Separators and Adjustment Sets in Markov Equivalent DAGs

Benito van der Zander* and **Maciej Liśkiewicz**
Institute of Theoretical Computer Science, University of Lübeck
Ratzeburger Allee 160, 23538 Lübeck, Germany
{benito,liskiewi}@tcs.uni-luebeck.de

Abstract

In practice the vast majority of causal effect estimations from observational data are computed using adjustment sets which avoid confounding by adjusting for appropriate covariates. Recently several graphical criteria for selecting adjustment sets have been proposed. They handle causal directed acyclic graphs (DAGs) as well as more general types of graphs that represent Markov equivalence classes of DAGs, including completed partially directed acyclic graphs (CPDAGs). Though expressed in graphical language, it is not obvious how the criteria can be used to obtain effective algorithms for finding adjustment sets. In this paper we provide a new criterion which leads to an efficient algorithmic framework to find, test and enumerate covariate adjustments for chain graphs – mixed graphs representing in a compact way a broad range of Markov equivalence classes of DAGs.

1 Introduction

Covariate adjustment is one of the most widely used techniques to estimate causal effects from observational data. The causal effect is the probability distribution of some outcomes in a post-treatment period resulting from the treatment (Pearl 2009). The primary difficulty in application of the adjustment approach is the selection of covariates one needs to adjust to compute the post-treatment distribution.

The concept of covariate adjustments is well-understood wben the structure encoding the causal relationships between variables of interest is fully known and represented as a directed acyclic graph (DAG). Pearl's back-door criterion (Pearl 1995) is probably the most well-known method of selecting possible sets for adjustment in DAGs. It is sufficient but not necessary. Due to Shpitser, VanderWeele, and Robins (2010) we know a criterion expressed in graphical language that is necessary and sufficient in the sense that it is satisfied if and only if the adjustment conditions are fulfilled. This reduces the properties of probability distributions to properties of causal graphs. Based on this criterion Textor and Liśkiewicz (2011) and van der Zander, Liśkiewicz, and Textor (2014) have proposed an algorithmic framework for efficient testing and finding covariate adjustments in DAGs.

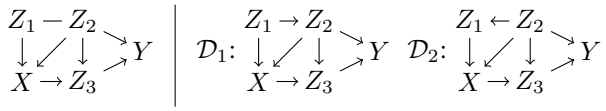

Figure 1: A chain graph (to the left) which represents two Markov equivalent DAGs: $\mathcal{D}_1$ and $\mathcal{D}_2$. Relative to exposure X and outcome Y, $\mathbf{Z} = \{Z_2\}$ is an adjustment set in both $\mathcal{D}_1$ and $\mathcal{D}_2$. Thus $\mathbf{Z}$ is an adjustment set in the chain graph.

However, in practice most statistical data and background knowledge can be explained by several causal DAGs equally well, so the true underlying DAG remains unknown. For example, the structure learning algorithm proposed by Verma and Pearl (1990; 1992) constructs, for a given list of conditional independence statements $\mathcal{M}$, a CPDAG (Andersson et al. 1997) representing all DAGs which are complete causal explanations of $\mathcal{M}$. Meek (1995) extends this algorithm providing a method to compute a complete causal explanation for $\mathcal{M}$ which is consistent with background knowledge represented as a set of required and forbidden directed edges. This results in a mixed graph, which might not be a CPDAG anymore.

In our study we assume that the learned causal structure is represented as a chain graph – a mixed graph containing no semi-directed cycles (Lauritzen and Wermuth 1989). A primary benefit of chain graphs is that they provide an elegant framework for modeling and analyzing a broad range of Markov equivalence classes of DAGs (Verma and Pearl 1990; Andersson et al. 1997).

Given a chain graph and the pre-intervention distribution we can compute causal effects using the covariate adjustment approach. However, the challenging task is now to find an adjustment set which is common for *every* DAG represented by the chain graph. Figure 1 shows an example for such an adjustment. The naive approach of searching for adjustment sets in all DAGs leads to exponential time algorithms since the number of DAGs represented by a chain graph can grow exponentially in the size of the graph.

Recently Perković et al. (2015) have presented a graphical criterion that is necessary and sufficient for CPDAGs.

*This work was supported by the Graduate School for Computing in Medicine and Life Sciences funded by Germany's Excellence Initiative [DFG GSC 235/2]

But the challenge remains to bridge the gap between their criterion and algorithmic efficiency. In this paper we solve a more general problem providing fast algorithms for adjustment sets in chain graphs. Thus, if a structure learning algorithm gives a mixed graph then our algorithms are applicable in all cases when the resulting graph does not have a semi-directed cycle.

