a certain graph. Fortunately, it is possible to characterize adjustment sets in graphical terms.

Definition 2 (Adjustment criterion (AC) [10]). *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*

- (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} .*

However, even this criterion does not lead to efficient algorithms to find adjustment sets, because there can be exponentially many paths and blocking one non-causal path might open other paths. We address this problem below by presenting a constructive adjustment criterion that reduces non-causal paths to ordinary m -separation, which can be reduced further to a reachability problem in undirected graphs.

5.1 Constructive back-door criterion

The graph of this reduction will be called the proper back-door graph:

Definition 3 (Proper back-door graph). *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG, and $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. The proper back-door graph, denoted as $\mathcal{G}_{\mathbf{XY}}^{pbd}$, is obtained from \mathcal{G} by removing the first edge of every proper causal path from \mathbf{X} to \mathbf{Y} .*

Note the difference between the proper back-door graph $\mathcal{G}_{\mathbf{XY}}^{pbd}$ and the famous back-door graph $\mathcal{G}_{\mathbf{X}}$ of [3]: in $\mathcal{G}_{\mathbf{X}}$ all edges leaving \mathbf{X} are removed while in $\mathcal{G}_{\mathbf{XY}}^{pbd}$ only those that lie on a proper causal path (see Figure 9 for an example). However, to construct $\mathcal{G}_{\mathbf{XY}}^{pbd}$ still only elementary operations are sufficient. Indeed, we remove all edges $X \rightarrow D$ in \mathbf{E} such that $X \in \mathbf{X}$ and D is in the subset, which we call $PCP(\mathbf{X}, \mathbf{Y})$, obtained as follows

$$PCP(\mathbf{X}, \mathbf{Y}) = (De_{\overline{\mathbf{X}}}(\mathbf{X}) \setminus \mathbf{X}) \cap An_{\underline{\mathbf{X}}}(\mathbf{Y}). \quad (3)$$

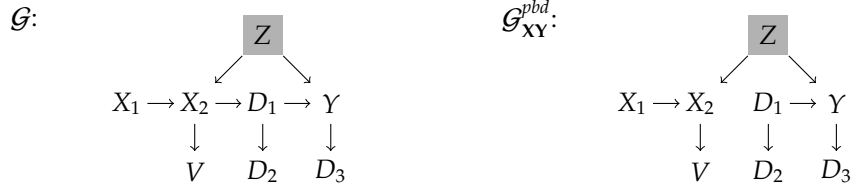


Figure 8: A DAG that permits exactly two adjustment sets for estimating the causal effect of $\mathbf{X} = \{X_1, X_2\}$ on $\mathbf{Y} = \{Y\}$: $\mathbf{Z} = \{Z\}$ and $\mathbf{Z}' = \{Z, V\}$. The set $\mathbf{V} \setminus Dpcp(\mathbf{X}, \mathbf{Y}) = \{X_1, X_2, Z, V\}$. So, every adjustment is 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 $X_2 \rightarrow D_1 \rightarrow Y$. Moreover, every adjustment must contain the variable Z to block the path between X_2 and Y in $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd}$.

Hence, the proper back-door graph can be constructed from \mathcal{G} in linear time $O(m + n)$. Note that using the notation PCP the proper back-door graph can be specified as

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

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

Definition 4 (Constructive back-door criterion (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) $\mathbf{Z} \subseteq \mathbf{V} \setminus Dpcp(\mathbf{X}, \mathbf{Y})$ and
- (b) \mathbf{Z} *d-separates* \mathbf{X} and \mathbf{Y} in the proper back-door graph $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd}$.

Figure 8 shows how the constructive back-door criterion can be applied to find an adjustment set in an exemplary DAG.

The CBC is surprisingly robust to modifications of its definition and the following extended definition incorporates various possible variants determined by specific parameters:

Definition 5 (Parametrization of the constructive back-door criterion (CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$))). *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. Let $\mathbf{A} \subseteq \mathbf{X} \cup \mathbf{Y}$, $\mathbf{B} \subseteq \mathbf{X}$, $\mathbf{C} \subseteq Dpcp(\mathbf{X}, \mathbf{Y})$. The set \mathbf{Z} satisfies CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$) relative to (\mathbf{X}, \mathbf{Y}) in \mathcal{G} if*

- (a') $\mathbf{Z} \subseteq \mathbf{V} \setminus De_{\overline{\mathbf{A}\mathbf{B}}}(\mathbf{PCP}(\mathbf{X}, \mathbf{Y}))$, and
- (b') \mathbf{Z} *d-separates* \mathbf{X} and \mathbf{Y} in the graph $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd, \mathbf{C}} := (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \rightarrow (\mathbf{PCP}(\mathbf{X}, \mathbf{Y}) \cup \mathbf{C})))$.

Clearly $CBC(\emptyset, \emptyset, \emptyset) = CBC$. The variants $CBC(\mathbf{X}, \emptyset, \emptyset)$ and $CBC(\emptyset, \emptyset, Dpcp(\mathbf{X}, \mathbf{Y}))$ might be the most interesting. Condition (a) of $CBC(\mathbf{X}, \emptyset, \emptyset)$ 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 2. Condition (a) of $CBC(\emptyset, \emptyset, Dpcp(\mathbf{X}, \mathbf{Y}))$ 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 $\mathcal{G}_{\mathbf{Y}}$. This would not lead to a valid criterion as it would allow an adjustment set $\{Z\}$ in the graph $\mathbf{X} \rightarrow \mathbf{Y} \rightarrow \mathbf{Z}$, 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 \mathbf{Y} are in $PCP(\mathbf{X}, \mathbf{Y})$ themselves. We can show that none of these modifications change the criterion:

Lemma 4. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG, and let $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. Let $\mathbf{A} \subseteq \mathbf{X} \cup \mathbf{Y}$, $\mathbf{B} \subseteq \mathbf{X}$, $\mathbf{C} \subseteq De(PCP(\mathbf{X}, \mathbf{Y}))$. 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 $\mathbf{Z} \subseteq \mathbf{V} \setminus Dpcp(\mathbf{X}, \mathbf{Y}) = \mathbf{V} \setminus De(PCP(\mathbf{X}, \mathbf{Y}))$ and $De_{\overline{\mathbf{AB}}}(PCP(\mathbf{X}, \mathbf{Y})) \subseteq De(PCP(\mathbf{X}, \mathbf{Y}))$, the condition (a') $\mathbf{Z} \subseteq \mathbf{V} \setminus De_{\overline{\mathbf{AB}}}(PCP(\mathbf{X}, \mathbf{Y}))$ is satisfied for CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$). \mathbf{Z} d -separates \mathbf{X} and \mathbf{Y} in $\mathcal{G}_{\mathbf{XY}}^{pbd}$, and thus also in $\mathcal{G}_{\mathbf{XY}}^{pbd, \mathbf{C}}$, because every edge or path of $\mathcal{G}_{\mathbf{XY}}^{pbd, \mathbf{C}}$ also exists in $\mathcal{G}_{\mathbf{XY}}^{pbd}$. 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 satisfies CBC (a), there exists a node $Z \in \mathbf{V} \setminus De_{\overline{\mathbf{AB}}}(PCP(\mathbf{X}, \mathbf{Y}))$ that is not in $\mathbf{V} \setminus Dpcp(\mathbf{X}, \mathbf{Y}) = \mathbf{V} \setminus De(PCP(\mathbf{X}, \mathbf{Y}))$. Then there must exist a proper causal path from \mathbf{X} to \mathbf{Y} on which a node $W \in \mathbf{V} \setminus \mathbf{X}$ is an ancestor of Z in \mathcal{G} , but not in $\mathcal{G}_{\overline{\mathbf{AB}}}$, i.e. there is a 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 choosen such that the length of the subpath between W and Z is minimal.

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

We will see the usefulness of the parametrization of the constructive back-door criterion in the proof of the main result of this subsection:

Theorem 1. *The constructive back-door criterion (CBC) is equivalent to the adjustment criterion (AC).*

Proof. First observe that the condition (a) of the adjustment criterion AC is identical to condition (a') of the constructive back-door criterion CBC($\mathbf{X}, \emptyset, \emptyset$). Assume conditions (a) and (b) of the adjustment criterion AC hold. Due to Lemma 4, 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 $\mathcal{G}_{\mathbf{XY}}^{pbd}$. Because $\mathcal{G}_{\mathbf{XY}}^{pbd}$ 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 $\mathcal{G}_{\mathbf{XY}}^{pbd}$ 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 $\mathcal{G}_{\mathbf{XY}}^{pbd}$ but open in \mathcal{G} . There can be two reasons why π is blocked in $\mathcal{G}_{\mathbf{XY}}^{pbd}$: (1) The path starts with an edge $X \rightarrow D$ that does not exist in $\mathcal{G}_{\mathbf{XY}}^{pbd}$. 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 $\mathcal{G}_{\mathbf{XY}}^{pbd}$. Then there must be a directed path from C to \mathbf{Z} via an edge $X \rightarrow D$ with $D \in An(\mathbf{Z}) \cap PCP(\mathbf{X}, \mathbf{Y})$, contradicting CBC (a). \square

5.2 CBC vs Pearl's back-door criterion

In this section we relate our constructive back-door criterion to the well-known back-door criterion by Pearl [3]:

Definition 6 (Pearl's back-door criterion (BC) [3]). *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 .

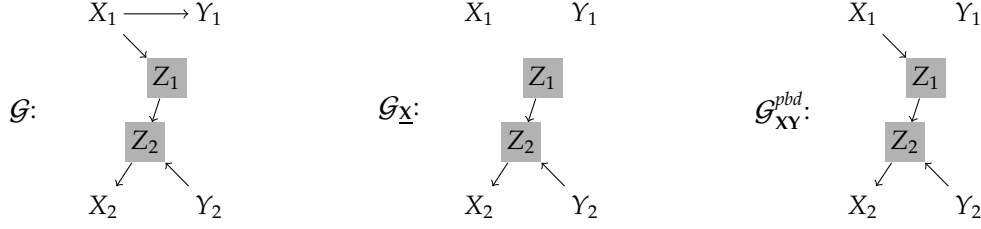


Figure 9: A DAG where for $\mathbf{X} = \{X_1, X_2\}$ and $\mathbf{Y} = \{Y_1, Y_2\}$, $\mathbf{Z} = \{Z_1, Z_2\}$ is a valid and minimum adjustment, but no set fulfills the back-door criterion [3] (Definition 6), and the parents of \mathbf{X} are not a valid adjustment set either.

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

In [1] 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 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 [16], 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.