Our algorithms reduce the problems of testing, finding, and enumeration of adjustment sets to the d-connectivity problem in a subclass of chain graphs, which we call *restricted chain graphs* (RCGs). This class includes both DAGs and CPDAGs, and seems to remain a powerful model for analyzing causal relationships. We provide a new adjustment criterion for restricted chain graphs which leads to an efficient algorithmic framework for solving problems involving covariate adjustments.

The paper is organized as follows. The next two sections present definitions and backgrounds of covariate adjustments. In Section 4 we provide our algorithm for finding adjustment sets in chain graphs. Sections 5 to 8 analyze the correctness and complexity of the algorithm.

2 Definitions

We consider mixed graphs $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ with nodes (vertices, variables) $\mathbf{V}$ and directed ($A \to B$) and undirected ($A - B$) edges $\mathbf{E}$. By n we denote $n = |\mathbf{V}|$, $m = |\mathbf{E}|$. If a graph contains only directed edges we call it a directed graph and denote it as $\mathcal{D}$. A DAG is a directed graph with no directed cycles. Nodes linked by an edge are *adjacent*. If there is an edge $A \to B$, A is a *parent* of B and B a *child* of A. A *path* is a sequence $V_0, \ldots, V_k$ of pairwise distinct nodes such that for all i, with $0 \leq i < k$, there exists an edge connecting V_i and V_{i+1}. A node V_i on $V_0, \ldots, V_k$ is a *collider* if it occurs on the path as $V_{i-1} \to V_i \leftarrow V_{i+1}$, and a *non-collider* otherwise. A path $\pi = V_0, \ldots, V_k$ is called *possible directed* (*possible causal*) from V_0 to V_k if for every $0 \leq i < k$, the edge between V_i and V_{i+1} is not into V_i. If such a π contains only directed edges it is called *directed* or *causal*. A node X is a *possible ancestor* of Y, and Y is a *possible descendant* of X, if $X = Y$ or there exists a possible directed path π from X to Y. If π is a directed path, then X is an ancestor of Y and Y a descendant of X. Given node sets $\mathbf{X}$ and $\mathbf{Y}$, a path from $X \in \mathbf{X}$ to $Y \in \mathbf{Y}$ is called *proper* if it does not intersect $\mathbf{X}$ except at the endpoint. We refer to the set of all ancestors or possible ancestors of $\mathbf{X}$ as $An(\mathbf{X})$, resp. $possibleAn(\mathbf{X})$. Similarly, we use $De(\mathbf{Y})$ and $possibleDe(\mathbf{Y})$ to denote the descendants, resp. possible descendants of $\mathbf{Y}$. For any subset of nodes $\mathbf{W} \subseteq \mathbf{V}$ of a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ the *induced subgraph* of $\mathbf{W}$, written as $\mathcal{G}_{\mathbf{W}}$, is the graph on nodes $\mathbf{W}$ that contains an edge $e \in \mathbf{E}$ if and only if both end points of e are in $\mathbf{W}$. The skeleton of any mixed graph $\mathcal{G}$ is the undirected graph resulting from ignoring the directionality of all edges. A v-structure in a mixed graph $\mathcal{G}$ is an ordered triple of nodes (A, B, C) such they induce the subgraph $A \to B \leftarrow C$.

To extend the notion of d-connectivity to mixed graphs we use the definitions proposed by Zhang (2008). A node V on a path π in a mixed graph $\mathcal{G}$ is called a *definite non-collider*, if there is an induced subgraph $A \leftarrow V$ or $V \to B$ or $A - V - B$, where A and B are the nodes preceding/succeeding V on π. A non-endpoint vertex on π is said to be of *definite status*, if it is either a collider or a definite non-collider on π. A path is said to be of definite status if all its non-endpoint vertices are of definite status. Given a mixed graph, a path π between nodes X and Y, and a set $\mathbf{Z}$ (possibly empty and $X, Y \notin \mathbf{Z}$) we say that π is *d-connecting* relative to $\mathbf{Z}$ if every non-collider on π is not in $\mathbf{Z}$, and every collider on π has a descendant in $\mathbf{Z}$. Given pairwise different sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, set $\mathbf{Z}$ d-separates $\mathbf{X}$ and $\mathbf{Y}$ if there exists no d-connected definite status path between any $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$.