The situation changes drastically if the effect of multiple exposures is estimated. Theorem 3.2.5 in [3] 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 9 shows, this is not true: the set $\mathbf{Z} = Pa(X_1, X_2) = \{Z_2\}$ is not an adjustment set according to $\{X_1, X_2\}$ and \mathbf{Y} . 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 $X_1 \rightarrow X_2 \rightarrow Z \leftarrow Y$ 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 $\mathcal{G}_{\mathbf{X}}$, this is not the graph $\mathcal{G}_{\mathbf{X}}$, 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 9.

Table 3 summarizes the relationships between CBC and the Pearl's back-door criterion.

6 Algorithms for testing and computing adjustment sets in DAGs

Having proved the constructive back-door criterion, we are now licensed to apply our separation algorithms from Section 3 to solve adjustment set problems. This works because the adjustment relative to \mathbf{X} and \mathbf{Y} in \mathcal{G} corresponds to an m -separator between \mathbf{X} and \mathbf{Y} in $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd}$ subject to the constraint given by CBC (a). Table 4 gives an overview of the relevant tasks.

A small difference exists between testing and constructing adjustment sets when handling CBC (a): Testing requires us to check if the given set \mathbf{Z} contains nodes of $Dpcp(\mathbf{X}, \mathbf{Y})$, whereas constructing requires that the returned set \mathbf{Z} must not contain any of the nodes in $Dpcp(\mathbf{X}, \mathbf{Y})$. The latter requirement can be straightforwardly implemented by imposing the constraint $\mathbf{Z} \subseteq \mathbf{R}' = \mathbf{R} \setminus Dpcp(\mathbf{X}, \mathbf{Y})$, which can be given as parameter to our separation algorithms.

Hence TESTADJ can be solved by testing if $\mathbf{Z} \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$ and if \mathbf{Z} is a d -separator in the proper back-door graph $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd}$ using algorithm TESTSEP. Since $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd}$ can be constructed from \mathcal{G} in linear time, the total time complexity

Statement for arbitrary DAGs and all sets \mathbf{Z} :		proof
\mathbf{Z} satisfies CBC	\nRightarrow \mathbf{Z} satisfies back-door	$\mathbf{Z} \leftarrow \mathbf{X} \rightarrow \mathbf{Y}$
\mathbf{Z} satisfies CBC	\nRightarrow $\exists \mathbf{Z}'$ satisfying back-door	see Figure 9
\mathbf{Z} satisfies CBC and \mathbf{Z} is minimal	\nRightarrow \mathbf{Z} satisfies back-door	see Figure 9
\mathbf{Z} satisfies CBC and \mathbf{Z} is minimal	\nRightarrow $\exists \mathbf{Z}'$ satisfying back-door	see Figure 9
Statement for all \mathbf{X} -loop-free DAGs (e.g., for singleton \mathbf{X}) and all sets \mathbf{Z} :		
\mathbf{Z} satisfies CBC	\Rightarrow \mathbf{Z} satisfies back-door	$\mathbf{Z} \leftarrow \mathbf{X} \rightarrow \mathbf{Y}$
\mathbf{Z} satisfies CBC	\Rightarrow $\exists \mathbf{Z}'$ satisfying back-door	via minimal $\mathbf{Z}' \subseteq \mathbf{Z}$
\mathbf{Z} satisfies CBC and \mathbf{Z} is minimal	\Rightarrow \mathbf{Z} satisfies back-door	see [1]
\mathbf{Z} satisfies CBC and \mathbf{Z} is minimal	\Rightarrow $\exists \mathbf{Z}'$ satisfying back-door	$\mathbf{Z}' = \mathbf{Z}$

Table 3: A summary of the relationship between the existence of a Pearl back-door-adjustment set and the existence of an CBC-adjustment set in unconstrained DAGs and \mathbf{X} -loop-free DAGs. Symbol \nRightarrow 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 "back-door" and in the right hand sides "back-door" by "CBC", then the corresponding implications are always true.

		Runtime
Verification: For given $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and constraint \mathbf{I} decide if ...		
TESTADJ	\mathbf{Z} is an adjustment for (\mathbf{X}, \mathbf{Y})	$O(n + m)$
TESTMINADJ	$\mathbf{Z} \supseteq \mathbf{I}$ is an adjustment for (\mathbf{X}, \mathbf{Y}) and \mathbf{Z} is ...	
	\mathbf{I} -minimal	$O(n^2)$
	strongly-minimal	$O(n^2)$
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 $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$O(n + m)$
FINDMINADJ	adjustment \mathbf{Z} for (\mathbf{X}, \mathbf{Y}) with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...	
	\mathbf{I} -minimal	$O(n^2)$
	strongly-minimal	NP-hard
FINDMINCOSTADJ	adjustment \mathbf{Z} for (\mathbf{X}, \mathbf{Y}) with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ which is ...	
	\mathbf{I} -minimum	$O(n^3)$
	strongly-minimum	$O(n^3)$
Enumeration: For given $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ enumerate all ...		Delay
LISTADJ	adjustments \mathbf{Z} for (\mathbf{X}, \mathbf{Y}) with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$O(n(n + m))$
LISTMINADJ	\mathbf{I} -minimal adjustments \mathbf{Z} with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$	$O(n^3)$

Table 4: Definitions of algorithmic tasks related to adjustment in DAGs, similar to the definitions of Table 2, and the associated time complexities provided in this section. 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. A minimum adjustment 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 will output \perp if no set fulfilling the listed condition exists. Delay complexity for e.g. LISTMINADJ refers to the time needed to output one solution when there can be exponentially many solutions (see [26]).

of this algorithm is $O(n + m)$.

TESTMINADJ can be solved by testing again if $Z \cap Dpcp(X, Y) = \emptyset$ and calling TESTMINSEP to verify that Z is minimal within the back-door graph \mathcal{G}_{XY}^{pbd} . This leads to a runtime of $O(n^2)$ which is optimal for dense graphs. Alternatively TESTMINSEPARSE with its runtime of $O(|Ant(X \cup Y)| \cdot |\text{Edges of } (\mathcal{G}_{XY}^{pbd})^a|) = O(n(n + m))$ can be used in sparse graphs. It is worth noting that since the back-door graph is formed by removing edges, it is even sparser than the input graph. This approach can only work, 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 which generalizes results of [14]:

Corollary 5. *If no single node Z can be removed from an adjustment set Z such that the resulting set $Z' = Z \setminus Z$ is no longer an adjustment set, then Z is minimal.*

The problem FINDADJ can be solved by a closed form solution. For a DAG $\mathcal{G} = (V, E)$ and constraints I, R we define the set

$$Adj(X, Y) = An(X \cup Y \cup I) \cap R \setminus (X \cup Y \cup Dpcp(X, Y)).$$

Theorem 2. *Let $\mathcal{G} = (V, E)$ be a DAG, let $X, Y \subseteq V$ be distinct node sets and I, R constraining node sets with $I \subseteq R \setminus (X \cup Y \cup Dpcp(X, Y))$. Then the following statements are equivalent:*

1. *There exists an adjustment Z in \mathcal{G} w.r.t. X and Y with $I \subseteq Z \subseteq R$.*
2. *$Adj(X, Y)$ is an adjustment w.r.t. X and Y .*
3. *$Adj(X, Y)$ d -separates X and Y in the proper back-door graph \mathcal{G}_{XY}^{pbd} .*

Proof. The implication (3) \Rightarrow (2) follows directly from the criterion Def. 4 and the definition of $Adj(X, Y)$. Since the implication (2) \Rightarrow (1) is obvious, it remains to prove (1) \Rightarrow (3).

Assume there exists an adjustment set Z_0 w.r.t. X and Y . From Theorem 1 we know that $Z_0 \cap Dpcp(X, Y) = \emptyset$ and that Z_0 d -separates X and Y in \mathcal{G}_{XY}^{pbd} . Our task is to show that $Adj(X, Y)$ d -separates X and Y in \mathcal{G}_{XY}^{pbd} . This follows from Lemma 1 used for the proper back-door graph \mathcal{G}_{XY}^{pbd} if we take $I' = I$, $R' = R \setminus (X \cup Y \cup Dpcp(X, Y))$. \square

From Equation (3) and the definition $Dpcp(X, Y) = De(PCP(X, Y))$ we then obtain immediately:

Corollary 6. *Given two distinct sets $X, Y \subseteq V$, $Adj(X, Y)$ can be found in $O(n + m)$ time.*

The remaining problems, FINDMINADJ, FINDMINCOSTADJ, LISTADJ and LISTMINADJ can be solved using the corresponding algorithms for finding, resp. listing m -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 2 and Table 4. The NP-hardness of finding strongly-minimal adjustment sets follows from Proposition 7 and the fact that the graph constructed in the proof of the proposition 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.

7 Extending the CBC

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 [3] is a complete method that characterizes all identifiable causal effect, but which comes at a price of substantially increased formula and runtime complexity. In this section, we however show that many cases in which covariate adjustment is not applicable do not require the power of the do-calculus either.

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

7.1 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 there is a common ancestor of X and Y , then this case is covered by the CBC; however, if there is a reverse causal path from Y to X , then the (void) effect is not obtainable through the adjustment formula. In both of these cases, however, we simply have $P(y \mid do(x)) = P(y)$, 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 11. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG and 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 $\mathcal{G}_{\overline{\mathbf{R}}}$, expressed symbolically as*

$$(\mathbf{Y} \perp\!\!\!\perp \mathbf{X})_{\mathcal{G}_{\overline{\mathbf{R}}}} \quad (4)$$

if and only if the effect of intervention of \mathbf{X} on \mathbf{Y} is given by the plain formula $P(\mathbf{y} \mid do(\mathbf{x})) = P(\mathbf{y})$. Particularly, if $Y \in An(X)$ then $(Y \perp\!\!\!\perp X)_{\mathcal{G}_{\overline{\mathbf{R}}}}$ 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; for the precise definition of the do-calculus rules see Theorem 4 in the appendix).

The completeness of the statement can be shown similarly to the completeness of the adjustment criterion [10]. If \mathbf{Y} and \mathbf{X} are not d -separated in $\mathcal{G}_{\overline{\mathbf{R}}}$, there exists a shortest causal path $X \rightarrow \dots \rightarrow Y$ 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

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

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 consistent with \mathcal{G} (though not faithful, but faithfulness is not required) and we have

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

□

7.2 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 12 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 when 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 [3, 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 intervention $do(X = x)$ can be given both by the plain expression $P(y \mid do(x)) = P(y)$ and by adjustment in parents of X .

Proposition 12. *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:

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

□

If some of parents $Pa(X)$ are unobserved, the causal effect might not be identifiable at all, like e.g. in the DAG \mathcal{G}_1 in Figure 2. To decide if the effect is identifiable in such a case, one can use the CBC criterion which, like for \mathcal{G}_3 in Figure 2, can confirm the identifiability. However, while the CBC is complete to decide if the effect is expressible via covariate adjustment it is not complete to decide if the effect is identifiable or not. For an example, see the DAG \mathcal{G}_2 in Figure 2. To solve the identification problem in this case, when the CBC does not work, one has to use a complete criterion, like this based on the do-calculus.

7.3 Identification when \mathbf{X} and \mathbf{Y} partition \mathbf{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.

Lemma 5. *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*

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

(b) *If no edge $X \rightarrow Y$ 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 $Y \rightarrow X$ 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. For the sake of brevity we will identify values and nodes, i.e. write $P(\mathbf{Y} \mid do(\mathbf{X}))$ to mean $P(\mathbf{y} \mid do(\mathbf{x}))$.

To show (a) let Y_1, Y_2, \dots, Y_k be a topological ordering of \mathbf{Y} , i.e. $\mathbf{Y} = \{Y_1, \dots, Y_k\}$, and for $i < j$ we have: $Y_j \notin An(Y_i)$, $\mathbf{Y} \cap An(Y_i) = \{Y_1, \dots, Y_i\} \cap An(Y_i)$, and $\mathbf{Y} \cap Pa(Y_i) = \{Y_1, \dots, Y_{i-1}\} \cap Pa(Y_i)$. We calculate $P(\mathbf{Y} \mid do(\mathbf{X}))$ using rules of the do-calculus:

$$\begin{aligned} P(\mathbf{Y} \mid do(\mathbf{X})) &= P(Y_1 \mid do(\mathbf{X}))P(Y_2 \mid do(\mathbf{X}), Y_1) \dots P(Y_k \mid do(\mathbf{X}), Y_1, \dots, Y_{k-1}) \\ &= P(Y_1 \mid do(\mathbf{X} \cap Pa(Y_1)))P(Y_2 \mid do(\mathbf{X} \cap Pa(Y_2)), Y_1) \dots P(Y_k \mid do(\mathbf{X} \cap Pa(Y_k)), Y_1, \dots, Y_{k-1}) \\ &\quad \text{Due to rule 3, since } (Y_i \perp\!\!\!\perp \mathbf{X} \setminus Pa(Y_i) \mid \mathbf{X} \cap Pa(Y_i), Y_1, \dots, Y_{i-1}) \text{ in } \mathcal{G}_{\overline{\mathbf{X} \cap Pa(Y_i)} \overline{(\mathbf{X} \setminus Pa(Y_i))} (Y_1, \dots, Y_{i-1})}. \end{aligned}$$

Explanation: blocking the nodes as specified above, there cannot be an open d -path between Y_i and $X \in \mathbf{X} \setminus Pa(Y_i)$ starting with $Y_i \leftarrow \mathbf{X}$ or $Y_i \leftarrow \mathbf{Y}$. If $X \in De(Y_i)$, X cannot be an ancestor of an Y_1, \dots, Y_{i-1} , so the edge incoming into X is deleted in $\mathcal{G}_{\overline{\mathbf{X} \cap Pa(Y_i)} \overline{(\mathbf{X} \setminus Pa(Y_i))} (Y_1, \dots, Y_{i-1})$, so there is no d -path $Y_i \rightarrow \dots \rightarrow X$. No d -path starting with $Y_i \rightarrow \dots \rightarrow \mathbf{Y} \leftarrow$ is open, as no descendant of Y_i can be in $\mathbf{X} \cap Pa(Y_i), Y_1, \dots, Y_{i-1}$. These two cases cover all d -paths starting with $Y_i \rightarrow$ as there are no nodes not in $\mathbf{X} \cup \mathbf{Y}$.

$$\begin{aligned} &= P(Y_1 \mid \mathbf{X} \cap Pa(Y_1))P(Y_2 \mid \mathbf{X} \cap Pa(Y_2), Y_1) \dots P(Y_k \mid \mathbf{X} \cap Pa(Y_k), Y_1, \dots, Y_{k-1}) \\ &\quad \text{Due to rule 2, since } (Y_i \perp\!\!\!\perp \mathbf{X} \cap Pa(Y_i) \mid Y_1, \dots, Y_{i-1}) \text{ in } \mathcal{G}_{\overline{\mathbf{X} \cap Pa(Y_i)}}. \end{aligned}$$

Explanation: because under these constraints there cannot be an open d -path between Y_i and $X \in \mathbf{X} \cap Pa(Y_i)$ starting with $Y_i \leftarrow \mathbf{X}$ or $Y_i \leftarrow \mathbf{Y}$ as outgoing edges of $\mathbf{X} \cap Pa(Y_i)$ are removed and Y_1, \dots, Y_{i-1} are observed. There is no causal path $Y_i \rightarrow \dots \rightarrow Pa(Y_i)$ in a DAG. No d -path starting in $Y_i \rightarrow \dots \rightarrow$ and containing a collider V is open, as no descendant of Y_i can be in Y_1, \dots, Y_{i-1} .

$$\begin{aligned} &= P(Y_1 \mid \mathbf{X} \cap Pa(Y_1))P(Y_2 \mid \mathbf{X} \cap Pa(Y_2), Y_1 \cap Pa(Y_2)) \dots P(Y_k \mid \mathbf{X} \cap Pa(Y_k), \{Y_1, \dots, Y_{k-1}\} \cap Pa(Y_k)) \\ &\quad \text{Due to rule 1, because } (Y_i \perp\!\!\!\perp \{Y_1, \dots, Y_{i-1}\} \setminus Pa(Y_i) \mid (\mathbf{X} \cup \{Y_1, \dots, Y_{i-1}\}) \cap Pa(Y_i)) \text{ in } \mathcal{G}_{\overline{\mathbf{X} \cap Pa(Y_i)}}. \end{aligned}$$

Explanation: under these constraints there cannot be an open d -path between Y_i and $Y_j \notin Pa(Y_i)$ with $j < i$ starting with $Y_i \leftarrow \mathbf{X}$ or $Y_i \leftarrow \mathbf{Y}$ as outgoing edges of $\mathbf{X} \cap Pa(Y_i)$ are removed and $\{Y_1, \dots, Y_{i-1}\} \cap Pa(Y_i)$ are observed. There is no causal path $Y_i \rightarrow \dots \rightarrow Y_j$ due to the topological ordering. No d -path $Y_i \rightarrow \dots \rightarrow V \leftarrow$ containing a collider is open, as no descendant of \mathbf{Y}_i can be in $Pa(Y_i)$.

$$\begin{aligned} &= P(Y_1 \mid \mathbf{X} \cap Pa(Y_1))P(Y_2 \mid \mathbf{X} \cap Pa(Y_2), \mathbf{Y} \cap Pa(Y_2)) \dots P(Y_k \mid \mathbf{X} \cap Pa(Y_k), \mathbf{Y} \cap Pa(Y_k)) \\ &= \prod_i P(Y_i \mid Pa(Y_i)). \end{aligned}$$

This completes the proof for (a). To see statement (b) note, that above we have shown

$$P(\mathbf{y} \mid do(\mathbf{x})) = P(Y_1 \mid \mathbf{X} \cap Pa(Y_1))P(Y_2 \mid \mathbf{X} \cap Pa(Y_2), Y_1) \dots P(Y_k \mid \mathbf{X} \cap Pa(Y_k), Y_1, \dots, Y_{k-1}).$$

By the assumption of (b) we have $\mathbf{X} \cap Pa(Y_i) = \emptyset$, so $P(\mathbf{y} \mid do(\mathbf{x})) = P(Y_1)P(Y_2 \mid Y_1) \dots P(Y_k \mid Y_1, \dots, Y_{k-1}) = P(\mathbf{y})$. To see (d) we notice that, since no node is outside of $\mathbf{X} \cup \mathbf{Y}$, the only possible adjustment set is $\mathbf{Z} = \emptyset$. Finally, we prove (c). An adjustment set \mathbf{Z} always satisfies the first condition of the CBC. The backdoor graph is formed by removing all edges $\mathbf{X} \rightarrow \mathbf{Y}$ as those edges form a causal path of length one. Edges $\mathbf{X} \rightarrow \mathbf{X}$ or $\mathbf{Y} \rightarrow \mathbf{Y}$ do not affect d -connectedness between \mathbf{X} and \mathbf{Y} . Thus \mathbf{Z} is an adjustment set, if and only if no edge $\mathbf{Y} \rightarrow \mathbf{X}$ exists. \square

When $\mathbf{V} = \mathbf{X} \cup \mathbf{Y}$, the R package `causaleffect` [32], which we used in our experiments that we describe later in this paper, returns the formula $P(\mathbf{y} \mid do(\mathbf{x})) = \prod_{Y \in \mathbf{Y}} P(Y \mid An(Y) \setminus Y)$ when configured to be fast and find any identification formula rather than a short one. It is easy to see that the above proof also holds for $An(Y) \setminus Y$ rather than $Pa(Y)$, since both sets block all paths into a node $Y \in \mathbf{Y}$, so it is worth to mention that $\prod_{Y \in \mathbf{Y}} P(Y \mid An(Y) \setminus Y) = \prod_{Y \in \mathbf{Y}} P(Y \mid Pa(Y))$.

8 Empirical analysis of identifiability by adjustment

As mentioned before, not all identifiable total effects are identifiable via covariate 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 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 [3]; (2) the CBC; (3) an extended version of the CBC that also covers the special cases discussed in Section 7; 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 6) in our analysis because it is still very present in the applied literature on DAGs (e.g., [4]) whereas the generalized version 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 how often the back-door criterion fails to find an adjustment set when our CBC criterion succeeds. In Section 5.2 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 7 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.

8.1 Instance generation

We evaluate identifiability on random DAGs, which we generate as described in Section 4. The random DAGs are generated with different numbers of variables \mathbf{V}

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

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

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

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 $P(\text{unobserved})$ 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 $\mathbb{E}[|\mathbf{R}|] > n \cdot (1 - P(\text{unobserved}))$, with the difference being very small for $n \gg 2k$, but substantial for $n \gtrsim 2k$. 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 $P(\text{unobserved})$ – the case discussed already in Section 7.3.

We perform experiments for each parametrization tuple

$$(n, l, k, P(\text{unobserved})), \tag{5}$$

where, recall, l determines the probability $P(\text{edge})$ as described in Section 4. In this section we will report our results in detail only for $P(\text{unobserved}) \in \{0, 0.75\}$. The remaining cases are shown in the appendix.

We generated 10 000 graphs for each parameter tuple using the function `GraphGenerator.randomDAG` of our `DAGitty` library [33] in `node.js`. Figure 10 shows example instances sampled for $n = 10$ and illustrates the four cases we are interested in.