A possible directed path $V_0, \ldots, V_k$ in a mixed graph $\mathcal{G}$ is a semi-directed cycle if there is an edge between V_k and V_0 in $\mathcal{G}$ and at least one of the edges is directed as $V_i \to V_{i+1}$ for $0 \leq i \leq k$. Here, $V_{k+1} = V_0$. A *chain graph* (CG) is a graph without semi-directed cycles (Lauritzen and Wermuth 1989). A Bayesian network for a set of variables $\mathbf{V} = \{X_1, \ldots, X_n\}$ consists of a pair $(\mathcal{D}, P)$, where $\mathcal{D}$ is a DAG with $\mathbf{V}$ as the set of nodes, and P is the joint probability function over the variables in $\mathbf{V}$ that factorizes according to $\mathcal{D}$ as follows $P(\mathbf{v}) = \prod_{j=1}^{n} P(x_j | pa_j)$, where $\mathbf{v}$ denotes a particular realization of variables $\mathbf{V}$ and pa_j denotes a particular realization of the parent variables of X_j in $\mathcal{D}$. When interpreted causally, an edge $X_i \to X_j$ is taken to represent a direct causal effect of X_i on X_j (Pearl 2009). Two DAGs are *Markov equivalent* if they imply the same set of conditional independencies. Due to Verma and Pearl (1990) we know that two DAGs are Markov equivalent if they have the same skeletons and the same v-structures.

Given a DAG $\mathcal{D} = (\mathbf{V}, \mathbf{E})$, the class of Markov equivalent graphs to $\mathcal{D}$, denoted as $[\mathcal{D}]$, is defined as $[\mathcal{D}] = \{\mathcal{D}' \mid \mathcal{D}' \text{ is Markov equivalent to } \mathcal{D}\}$. The graph representing $[\mathcal{D}]$, called a completed partially directed acyclic graph (CPDAG) or an essential graph, is a mixed graph denoted as $\mathcal{D}^* = (\mathbf{V}, \mathbf{E}^*)$, with the set of edges defined as follows: $A \to B$ is in $\mathbf{E}^*$ if $A \to B$ belongs to every $\mathcal{D}' \in [\mathcal{D}]$ and $A - B$ is in $\mathbf{E}^*$ if there exist $D', D'' \in [\mathcal{D}]$ such that $A \to B$ is an edge of $\mathcal{D}'$ and $A \leftarrow B$ an edge of $\mathcal{D}''$ (Andersson et al. 1997). A mixed graph $\mathcal{G}$ is called a CPDAG if $\mathcal{G} = \mathcal{D}^*$ for some DAG $\mathcal{D}$. Note that in general, it is not true that a DAG is a CPDAG. A simple counterexample is a DAG: $A \to B$.

Given a chain graph $\mathcal{G}$ a DAG $\mathcal{D}$ is a consistent DAG extension of $\mathcal{G}$ if and only if (1) $\mathcal{G}$ and $\mathcal{D}$ have the same skeletons, (2) if $A \to B$ is in $\mathcal{G}$ then $A \to B$ is in $\mathcal{D}$, and (3) $\mathcal{G}$ and $\mathcal{D}$ have the same v-structures. We refer to all consistent DAG extensions of a mixed graph $\mathcal{G}$ as $CE(\mathcal{G})$. Notice that if $\mathcal{G}$ is a CPDAG for some DAG $\mathcal{D}$ then $CE(\mathcal{G}) = [\mathcal{D}]$.

3 Covariate Adjustment in DAGs and CGs

We start this section with the formal definition of adjustment. Next we present known results for adjustments in DAGs and CPDAGs.

Let $\mathcal{D} = (\mathbf{V}, \mathbf{E})$ be a DAG encoding the factorization of a joint distribution for variables $\mathbf{V} = \{X_1, \ldots, X_n\}$. 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 $\mathbf{X} = \mathbf{x}$. This definition models an idealized experiment in which the variables in $\mathbf{X}$ can be set to

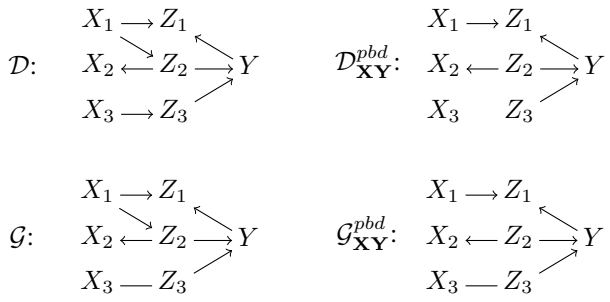

Figure 2: Proper back-door graphs for a DAG $\mathcal{D}$ and a chain graph $\mathcal{G}$ both with $\mathbf{X} = \{X_1, X_2, X_3\}, \mathbf{Y} = \{Y\}$.