8.2 Algorithms

The main goal of our experiments was to examine the influence of the instance complexity, like 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:

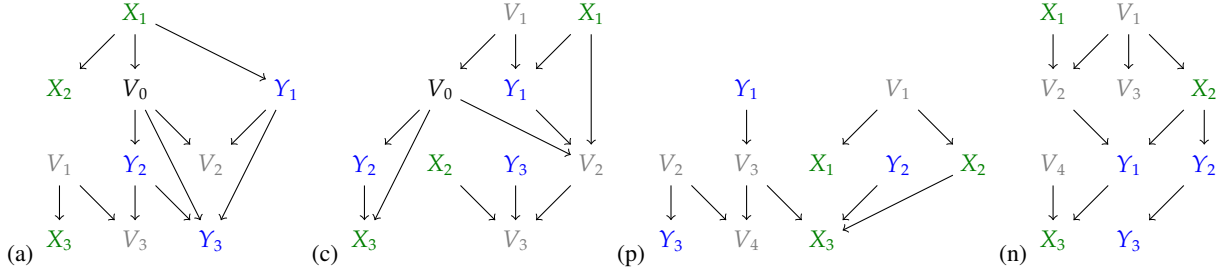


Figure 10: Example DAGs sampled for the parameter tuple $n = 10, P(\text{edge}) = 2/9, P(\text{unobserved}) = 0.5$, and $k = |\mathbf{X}| = |\mathbf{Y}| = 3$. Nodes are relabeled such that exposures are called X_1, X_2, X_3 , outcomes are called Y_1, Y_2, Y_3 , and all nodes except V_0 are unobserved. Case (a) adjustment 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) complex 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) plain 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) non-identifiable.

CBC: our constructive back-door criterion (Definition 4, Theorem 2). We used our `DAGitty` package, specifically the function `GraphAnalyzer.canonicalAdjustmentSet`, which implements algorithm `FIND-ADJ` based on CBC. We also tested another implementation of our CBC criterion, the `gac` function of the R package `pcalg` [34].

CBC⁺: combination of CBC and plain formula (Proposition 11). We implement the plain formula using the `DAGitty` function `GraphAnalyzer.dConnected`, which implements algorithm `TESTSEP` (Proposition 1).

BC: Pearl’s back-door criterion (Definition 6). 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} [19]. This means we can implement BC by trivial post-processing of the CBC output.

IDC: general identifiability as determined by do-calculus (see [3, Chapter 3.4.2]; we also recall the rules in Theorem 4 in the appendix). Specifically, we use the IDC algorithm by Shpitser and Pearl [24], which is well known to be complete for the identification of causal effects [22, 23], 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` [32]. Due to high time complexity, we were only able to use this algorithm for small instances.

8.3 Results

The results for all methods and parameters n, k, l described above are shown in Table 5 (for the case $P(\text{unobserved}) = 0$) and in Table 6 ($P(\text{unobserved}) = 0.75$). We now discuss the results in more detail.

Identification by adjustment sets or plain formulas Tables 5 and 6 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 11 summarizes the counts for CBC and CBC⁺ reported in Table 5 and 6 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 7.3. Moreover, for parameter values $l = 10$ and $l = 20$ the individual probabilities for edge selection, $P(\text{edge}) = \max\{l/(n-1), 1\}$, 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.

n	k	$l = 2$			$l = 5$			$l = 10$			$l = 20$		
		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: Numbers of instances for probability $P(\text{unobserved}) = 0$, i.e. all variables are unobserved, that are identifiable by use of BC, CBC, or CBC⁺ (as defined in Table 5). Gray cells highlight where the BC failed to identify more than 4% of those cases that CBC could identify. Since all variables are observed, all instances are identifiable, thus IDC is not used in this Table.

n	k	$l = 2$				$l = 5$				$l = 10$				$l = 20$			
		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 6: Numbers of instances for $P(\text{unobserved}) = 0.75$ that are identifiable by use of BC, CBC, CBC⁺ (as defined in Table 5), and by the complete identification do-calculus based algorithm (IDC). Gray cells highlight where the BC failed to identify more than 4% of those cases that CBC could identify. Due to high time complexity, we were unable to run the IDC algorithm on instances labelled with “-”.

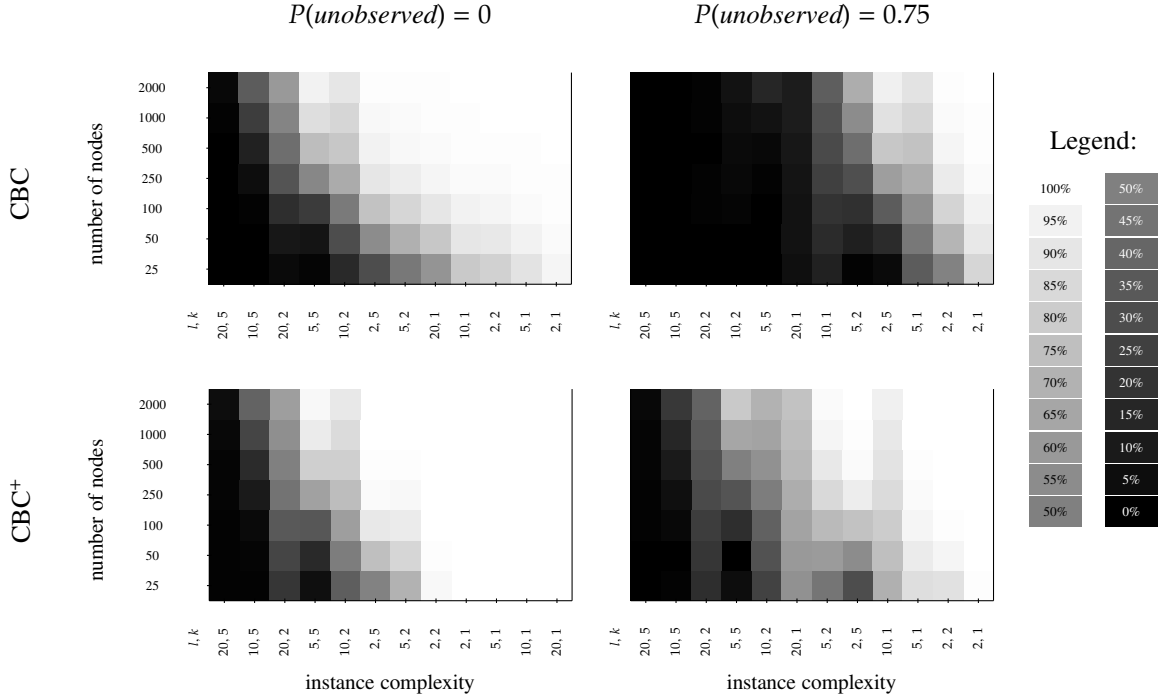


Figure 11: Heatmaps visualizing the number of instances that are identifiable by use our constructive back-door criterion (CBC, top row), and by the use of CBC or the plain formula (CBC⁺, bottom row). Black squares depict the worst case in which 0% of instances are identifiable by use of CBC or CBC⁺, 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.

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. The plain formula solves a surprisingly high number of cases for which adjustment does not work. This can be seen especially in dense graphs, e.g., DAGs in which each node has $l = 20$ neighbours on average, and for singleton \mathbf{X} and \mathbf{Y} , i.e. $k = 1$. For example, for $l = 20, k = 1, P(\text{unobserved}) = 0.75$, 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 11 as gray squares in the columns labeled as $(l, k) = (20, 1)$ (case: $P(\text{unobserved}) = 0.75$).

The difficulty of identification by adjustment grows with increasing k and l , but it decreases with increasing number of nodes n , both for $P(\text{unobserved}) = 0$ and for 0.75. In Figure 11, 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 $P(\text{unobserved}) = 0$ the counts grows very slowly with n reaching the maximum value of 3% of identifiable graphs for $n = 2000$; for $P(\text{unobserved}) = 0.75$ almost no instances are identifiable by adjustment (compare the upper panels in Fig. 11). However, as we can see in Table 6, for $n = 250$ only 2.5% of cases are identifiable at all. Figures 12 and 13 summarize the difficulty of identification stratified by n (Figure 12) and l (Figure 13), 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 and 6 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 5.2, for singletons $\mathbf{X} = \{X\}, \mathbf{Y} = \{Y\}$ the counts for BC and CBC are

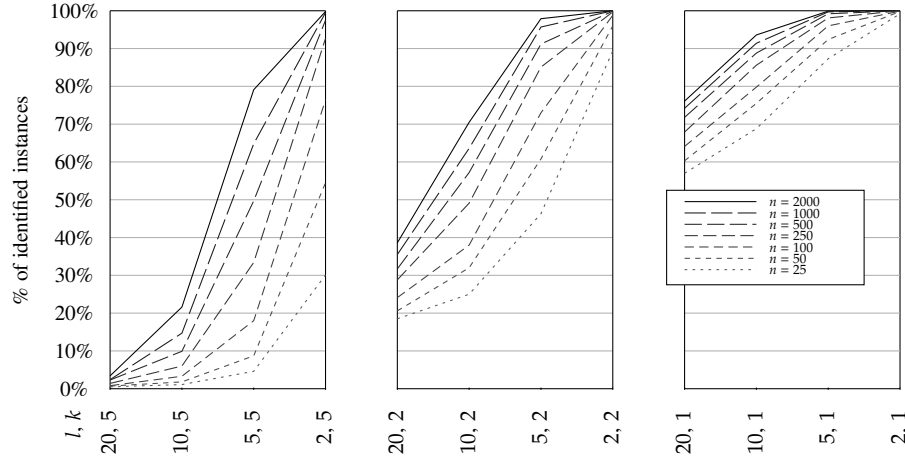


Figure 12: Case $P(\text{unobserved}) = 0.75$: Percent of identifiable graphs for fixed numbers of nodes $n \in \{25, 50, 100, 250, 500, 1000, 2000\}$ and with varying expected number of node neighbors l and cardinalities $|\mathbf{X}| = |\mathbf{Y}| = k$. The horizontal axis is labeled by (l, k) sorted lexicographically. The curves show the data for CBC^+ , i.e., for instances identifiable by adjustment or by plain formula.

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 $n = 10, |\mathbf{X}| = |\mathbf{Y}| = 3, m \approx n$, and $P(\text{unobserved}) = 0$, 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 $P(\text{edge})$ and $P(\text{unobserved})$.