given values. If $\mathbf{v}$ is consistent with $\mathbf{x}$, the post-intervention distribution can be expressed in a truncated factorization formula: $P(\mathbf{v}|do(\mathbf{x})) = \prod_{X_j \in \mathbf{V}\setminus\mathbf{X}} P(x_j|pa_j)$. Otherwise $P(\mathbf{v}|do(\mathbf{x})) = 0$. For pairwise disjoint sets of nodes $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$ in a DAG $\mathcal{D}$, set $\mathbf{Z}$ is called *adjustment* relative to ($\mathbf{X}$, $\mathbf{Y}$) if for every distribution P consistent with $\mathcal{D}$ we have $P(\mathbf{y}|do(\mathbf{x})) = \sum_{\mathbf{z}} P(\mathbf{y}|\mathbf{x}, \mathbf{z})P(\mathbf{z})$ (Pearl 2009).

For a chain graph $\mathcal{G}$, a set $\mathbf{Z}$ is an adjustment relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$, if $\mathbf{Z}$ is an adjustment relative to $(\mathbf{X}, \mathbf{Y})$ in any consistent DAG extension of $\mathcal{G}$.

Relying on the definition it is difficult to decide, if a given set is an adjustment in a DAG or not. Fortunately due to Shpitser, VanderWeele, and Robins (2010) we know a necessary and sufficient criterion for this property.

Definition 1 (**Adjustment Criterion (AC) for DAGs;** (Shpitser, VanderWeele, and Robins 2010; Shpitser 2012))**.** *Let $\mathcal{D} = (\mathbf{V}, \mathbf{E})$ be a DAG and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be pairwise disjoint subsets of $\mathbf{V}$. The set $\mathbf{Z}$ satisfies the* adjustment criterion *relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{D}$ if*

(a) *no element in $\mathbf{Z}$ is a descendant in $\mathcal{D}$ 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 $\mathcal{D}$ from $\mathbf{X}$ to $\mathbf{Y}$ are blocked by $\mathbf{Z}$.*

Most recently Perković et al. (2015) have generalized the criterion to CPDAGs and they have proven necessity and sufficiency of this generalized criterion for CPDAGs.

In (van der Zander, Liśkiewicz, and Textor 2014) a new criterion is proposed for DAGs which is equivalent to the AC (Definition 1). The crucial role here plays the *proper back-door graph*, in which the first edge of every proper causal path from $\mathbf{X}$ to $\mathbf{Y}$ is removed (see Fig. 2 for an example). Based on this notion the so called *constructive back-door criterion* for DAGs, is obtained from AC by replacing condition (b) by the following one: $\mathbf{Z}$ d-separates $\mathbf{X}$ and $\mathbf{Y}$ in the proper back-door graph. In this way the criterion reduces adjustment problems to d-separation problems.

4 Main Results

In this section we propose a method to find adjustment sets for a given chain graph. To describe our algorithm we introduce first several auxiliary definitions and notations. We refer to the nodes which lie on a proper possible causal path from $\mathbf{X}$ to $\mathbf{Y}$ as $PCP(\mathbf{X}, \mathbf{Y})$. So we let $PCP(\mathbf{X}, \mathbf{Y}) = \{W \in \mathbf{V}\setminus\mathbf{X} \mid W$ lies on a proper possible causal path from $\mathbf{X}$ to $\mathbf{Y}\}$ and generalize the proper back-door graphs for CGs as follows (for an example see Fig. 2):

Definition 2 (**Proper Back-Door Graph for CGs).** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a chain graph, and $\mathbf{X}, \mathbf{Y}$ be disjoint subsets of nodes of $\mathcal{G}$. The* proper back-door graph*, denoted as $\mathcal{G}^{pbd}_{\mathbf{XY}}$, is obtained from $\mathcal{G}$ by removing all edges $X \to D$ in $\mathbf{E}$ such that $X \in \mathbf{X}$ and $D \in PCP(\mathbf{X}, \mathbf{Y})$.*

In order to create efficient algorithms for CGs, we define a new, simpler kind of paths and provide a generalized back-door criterion based on those paths.