Complete identification by do-calculus compared to identification by adjustment or plain formula As explained above, in small graphs we checked for general identifiability of causal effects using the IDC algorithm [24]. Results are shown for $P(\text{unobserved}) = 0.75$ and $n \leq 250$ in Table 6. 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 $P(\text{unobserved}) = 0$ the causal effect is always identifiable, so we skip the counts for IDC in Table 5.

The cases with $n = 10, k = |\mathbf{X}| = |\mathbf{Y}| = 5$ (Table 6) 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 6) this number decreases with an increased 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 $P(\text{unobserved})$, similarly to the cases in Table 5, and all graphs must be identifiable as shown in Section 7.3.

Figures 12 and 13 present the data for CBC^+ (the same data as in the lower right panel of Figure 11) in comparison to identification by IDC. As we observed in Figure 11, the most difficult case for CBC^+ is $(l, k) = (20, 5)$ and the difficulty decreases with k and l when n is fixed (Figure 12). The situation is very similar for IDC. In Figure 13, we see that identifiability for both CBC^+ and IDC grows roughly in parallel.

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. Similarly to the results for adjustment sets, one can see that with increasing $|\mathbf{X}|, |\mathbf{Y}|, l$ the number of identifiable graphs decreases, when $P(\text{unobserved}) > 0$. Specifically, with $|\mathbf{X}| = |\mathbf{Y}| = 1$, all causal effects are already identified by plain formula or adjustment without involving the IDC algorithm.

n	k	$l = 2$			$l = 5$			$l = 10$			$l = 20$		
		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 7: 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 $P(\text{unobserved} = 0.75)$. 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. The values for IDC include the time spent on loading the generated graphs e.g., for $l = 20$, ranging from 0.7 ms ($n = 10$) to 294 ms ($n = 250$).

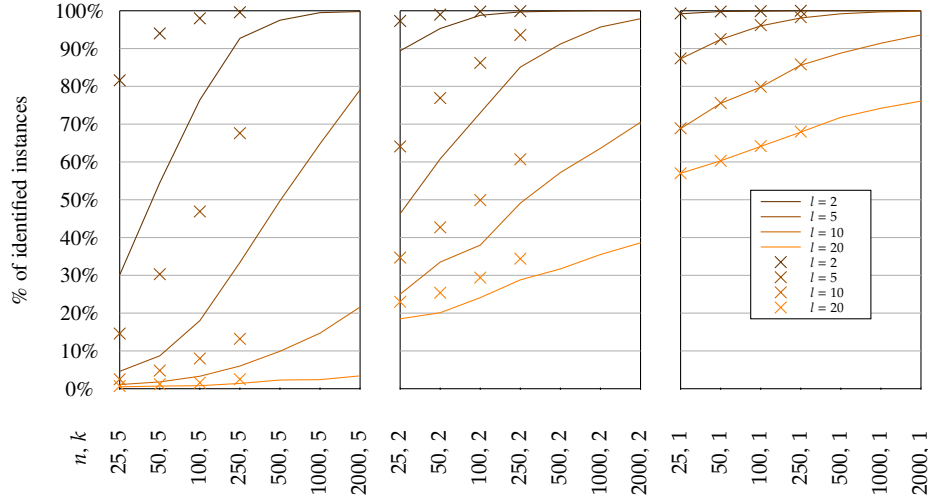


Figure 13: Case $P(\text{unobserved}) = 0.75$: Percent of identifiable graphs for fixed density parameter values $l \in \{2, 5, 10, 20\}$ and with varying the number of nodes n and cardinalities $|\mathbf{X}| = |\mathbf{Y}| = k$. The horizontal axis is labeled by (n, k) sorted lexicographically. The 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 $n \geq 500$.

Comparative runtimes of the algorithms Figure 14 (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 $P(\text{unobserved}) = 0.75$. Graphs with a lower probability $P(\text{unobserved})$ 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 $O(n + m)$ due to inefficient set operations, or that the time actually only depends on $O(\text{An}(\mathbf{X}, \mathbf{Y}))$ which can be much smaller than $O(m)$ 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 red plot in Figure 14. Since the IDC algorithm is far more complex than the expression of Theorem 2, 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 also investigated a different implementation of the CBC in the R package `pcalg` [34]. 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 $\text{Adj}(\mathbf{X}, \mathbf{Y})$ 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 $G = \mathcal{G}$, $\text{GNoInX} = \mathcal{G}_{\bar{\mathbf{X}}}$, $\text{GNoOutX} = \mathcal{G}_{\bar{\mathbf{X}}}$ 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 plotted in purple in Figure 14. 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 reasonable time.

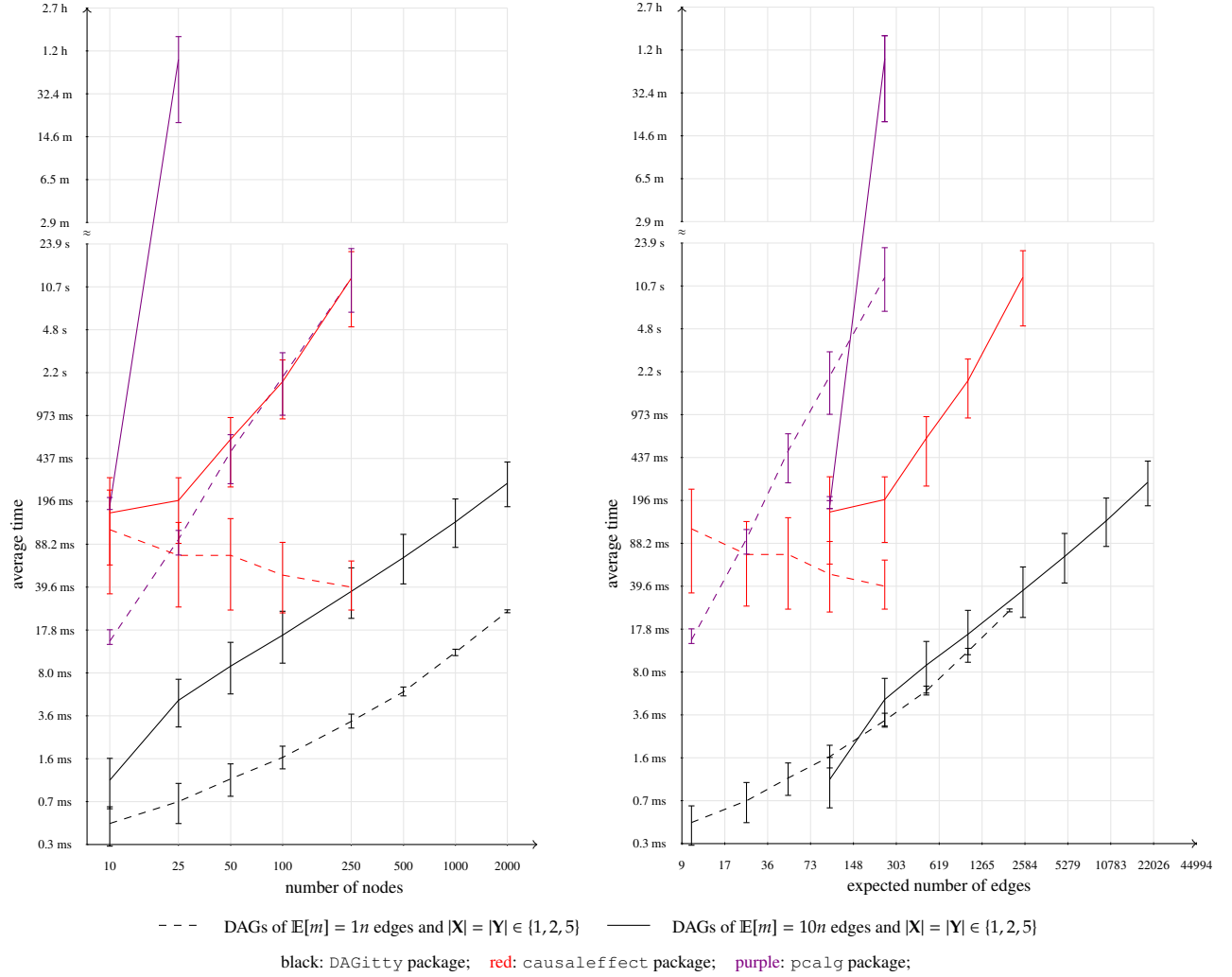


Figure 14: Average time needed to find an adjustment set and verify it according to the CBC in a single graph with $P(\text{unobserved}) = 0.75$, values $n \in \{10, 25, 50, 100, 250, 500, 1000, 2000\}$, $\mathbb{E}[m] \in \{1n, 10n\}$, and various cardinalities k of \mathbf{X}, \mathbf{Y} . The left 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 violet – the data for the `gac` function of the R package `pcalg`. Error bars show the minimum and maximum time taken. The plot shows that all small graphs ($n \leq 100$) are solved nearly instantaneously (time $\leq 100\text{ms}$) 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.

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 do-calculus, at least in the random graphs we tested. This finding is reassuring given that the implementation of our algorithmic framework in DAGitty is the only identification method of those we tested that is applicable to large graphs.

9 Adjustment in MAGs

In this final section, we generalize our complete constructive criterion for identifying covariate adjustment sets from DAGs to MAGs, making our algorithmic framework applicable to this class of graphical models as well. Two examples may illustrate why this generalization is not trivial. First, take $\mathcal{G} = X \rightarrow Y$. If \mathcal{G} is interpreted as a DAG, then the empty set is valid for adjustment. If \mathcal{G} is however taken as a MAG, then there exists no adjustment set as \mathcal{G} represents among others the DAG $U \rightarrow X \rightarrow Y$ where U is an unobserved confounder. Second, take $\mathcal{G} = A \rightarrow X \rightarrow Y$. In that case, the empty set is an adjustment set regardless of whether \mathcal{G} is interpreted as a DAG or a MAG. The reasons will become clear as we move on. First, let us recall the semantics of a MAG. The following definition can easily be given for AGs in general, but we do not need this generality for our purpose.

Definition 7 (DAG representation by MAGs [12]). *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a DAG, and let $\mathbf{S}, \mathbf{L} \subseteq \mathbf{V}$. The MAG $\mathcal{M} = \mathcal{G}_{\mathbf{S}}^{\mathbf{L}}$ is a graph with nodes $\mathbf{V} \setminus (\mathbf{S} \cup \mathbf{L})$ and defined as follows. (1) Two nodes U and V are adjacent in $\mathcal{G}_{\mathbf{S}}^{\mathbf{L}}$ if they cannot be m -separated by any \mathbf{Z} with $\mathbf{S} \subseteq \mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{L}$ in \mathcal{G} . (2) The edge between U and V is*

$U - V$ if $U \in \text{An}(\mathbf{S} \cup V)$ and $V \in \text{An}(\mathbf{S} \cup U)$;

$U \rightarrow V$ if $U \in \text{An}(\mathbf{S} \cup V)$ and $V \notin \text{An}(\mathbf{S} \cup U)$;

$U \leftrightarrow V$ if $U \notin \text{An}(\mathbf{S} \cup V)$ and $V \notin \text{An}(\mathbf{S} \cup U)$.