Definition 3 (**Almost Definite Status).** *Let π be a path in a mixed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$. A node V on π is called an* almost definite non-collider*, if it occurs as $A \leftarrow V$, $V \rightarrow B$ or as $A - V - B$ on π, where A and B are the nodes preceding/succeeding V on π. A non-endpoint vertex V on π is said to be of almost definite status, if it is either a collider or an almost definite non-collider on π. A path π is said to be of* almost definite status *if all non-endpoint vertices on the path are of almost definite status.*

The property of almost definite status only depends on the edges in the paths, not on those outside the path, and is so algorithmically easier to handle than definite status. For example in the CG $\mathcal{G}$: $A \overset{\frown}{-} B \overset{\frown}{-} C \to D \leftarrow E$ the path $A - B - C \to D \leftarrow E$ is of almost definite status. It is not of definite status, because A and C are connected and thus there exists a consistent DAG extension of $\mathcal{G}$ that contains a collider $A \to B \leftarrow C$. We state the following criterion by using these paths.

Definition 4 (**Constructive Back-Door Criterion for CGs).** *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be a chain graph, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. $\mathbf{Z}$ satisfies the* constructive back-door criterion *relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$ if*

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

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

Now we are ready to describe an algorithm to find in a given chain graph $\mathcal{G}$ an adjustment set relative to a given pair $(\mathbf{X}, \mathbf{Y})$. The rule used in Step 1 is applied to variables A, B, C if the induced graph of $\{A, B, C\}$ is $A \to B - C$. Moreover, recall, that a *chain component* of $\mathcal{G}$ (used in Step 2) is a connected component of the undirected graph obtained from $\mathcal{G}$ by removing all directed edges, and *chordal* means that every cycle of length ≥ 4 possesses a chord i.e. two nonconsecutive adjacent vertices.

Function FINDADJSET($\mathcal{G}, \mathbf{X}, \mathbf{Y}$)

1. Close $\mathcal{G}$ under the rule $A \to B - C \Rightarrow A \to B \to C$. If a new v-structure occurs then return $\perp$ and exit.
2. If some chain component of the resulting graph is not chordal then return $\perp$ and exit.
3. Let $\mathcal{R}$ denote the resulting graph.
4. Return a set $\mathbf{Z}$ satisfying the constructive back-door criterion for $\mathcal{R}$.

Using this algorithm we get our main results.

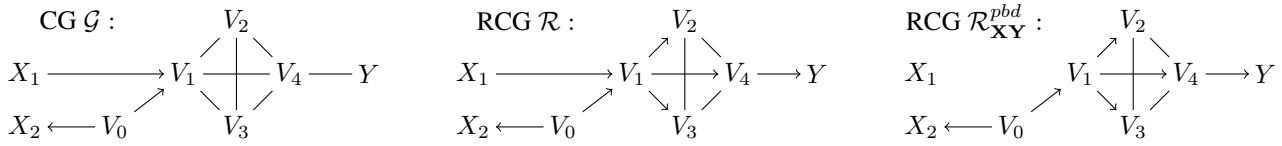

Figure 3: An execution of FINDADJSET on an input chain graph $\mathcal{G}$ with $\mathbf{X} = \{X_1, X_2\}$, $\mathbf{Y} = \{Y\}$. The RCG $\mathcal{R}$ is constructed from $\mathcal{G}$ in Step 1 and it satisfies $CE(\mathcal{G}) = CE(\mathcal{R})$. The only chain component $\{V_2, V_3, V_4\}$ is chordal. Step 4 constructs the proper back-door graph $\mathcal{R}^{pbd}_{\mathbf{XY}}$ and *possibleDe*$(PCP(\mathbf{X}, \mathbf{Y})) = \{V_1, V_2, V_3, V_4, Y\}$. The (only) adjustment set is $\mathbf{Z} = \{V_0\}$.

Theorem 5. *Given a chain graph $\mathcal{G}$ and sets of disjoint nodes $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$ the problem of finding an adjustment set $\mathbf{Z}$ relative to $(\mathbf{X}, \mathbf{Y})$ can be solved in time $\mathcal{O}(n^4)$.*

Figure 3 illustrates the execution of the algorithm on an example chain graph. To prove the theorem we first introduce a subclass of chain graphs. Then the proof follows from the propositions below.

Definition 6 (Restricted Chain Graph). *A chain graph $\mathcal{G}$ is a* restricted chain graph *(RCG) if and only if (1) every chain component of $\mathcal{G}$ is chordal, and (2) the configuration $A \to B - C$ does not exist as induced subgraph of $\mathcal{G}$.*

Proposition 7. *If $CE(\mathcal{G}) \neq \emptyset$ then algorithm* FINDADJSET *generates in Step 3 an RCG $\mathcal{R}$ with $CE(\mathcal{G}) = CE(\mathcal{R})$. Otherwise the algorithm returns $\perp$. Moreover the Steps 1-2 of* FINDADJSET *can be implemented by an algorithm running in time $\mathcal{O}(k^2 m) \leq \mathcal{O}(n^4)$, where k describes the maximum degree of nodes in $\mathcal{G}$.*

Proposition 8. *If $CE(\mathcal{G}) \neq \emptyset$ then algorithm* FINDADJSET *computes in Step 4 an adjustment set relative to $(\mathbf{X}, \mathbf{Y})$ if and only if such a set exists. Moreover the resulting adjustment set can be computed in time $\mathcal{O}(n + m)$.*

We prove Proposition 7 in Section 5. Next, in Section 6 and 7 we discuss properties of RCGs and provide a criterion for covariate adjustments in RCGs which we apply in Section 8 to prove Proposition 8.

Algorithm FINDADJSET requires $\mathcal{O}(n^4)$ time in the general case. But, if the input graph is already an RCG, only Step 4 needs to be performed, and the problem can be solved in linear time. It is easy to see that any DAG is an RCG. Moreover, every chordal undirected graph is an RCG, too. From the characterization of CPDAGs given by Andersson et al. (1997) it follows that every CPDAG is also an RCG.

Using our method we can solve further problems involving covariate adjustment in chain graphs: testing, enumerating all adjustment sets, and finding a minimal or minimum adjustment set. To this end we modify Step 4. Due to the constructive back-door criterion the problems can be solved by finding and enumerating separating sets in an RCG. Algorithms for these generalizations are described in Section 8.

5 Reducing a CG to an RCG

The proof that algorithm FINDADJSET, for a given CG $\mathcal{G}$, computes in Step 3 an appropriate RCG $\mathcal{R}$ requires three lemmas.

Lemma 9. *Let $\mathcal{G}$ be a chain graph and let $\mathcal{G}_r$ be obtained from $\mathcal{G}$ after a single application of the rule $A \to B - C \Rightarrow A \to B \to C$. If $\mathcal{G}$ and $\mathcal{G}_r$ have the same v-structures, then $CE(\mathcal{G}) = CE(\mathcal{G}_r)$; Otherwise, if $\mathcal{G}$ and $\mathcal{G}_r$ do not have the same v-structures, then $CE(\mathcal{G}) = \emptyset$.*

Lemma 10. *Let $\mathcal{G}$ be a chain graph and let $\mathcal{G}_r^*$ be the closure of $\mathcal{G}$ under the rule $A \to B - C \Rightarrow A \to B \to C$. Then $\mathcal{G}_r^*$ is a chain graph.*

Lemma 11. *Every chain component of a chain graph $\mathcal{G}$ with $CE(\mathcal{G}) \neq \emptyset$ is chordal.*

It follows from the above lemmas that the algorithm does not abort with $\perp$, if $CE(\mathcal{G}) \neq \emptyset$, and that $CE(\mathcal{R}) = CE(\mathcal{G})$. It is also guaranteed that $\mathcal{R}$ is an RCG.

The stated runtime follows from the straightforward implementation of the algorithm. Chordality can be tested in linear time by lexicographic breadth-first search (Rose, Tarjan, and Lueker 1976).

This completes the proof of Proposition 7.

6 Properties of RCGs

We first show that a possible directed path in an RCG can be converted to a directed path, if it starts with a directed edge, which is the key difference between general chain graphs and RCGs.

Lemma 12. *If in an RCG $\mathcal{G}$ a possible directed path $\pi = V_1, \ldots, V_k$ from V_1 to V_k contains a node V_i with a subpath $V_{i-1} \to V_i - V_{i+1}$ then in $\mathcal{G}$ there exists a possible directed path $\pi' = V_1, \ldots, V_{i-1} \to V_{i+1}, \ldots, V_k$.*

Proof. Let π be the shortest possible directed path from V to W for which no such π' exists. Then π contains a subpath $A \to B - C$ which must not occur as induced subgraph in an RCG, so A and C are connected. If they are connected like in $A \overleftarrow{\to} B - C$ or $A \overrightarrow{\to} B - C$ there exists a semi-directed cycle. So they are connected as $A \overrightarrow{\to} B \overleftarrow{-} C$ □

The lemma expresses that on every possible directed path, if it contains as a subpath $V_{i-1} \to V_i - V_{i+1}$ then in the graph there must exist a directed edge $V_{i-1} \to V_{i+1}$. Thus edge $V_i - V_{i+1}$ can be removed from the path, which iteratively results in a path in which no such subpaths exist.