We call \mathbf{L} latent variables and \mathbf{S} selection variables. We say there is selection bias if $\mathbf{S} \neq \emptyset$.

Hence, every MAG represents an infinite set of underlying DAGs that all share the same ancestral relationships.

Lemma 6 (Preservation of separating sets [12]). *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 [35]². 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 [36].

We now extend the concept of adjustment to MAGs in the usual way [16].

Definition 8 (Adjustment in MAGs). *Given a MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ and two variable sets $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$, $\mathbf{Z} \subseteq \mathbf{V}$ is an adjustment set for (\mathbf{X}, \mathbf{Y}) in \mathcal{M} if for any DAG $\mathcal{G} = (\mathbf{V}', \mathbf{E}')$ for which $\mathcal{G}_{\mathbf{L}}^{\mathbf{L}} = \mathcal{M}$ with $\mathbf{L} = \mathbf{V}' \setminus \mathbf{V}$ the set \mathbf{Z} is an adjustment set for (\mathbf{X}, \mathbf{Y}) in \mathcal{G} .*

This definition is equivalent to requiring that $P(\mathbf{y} \mid \text{do}(\mathbf{x}))$ is equal to $\sum_{\mathbf{z}} P(\mathbf{y} \mid \mathbf{x}, \mathbf{z})P(\mathbf{z})$ for every probability distribution $P(\mathbf{v}')$ consistent with a DAG $\mathcal{G} = (\mathbf{V}', \mathbf{E}')$ for which $\mathcal{G}_{\mathbf{L}}^{\mathbf{L}} = \mathcal{M}$ 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 $P(\mathbf{y} \mid \text{do}(\mathbf{x}))$ 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 [36]. 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} .

There is no reason to further generalize this definition to ancestral graphs rather than MAGs as $\mathcal{G}_{\mathbf{S}}^{\mathbf{L}}$ is always a MAG and every ancestral graph can be converted to a MAG with the same conditional independences [12].

² A counterexample is the graph $A \leftarrow X \rightarrow Y$, 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 $\mathbf{Y} \perp\!\!\!\perp \mathbf{S} \mid \mathbf{X}$ [36], which is so restrictive that most statisticians would probably not even speak of “selection bias” anymore in such a case.

9.1 Adjustment amenability

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

Definition 9 (Visible edge [35]). *Given a MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$, an edge $X \rightarrow D \in \mathbf{E}$ is called visible if in all DAGs $\mathcal{G} = (\mathbf{V}', \mathbf{E}')$ with $\mathcal{G}^{\perp}_{\mathbf{S}} = \mathcal{M}$ for some $\mathbf{S}, \mathbf{L} \subseteq \mathbf{V}'$, all d -connected walks between X and D in \mathcal{G} that contain only nodes of $\mathbf{S} \cup \mathbf{L} \cup X \cup D$ are directed paths.*

Intuitively, an invisible directed edge $X \rightarrow D$ 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 7 (Graphical conditions for edge visibility [35]). *In a MAG $\mathcal{M} = (\mathbf{V}, \mathbf{E})$, an edge $X \rightarrow D$ in \mathbf{E} is visible if and only if there is a node A not adjacent to D where (1) $A \rightarrow X \in \mathbf{E}$ or $A \leftrightarrow X \in \mathbf{E}$, or (2) there is a collider path $A \leftrightarrow V_1 \leftrightarrow \dots \leftrightarrow V_n \leftrightarrow X$ or $A \rightarrow V_1 \leftrightarrow \dots \leftrightarrow V_n \leftrightarrow X$ where all V_i are parents of D .*

Definition 10. *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 8. *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 adjustment set for (\mathbf{X}, \mathbf{Y}) in \mathcal{M} .*

Proof. If the first edge $X \rightarrow D$ 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} . \square

Let $N(V)$ 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( $\mathcal{M}, \mathbf{X}, \mathbf{Y}$ )
  for all  $D$  in  $Ch(\mathbf{X}) \cap PCP(\mathbf{X}, \mathbf{Y})$  do
     $\mathbf{C} \leftarrow \emptyset$ ;  $\mathbf{A} \leftarrow \emptyset$ 
    function CHECK( $V$ )
      if  $\mathbf{C}[V]$  then return  $\mathbf{A}[V]$ 
       $\mathbf{C}[V] \leftarrow \text{true}$ 
       $\mathbf{A}[V] \leftarrow ((Pa(V) \cup Sp(V)) \setminus N(D) \neq \emptyset)$ 
      for all  $W \in Sp(V) \cap Pa(D)$  do
        if CHECK( $W$ ) then  $\mathbf{A}[V] \leftarrow \text{true}$ 
      return  $\mathbf{A}[V]$ 
    for all  $X$  in  $\mathbf{X} \cap Pa(D)$  do
      if  $\neg \text{CHECK}(X)$  then return false

```

Analysis of the Algorithm. The algorithm checks for every edge $X \rightarrow D$ on a proper causal path to \mathbf{Y} whether it satisfies the amenability conditions of Lemma 7 by searching a collider path through the parents of D to a node Z not connected to D ; note that condition (1) of Lemma 7 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 $O(n + m)$. Therefore, the entire algorithm runs in $O(k(n + m))$ where $k \leq |Ch(\mathbf{X})|$. \square

9.2 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 2 and 3), replacing “DAG” with “MAG” and d - with m -separation for the latter.

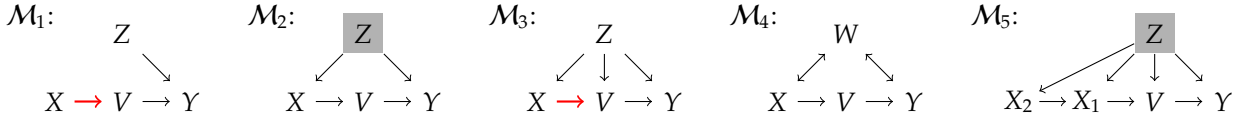


Figure 15: Five MAGs in which we search for an adjustment relative to (X, Y) or $(\{X_1, X_2\}, Y)$. \mathcal{M}_1 and \mathcal{M}_3 are not adjustment amenable, since the edge $X \rightarrow V$ is not visible, so no adjustment exists. In the other three MAGs the edge is visible, due to the node Z in \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 11 (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} .*

Note that the above definition uses “descendants in \mathcal{M} ” instead “descendants in $\mathcal{M}_{\overline{\mathbf{X}}}$ ” as Definition 2. However, Lemma 4 implies that they are the same.

Definition 12 (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 $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd}$, is obtained from \mathcal{M} by removing the first edge of every proper causal path from \mathbf{X} to \mathbf{Y} .*

Definition 13 (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) *$\mathbf{Z} \subseteq \mathbf{V} \setminus D_{pcp}(\mathbf{X}, \mathbf{Y})$ and*
- (b) *\mathbf{Z} m -separates \mathbf{X} and \mathbf{Y} in the proper back-door graph $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd}$.*

Figure 15 shows some examples. Similarly to the CBC for DAGs certain parameters of the CBC for MAGs can be changed without affecting the characterized sets:

Definition 14 (Parametrization of the Constructive back-door criterion (CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$))). *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. Let $\mathbf{A} \subseteq \mathbf{X} \cup \mathbf{Y}$, $\mathbf{B} \subseteq \mathbf{X}$, $\mathbf{C} \subseteq De(PCP(\mathbf{X}, \mathbf{Y}))$. The set \mathbf{Z} satisfies the CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$) relative to (\mathbf{X}, \mathbf{Y}) in \mathcal{M} if*

- (a) *$\mathbf{Z} \subseteq \mathbf{V} \setminus De_{\overline{\mathbf{A}\mathbf{B}}}(\overline{PCP(\mathbf{X}, \mathbf{Y})})$, and*
- (b) *\mathbf{Z} d -separates \mathbf{X} and \mathbf{Y} in the graph $\mathcal{G}_{\mathbf{X}\mathbf{Y}}^{pbd, \mathbf{C}} := (\mathbf{V}, \mathbf{E} \setminus (\mathbf{X} \rightarrow (PCP(\mathbf{X}, \mathbf{Y}) \cup \mathbf{C})))$.*

With these definitions we prove:

Theorem 3. *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 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} .*
- (iv) *\mathbf{Z} fulfills a variant of constructive back-door criterion (CBC($\mathbf{A}, \mathbf{B}, \mathbf{C}$)) w.r.t. (\mathbf{X}, \mathbf{Y}) in \mathcal{M} for $\mathbf{A} \subseteq \mathbf{X} \cup \mathbf{Y}$, $\mathbf{B} \subseteq \mathbf{X}$, $\mathbf{C} \subseteq De(PCP(\mathbf{X}, \mathbf{Y}))$.*



Figure 16: Illustration of the case in the proof of Theorem 3 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.

Proof. The equivalence of (ii), (iii) and (iv) is established by observing that the proofs of Theorem 1 and Lemma 4 generalize to m -separation. Below we establish equivalence of (i) and (ii).

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

$\neg(i) \Rightarrow \neg(ii)$: Let \mathcal{G} be a DAG with $\mathcal{G}_{\emptyset}^{\mathbf{L}} = \mathcal{M}$ in which Z violates the AC. We show that (a) if $Z \cap Dpcp(\mathbf{X}, \mathbf{Y}) \neq \emptyset$ in \mathcal{G} then $Z \cap Dpcp(\mathbf{X}, \mathbf{Y}) \neq \emptyset$ in \mathcal{M} as well, or there exists a proper non-causal path in \mathcal{M} that cannot be m -separated; and (b) if $Z \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$ in \mathcal{G} and 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} , Z contains a node Z in $Dpcp(\mathbf{X}, \mathbf{Y})$, and let $\mathbf{W} = PCP(\mathbf{X}, \mathbf{Y}) \cap An(Z)$. 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 $W_2 \in PCP_{\mathcal{G}}(\mathbf{X}, \mathbf{Y}) \setminus An(Z)$ (possibly $W_2 \in \mathbf{Y}$) such that $W_2 \leftrightarrow A$, $X \rightarrow W_2$, and $X \rightarrow A$ are edges in \mathcal{M} , where $A \in An(Z)$ (possibly $A = Z$; see Figure 16). Then \mathcal{M} contains an m -connected proper non-causal path $X \rightarrow A \leftrightarrow W \rightarrow W_2 \rightarrow \dots \rightarrow Y$.

(b) Suppose that in \mathcal{G} , $Z \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$, 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 $Y \in \mathbf{Y}$ (Lemma 10), which is d -connected by Z in \mathcal{G} . Let $w_{\mathcal{M}}$ denote the subsequence of $w_{\mathcal{G}}$ formed by nodes in \mathcal{M} , which includes all colliders on $w_{\mathcal{G}}$. The sequence $w_{\mathcal{M}}$ is a path in \mathcal{M} , but is not necessarily m -connected by Z ; all colliders on $w_{\mathcal{M}}$ are in Z because every non- Z must be a parent of at least one of its neighbors, but there can subsequences U, Z_1, \dots, Z_k, V on $w_{\mathcal{M}}$ where all $Z_i \in Z$ 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 Z -nodes (Lemma 15). Let $w'_{\mathcal{M}}$ be the resulting 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 10), violating the AC. It therefore remains to show that $w'_{\mathcal{M}}$ is not a proper causal path. 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 Z -nodes in $w_{\mathcal{M}}$ have been bypassed in $w'_{\mathcal{M}}$ by edges pointing away from X . In that case, one can show by several case distinctions that the first edge $X \rightarrow D$ of $w'_{\mathcal{M}}$, where $D \notin Z$, cannot be visible (see Figure 17 for an example of such a case).

For simplicity, assume that \mathcal{M} contains a subpath $A \rightarrow X \rightarrow D$ where A is not adjacent to D ; the other cases of edge visibility like $A \leftrightarrow X \rightarrow D$ (Lemma 7). are treated analogously. In \mathcal{G} , there are inducing paths (possibly several, see Definition 15) 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, D are adjacent (Lemma 18), 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 $Z \in De(X)$ because $w_{\mathcal{G}}$ is not causal. (a) $Z \in De(D)$. This contradicts $Z \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$. So (b) $Z \notin De(D)$. Then by construction of $w'_{\mathcal{M}}$ (Lemma 15), $w_{\mathcal{M}}$ must start with an inducing Z -trail $X \rightarrow Z, Z_1, \dots, Z_n, D$, which is also an inducing path from X to D in \mathcal{G} w.r.t \emptyset, \mathbf{L} . Then Z, Z_1, \dots, Z_n, 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 $X \rightarrow Z, D$ in \mathcal{M} . (i) If $X \rightarrow Z \rightarrow D$, then Z lies on a proper causal path, contradicting $Z \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$. (ii) 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 . \square

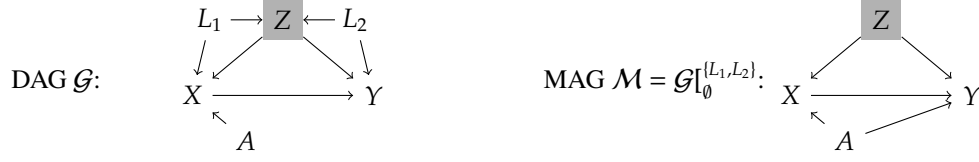


Figure 17: Case (b) in the proof of Theorem 3: A proper non-causal path $w_G = X \leftarrow L_1 \rightarrow Z \leftarrow L_2 \rightarrow Y$ in a DAG is d -connected by Z , but the corresponding proper non-causal path $w_M = X \leftarrow Z \rightarrow Y$ is not m -connected in the MAG, and its m -connected subpath $w'_M = X \rightarrow Y$ is proper causal. However, this also renders the edge $X \rightarrow Y$ invisible, because otherwise A could be m -separated from Y by $U = \{X, Z\}$ in M but not in G .

9.3 Adjustment set construction

In the previous section, we have already shown that the CBC is equivalent to the AC for MAGs as well; hence, adjustment sets for a given MAG M can be found by forming the proper back-door graph M_{XY}^{pbd} and then applying the algorithms from the previous section. In principle, care must be taken when removing edges from MAGs as the result might not be a MAG; however, this is not the case when removing only directed edges.

Lemma 9 (Closure of maximality under removal of directed edges). *Given a MAG M , every graph M' formed by removing only directed edges from M is also a MAG.*

Proof. Suppose the converse, i.e. 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 [12]. 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 M via the edge $X \rightarrow D$, a contradiction. \square

Corollary 7. *For every MAG M , the proper back-door graph M_{XY}^{pbd} is also a MAG.*

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 7. For each child D of X in $PCP(X, Y)$, we can test the visibility of all edges $X \rightarrow D$ simultaneously using depth first search. This means that we can check all potentially problematic edges in time $O(n + m)$. If all tests pass, we are licensed to apply the CBC, as shown above. Hence, we can solve all algorithmic tasks in Table 4 for MAGs in the same way as for DAGs after an $O(k(n + m))$ check of adjustment amenability, where $k \leq |Ch(X)|$.

Hence our algorithms can construct an adjustment set for a given MAG M and variables X, Y in $O((k + 1)(n + m))$ time. If an additional set Z is given, it can be verified that Z is an adjustment set in the same time.

Minimal adjustments sets can be constructed and verified in $O(k(n + m) + n^2)$ or $O(n(n + m))$ time using our algorithms FINDMINADJ, TESTMINADJ for dense or sparse graphs. The algorithms for the remaining problems of finding a minimum cost adjustment set FINDMINCOSTADJ and enumerating adjustment sets LISTADJ or LISTMINADJ in MAGs have the same runtime as the corresponding algorithms in DAGs, since their time surpasses the time required for the adjustment amenability test.

10 Discussion

We provide a framework of efficient algorithms to verify, find, and enumerate m -separating sets in MAGs, which we then harness to solve the same problems for adjustment sets in DAGs and MAGs. In both graph classes, this provides a complete and informative answer to the question when, and how, a desired causal effect between multivariate exposures and outcomes can be estimated by covariate adjustment. Adjustment sets in MAGs allow some unmeasured confounders. Even for DAG-based analyses, one can by interpreting the same graph as a MAG possibly generate an

adjustment set that is provably valid for a much larger class of DAGs. This allows researchers to partly relax the often untenable assumption that all relevant variables are known and were measured.

Our results rest on two key concepts: reduction of adjustment to m -separation in a subgraph (the *proper back-door graph*), and *adjustment amenability* for graphical models that are more causally ambiguous than DAGs. Since the publication of the preliminary version of this work [1, 2], these techniques were shown to be applicable to adjustment in four additional classes of graphical causal models: CPDAGs [19], PAGs [19], chain graphs [18], and maximal PDAGs [20]. Likewise, it has been shown that our algorithms remain applicable in the presence of selection bias [37]. As we have illustrated briefly in Section 4, we expect our algorithmic framework to be useful in other areas as well, due to the central of m -separation in the theory of graphical models. The practical feasibility of our algorithms is illustrated by the fact that they underpin both the web application “dagitty.net” as well as the associated R package [33], which currently have a substantial user community.

An interesting open question to be pursued in further research would be whether the approaches presented here could be generalized to accommodate confounding that arose by chance in a given sample, rather than for structural reasons [38].

11 Acknowledgment

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) grant LI 634/4-1.

12 References

References

- [1] J. Textor, M. 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.
- [2] B. van der Zander, M. Liśkiewicz, J. Textor, Constructing separators and adjustment sets in ancestral graphs, in: Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2014, pp. 907–916.
- [3] J. Pearl, Causality, Cambridge University Press, 2009.