From this simple lemma we can conclude properties which are very useful to analyze RCGs. Particularly, that a possible directed path between V and W implies the existence of a path between V and W with at most one undirected subpath followed by a directed subpath. Moreover we can get various invariances when transforming the initial

path to the final one, like the fact that the possible descendants do not change.

Let us now consider the relationship between definite status and almost definite status paths in RCGs:

Lemma 13. *Let $\mathcal{G}$ be an RCG, X and Y nodes, and let $\mathbf{Z}$ be a subset of nodes of $\mathcal{G}$ with $X, Y \notin \mathbf{Z}$. Then there exists a d-connected definite status path between X and Y given $\mathbf{Z}$ if and only if there exists a d-connected almost definite status path between X and Y given $\mathbf{Z}$. Moreover both paths have the same directed edges.*

Proof. Let π be the shortest d-connected almost definite status path between X and Y. If it is not a definite status path it contains nodes $U-V-W$, such that U and W are connected by an edge. This edge cannot be directed or there would be a semi-directed cycle. Thus we can replace $U - V - W$ by $U - W$. In both cases we get a shorter path. Since U and W still have the same kind of adjacent edges, it is a shorter d-connected almost definite status path. The proof of the opposite implication is trivial since every definite status path is an almost definite status path. □

Zhang (2008) proves the following lemma for PAGs, and it is not hard to see that it also holds for RCGs:

Lemma 14. *A d-connected given $\mathbf{Z}$ definite status path between X and Y exists in an RCG $\mathcal{G}$ if and only if there exists a d-connected path between X and Y given $\mathbf{Z}$ in one (every) DAG $\mathcal{D} \in CE(\mathcal{G})$.*

In RCGs this lemma also holds for almost definite status paths due to Lemma 13. Since every definite non-collider is not a collider in any consistent DAG extension of $\mathcal{G}$, the path in a DAG corresponding to a definite status path in an RCG $\mathcal{G}$ has exactly the same nodes. This is, however, not true for the other direction or for almost definite status paths.

D-separation is not monotonic, i.e. adding a node to a separating set, can unblock a path and result in a non-separating set. Thus it is helpful, e.g. for finding minimal sets, to convert a d-separation problem to a vertex cut separation problem in an undirected graph. In DAGs such a conversion can be done by moralization, which generalizes to RCGs in a straightforward way:

Definition 15. *The moral graph $\mathcal{G}^m$ of an RCG $\mathcal{G}$ is an undirected graph with the same node set that results from connecting all unconnected parents of a common child with an undirected edge, and replacing every directed edge with an undirected edge.*

Lemma 16. *Given an RCG $\mathcal{G}$ and three disjoint sets $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$, set $\mathbf{Z}$ d-separates $\mathbf{X}$ and $\mathbf{Y}$ if and only if $\mathbf{Z}$ intersects every path between $\mathbf{X}$ and $\mathbf{Y}$ in $(\mathcal{G}_{possibleAn(\mathbf{X},\mathbf{Y},\mathbf{Z})})^m$.*

This also shows that in RCGs separation based on (almost) definite status paths is equivalent to other definitions of separation proposed for general chain graphs (Frydenberg 1990; Bouckaert and Studenỳ 1995) that are also equivalent to separation in the moral graph.

7 Covariate Adjustments in RCGs

In this section we prove the first statement of Proposition 8, i.e. the correctness of Step 4 of algorithm FINDADJSET, showing the following:

Theorem 17. *Let $\mathcal{G}$ be an RCG. Then the constructive back-door criterion (Definition 4) holds in $\mathcal{G}$ for sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, if and only if $\mathbf{Z}$ is an adjustment set relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$.*

To prove the theorem we show that the criterion holds for an arbitrary RCG $\mathcal{G}$ if and only if it holds for every DAG in $CE(\mathcal{G})$. An alternative proof could show that the generalized adjustment criterion (GAC) of (Perković et al. 2015) can be stated for RCGs and that their proof also applies to RCGs. The method of (van der Zander, Liśkiewicz, and Textor 2014) could then be used to transform the modified GAC in terms of a constructive back-door graph. However, in both proofs the technical difficulty emerges that not every proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ of an RCG $\mathcal{G}$ is an RCG itself. To cope with this problem we need two auxiliary lemmas. The first one shows that the result of Lemma 13 holds in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ even if $\mathcal{G}^{pbd}_{\mathbf{XY}}$ is not an RCG.