- [4] F. Elwert, Graphical causal models, in: Handbook of Causal Analysis for Social Research, Handbooks of Sociology and Social Research, Springer, 2013, pp. 245–273.
- [5] K. J. Rothman, S. Greenland, T. L. Lash, Modern Epidemiology, Wolters Kluwer, 2008.
- [6] I. Shrier, Letter to the editor, Statistics in Medicine 27 (2008) 2740–2741.
- [7] D. Rubin, Author’s reply, Statistics in Medicine 27 (2008) 2741–2742.
- [8] T. J. VanderWeele, On the relative nature of overadjustment and unnecessary adjustment, Epidemiology 20 (4) (2009) 496–499. doi:10.1097/EDE.0b013e3181a82f12.
- [9] A. Glynn, K. 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, Journal of the American Statistical Association doi:10.1080/01621459.2017.1398657.
- [10] I. Shpitser, T. VanderWeele, J. 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.
- [11] S. Greenland, Hierarchical regression for epidemiologic analyses of multiple exposures, Environmental Health Perspectives 102 Suppl 8 (1994) 33–39.

- [12] T. Richardson, P. Spirtes, Ancestral graph Markov models, *Annals of Statistics* 30 (2002) 927–1223.
- [13] R. D. Shachter, Bayes-ball: The rational pastime, in: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1998, pp. 480–487.
- [14] J. Tian, A. Paz, J. Pearl, Finding minimal d-separators, Tech. Rep. R-254, University of California, Los Angeles (1998).
- [15] S. Acid, L. M. de Campos, Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs, *Journal of Artificial Intelligence Research* 18 (2003) 445–490. doi:10.1613/jair.1061.
- [16] M. H. Maathuis, D. Colombo, A generalized backdoor criterion, *Annals of Statistics* 43 (3) (2015) 1060–1088. doi:10.1214/14-AOS1295.
- [17] E. Perković, J. Textor, M. Kalisch, M. Maathuis, A complete generalized adjustment criterion, in: *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, AUAI Press, 2015, pp. 682–691.
- [18] B. van der Zander, M. Liśkiewicz, Separators and Adjustment Sets in Markov Equivalent DAGs, in: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016, pp. 3315–3321.
- [19] E. Perković, J. Textor, M. Kalisch, M. Maathuis, Complete graphical characterization and construction of adjustment sets in Markov equivalence classes of ancestral graphs, Tech. Rep. 1606.06904, arXiv:, preprint, arxiv.org/abs/1606.06903 (2016).
- [20] E. Perković, M. Kalisch, M. Maathuis, Interpreting and using CPDAGs with background knowledge, in: *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017, p. , available at arxiv.org/abs/1707.02171.
- [21] J. Pearl, Causal diagrams for empirical research, *Biometrika* 82 (4) (1995) 669–688.
- [22] I. Shpitser, J. 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.
- [23] Y. Huang, M. 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.
- [24] I. Shpitser, J. Pearl, Identification of conditional interventional distributions, in: *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, AUAI Press, 2006, pp. 437–444.
- [25] S. Acid, L. 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.
- [26] K. Takata, Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph, *Discrete Applied Mathematics* 158 (2010) 1660–1667. doi:10.1016/j.dam.2010.05.013.
- [27] S. Even, *Graph Algorithms*, Computer Science Press, 1979.
- [28] J. 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.
- [29] M. Garey, D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, W. H. Freeman and Company, 1979.
- [30] J. Pearl, P. Meshkat, Testing regression models with fewer regressors, in: *Proceedings of AI-STAT*, 1998, pp. 255–259.

- [31] F. Thoemmes, Y. Rosseel, J. Textor, Local fit evaluation of structural equation models using graphical criteria, *Psychological Methods* (2017) doi:10.1037/met0000147.
- [32] S. Tikka, J. Karvanen, Identifying Causal Effects with the R Package causaleffect, *Journal of Statistical Software* 76 (12) (2017). doi:10.18637/jss.v076.i12.
- [33] J. Textor, B. van der Zander, M. S. Gilthorpe, M. Liškiewicz, G. T. Ellison, Robust causal inference using directed acyclic graphs: the R package ‘dagitty’, *International Journal of Epidemiology* 45 (6) (2016) 1887–1894. doi:10.1093/ije/dyw341.
- [34] M. Kalisch, M. Mächler, D. Colombo, M. Maathuis, P. Bühlmann, Causal inference using graphical models with the R package pcalg, *Journal of Statistical Software* 47 (11) (2012). doi:10.18637/jss.v047.i11.
- [35] J. Zhang, Causal reasoning with ancestral graphs, *Journal of Machine Learning Research* 9 (2008) 1437–1474.
- [36] E. Bareinboim, J. Tian, J. Pearl, Recovering from Selection Bias in Causal and Statistical Inference, in: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014, pp. 2410–2416.
- [37] J. D. Correa, E. Bareinboim, Causal effect identification by adjustment under confounding and selection biases, in: *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 3740–3746.
- [38] S. Greenland, M. A. Mansournia, Limitations of individual causal models, causal graphs, and ignorability assumptions, as illustrated by random confounding and design unfaithfulness, *Eur. J. Epidemiol.* 30 (10) (2015) 1101–1110.

13 Appendix

13.1 Auxiliary lemmas and proofs

In this section, we prove several auxiliary lemmas that are necessary for the proof of Theorem 3.

Lemma 10. *Given a DAG \mathcal{G} and sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ satisfying $\mathbf{Z} \cap Dpcp(\mathbf{X}, \mathbf{Y}) = \emptyset$, \mathbf{Z} m -connects a proper non-causal path between \mathbf{X} and \mathbf{Y} if and only if it m -connects a proper non-causal walk between \mathbf{X} and \mathbf{Y} .*

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 by inserting $C_i \rightarrow \dots \rightarrow Z_i \leftarrow \dots \leftarrow C_i$ 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 $\mathbf{X} \leftarrow \dots \leftarrow C_i$. \square

In all of the below, $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a DAG, $\mathbf{Z}, \mathbf{L} \subseteq \mathbf{V}$ are disjoint, and $\mathcal{M} = \mathcal{G}_{\mathbf{L}_0}^{\mathbf{L}}$.

Definition 15 (Inducing path [12]). *A path $\pi = V_1, \dots, V_{n+1}$ is called inducing with respect to \mathbf{Z}, \mathbf{L} if all non-colliders on π except V_1 and V_{n+1} are in \mathbf{L} , and all colliders on π are in $An(\{V_1, V_{n+1}\} \cup \mathbf{Z})$.*

Every inducing path w.r.t. \mathbf{Z}, \mathbf{L} is m -connected by \mathbf{Z} .

Lemma 11 ([12]). *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 $\mathcal{G}_{\mathbf{L}_0}^{\mathbf{L}}$.*

Proof. This is Theorem 4.2, cases (v) and (vi), in [12]. \square

Lemma 12. *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 [12]. For the second part on arrowheads, suppose π does not have an arrowhead at U , then π starts with an edge $U \rightarrow D$. Hence $D \notin \text{An}(U)$, so $D \in \text{An}(V)$ because π is an inducing path and therefore also $U \in \text{An}(V)$. Hence, the edge between U and V in $\mathcal{G}[\emptyset, \mathbf{L}]$ must be $U \rightarrow V$. The argument for V is identical. \square

Lemma 13. *Suppose Z_0, Z_1, 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 $\text{An}(Z_0 \cup Z_2)$. Because Z_1 is not a collider on Z_0, Z_1, Z_2 , at least one of the edges Z_0, Z_1 and Z_1, 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 $\text{An}(Z_0 \cup Z_1) = \text{An}(Z_0) \subseteq \text{An}(Z_0 \cup Z_2)$, and all colliders on π_{12} are in $\text{An}(Z_1 \cup Z_2) \subseteq \text{An}(Z_0 \cup Z_2)$. Z_1 itself is a collider on w_{012} and is also in $\text{An}(Z_0)$. 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} . \square

Definition 16 (Inducing \mathbf{Z} -trail). *Let $\pi = V_1, \dots, V_{n+1}$ be a path in $\mathcal{G}[\emptyset, \mathbf{L}]$ such that $V_2, \dots, V_n \in \mathbf{Z}$, $V_1, V_{n+1} \notin \mathbf{Z}$, and for each $i \in \{1, \dots, n\}$, there is an inducing path w.r.t. \emptyset, \mathbf{L} linking V_i, V_{i+1} that has an arrowhead at V_i (V_{i+1}) if $V_i \in \mathbf{Z}$ ($V_{i+1} \in \mathbf{Z}$). Then π is called an inducing \mathbf{Z} -trail.*

Lemma 14. *Let $\pi = V_1, \dots, V_{n+1}$ be an inducing \mathbf{Z} -trail, and let π' be a subsequence of π formed by removing one node V_i of π such that $V_i \in \mathbf{Z}$ is a non-collider on π . Then π' is an inducing \mathbf{Z} -trail.*

Proof. According to Lemma 13, 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. \square

Corollary 8. *Every inducing \mathbf{Z} -trail $\pi = V_1, \dots, V_{n+1}$ 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 Lemma 14, 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} . \square

Lemma 15. *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 $w_{\mathcal{M}} = V_1, \dots, V_{n+1}$ 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, \dots, V_{n+1} , $n > 1$, of $w_{\mathcal{M}}$ where $V_2, \dots, V_n \in \mathbf{Z}$, $V_1, V_{n+1} \notin \mathbf{Z}$, which now are all paths in $w_{\mathcal{M}}$. On those subsequences, every V_i must be adjacent in \mathcal{G} to V_{i+1} via a path containing no colliders, and all non-endpoints on 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, \dots, V_{n+1} is an inducing \mathbf{Z} -trail, and has a subpath which m -connects V_1, V_{n+1} given \mathbf{Z} . Transform $w_{\mathcal{M}}$ to $w'_{\mathcal{M}}$ by replacing all inducing \mathbf{Z} -trails by their m -connected subpaths. According to Lemma 12, 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} . \square

Corollary 9. *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 $w_{\mathcal{G}}$.*

Lemma 16. 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 π_{01} and π_{12} 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} . \square

Lemma 17. Suppose Z_0, Z_1, \dots, Z_{k+1} is a path in $\mathcal{G}_{\emptyset}^{\mathbf{L}}$ with an arrowhead at Z_{k+1} on which all Z_1, \dots, Z_k are colliders. Then there exists an inducing path from Z_0 to Z_{k+1} w.r.t. $\{Z_1, \dots, Z_k\}, \mathbf{L}$ with an arrowhead at Z_{k+1} .

Proof. Because all Z_i, Z_{i+1} are adjacent and all Z_1, \dots, 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, \dots, Z_k (Lemma 12). The claim follows by repeatedly applying Lemma 16 to the $\pi_{i,i+1}$'s. \square

Lemma 18. Suppose $A \rightarrow V_1 \leftrightarrow \dots \leftrightarrow V_k \leftrightarrow X \rightarrow D$ or $A \leftrightarrow V_1 \leftrightarrow \dots \leftrightarrow V_k \leftrightarrow X \rightarrow D$ is a path in $\mathcal{G}_{\emptyset}^{\mathbf{L}}$ (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 $\mathcal{G}_{\emptyset}^{\mathbf{L}}$.

Proof. Assume the path is $A \rightarrow V_1 \leftrightarrow \dots \leftrightarrow V_k \leftrightarrow X \rightarrow D$. 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 $\mathcal{G}_{\emptyset}^{\mathbf{L}}$, and let \mathbf{Z} be such a separator. If V_1 is not in \mathbf{Z} then the path $A \rightarrow V_1 \rightarrow D$ 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 $A \rightarrow V_1 \leftrightarrow \dots \leftrightarrow V_{i-1} \leftrightarrow V_i \rightarrow D$ is not blocked, so $V_i \in \mathbf{Z}$ for all i .

There exist an inducing path π_{AX} from A to X with an arrowhead at X w.r.t. to $\{V_1, \dots, V_k\}, \mathbf{L}$ (Lemma 17) which can be combined with π_{XD} to an inducing path from A to D w.r.t. to $\{V_1, \dots, V_k, X\}, \mathbf{L}$ (Lemma 16).

Hence no m -separator of A, D can contain $\{X, V_1, \dots, V_k\}$ (Lemma 11). Then there cannot exist an m -separator, because every separator must include V_1, \dots, V_k and the path $A \rightarrow V_1 \leftrightarrow V_2 \leftrightarrow \dots \leftrightarrow V_k \leftrightarrow X \rightarrow D$ is open without $X \in \mathbf{Z}$. \square

13.2 Further experimental results

Tables 8 and 9 in this section show the results of versions of the experiments presented in Section 8 in Tables 5 and 6 in which the parameter controlling the number of unobserved variables is set to 0.25 or 0.5.

13.3 Do-calculus

Some of our proofs use the rules of the do-calculus [3] that exchange actions and observations or remove them altogether. We state those rules here for sake of self-containedness.

Theorem 4 (Do-Calculus [3]). Given a DAG \mathcal{G} and disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W}$ the following rules are valid for all probability distributions P consistent 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)

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

Rule 3. (Insertion/deletion of actions)

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

n	k	$l = 2$			$l = 5$			$l = 10$			$l = 20$		
		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 8: Numbers of instances for $P(\text{unobserved}) = 0.25$ that are identifiable We did not run the IDC algorithm on these data due to its high time complexity. Gray cells highlight where the BC failed to identify more than 4% of those cases that CBC could identify.

n	k	$l = 2$			$l = 5$			$l = 10$			$l = 20$		
		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 9: Numbers of instances for $P(\text{unobserved}) = 0.5$ that are identifiable by use of BC, CBC, CBC⁺ (as defined in Table 5). We did not run the IDC algorithm on these data due to its high time complexity. Gray cells highlight where the BC failed to identify more than 4% of those cases that CBC could identify.