Lemma 18. *Let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be an RCG, and let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{V}$ be pairwise disjoint subsets of variables. In the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ the set $\mathbf{Z}$ blocks every almost definite status path between $\mathbf{X}$ and $\mathbf{Y}$ if and only if $\mathbf{Z}$ blocks every definite status path between $\mathbf{X}$ and $\mathbf{Y}$.*

The second lemma shows that the proper back-door graph always is an RCG, if there exists at least one adjustment set.

Lemma 19. *If the proper back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ of an RCG $\mathcal{G}$ is not an RCG, no adjustment set exists relative to $(\mathbf{X}, \mathbf{Y})$ in $\mathcal{G}$.*

These lemmas are also useful to obtain fast algorithms. Due to Lemma 19 the algorithms can assume that the proper back-door graph RCG is an RCG, as soon as they have found an adjustment set (possible in linear time) and do not need to explicitly test the RCG-properties which would take $\mathcal{O}(n^{2.373})$ time (see the next section).

With the help of Lemma 18 the algorithms can work with *almost* definite status paths, which are more convenient to handle than definite status paths because testing if a path is of definite status requires $\mathcal{O}(nm) = \mathcal{O}(n^3)$ time to verify that the nodes surrounding a definite non-collider with undirected edges on the path are not adjacent.

This efficiency becomes relevant, if the input graph is already an RCG, e.g. a DAG or CPDAG, and we can skip the $\mathcal{O}(n^4)$ algorithm to transform it to an RCG.

8 Algorithms for RCGs

Recognition of RCGs. Several structure learning algorithms return a mixed graph making no additional assumptions about its properties. Thus, before further processing could be performed, it is necessary to verify if the returned graph satisfies all required conditions. Our general algorithm assumes that the input graph is a CG. However, if we know that it is an RCG, Steps 1 and 2 can be omitted.

Because every class of Markov equivalent DAGs is represented by a unique CPDAG, it is possible to test if a given

chain graph $\mathcal{G}$ is a CPDAG by finding one consistent DAG extension $\mathcal{D}$ of $\mathcal{G}$, generating the CPDAG $\mathcal{G}'$ for $\mathcal{D}$ and comparing the resulting graph $\mathcal{G}'$ with $\mathcal{G}$. Graph $\mathcal{G}$ is a CPDAG, if and only if $\mathcal{G} = \mathcal{G}'$. Using the CG-to-DAG conversion algorithm of (Andersson, Madigan, and Perlman 1997) and the DAG-to-CPDAG conversion of (Chickering 1995) this can be done in time $\mathcal{O}(m \log n)$. Alternatively, if the degree of the graph is bounded by a constant k, the algorithm of (Chickering 2002) decreases the running time to $\mathcal{O}((n+m)k^2)$.

However, to recognize RCGs such an approach does not work and one needs to test the conditions of Definition 6 directly. The first property – the chordality of components – can be tested with lexicographic breadth-first search in linear time (Rose, Tarjan, and Lueker 1976). A naive test of the second condition, that $A \to B - C$ does not exist as induced subgraph, is possible in time $\mathcal{O}(nm)$. Here we present a more sophisticated method: Let D, U, M be three adjacency matrices corresponding to directed, undirected, resp. missing edges. I.e. $D[i,j] = 1$ if $i \to j \in \mathbf{E}$, $U[i,j] = 1$ if $i - j \in \mathbf{E}$ and $M[i,j] = 1$ if no edge exists between i and j. All other matrix elements are 0. The trace of the product $Tr[D \cdot U \cdot M]$ is zero if and only if the second condition is satisfied by the graph, since it corresponds to cycles $i \to j - k$ -no-edge- i. Thus the second condition can be verified in time $\mathcal{O}(n^\alpha)$, with $\alpha < 2.373$ (Le Gall 2014), using a fast matrix multiplication algorithm. This dominates the time complexity of the whole recognition algorithm.

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

Testing, computing, and enumerating separating sets. Before we can describe the algorithms involving adjustment sets, we need to describe the algorithms for separating sets, since the constructive back-door criterion reduces adjustment to separation.

A modified Bayes-Ball algorithm (Shachter 1998) can be used to test if a given set $\mathbf{Z}$ d-separates $\mathbf{X}$ and $\mathbf{Y}$. Thereby a standard search is performed and the algorithm only continues through a node when the entering and leaving edge form an almost definite status path. As there are only three kinds of edges tracking the kind of the entering and leaving edge requires a constant overhead and the algorithm runs in $\mathcal{O}(n+m)$.

The algorithms to find or enumerate separating sets will take as arguments an RCG, disjoint node sets $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ and will return one or more sets $\mathbf{Z}$ that d-separate $\mathbf{X}$ from $\mathbf{Y}$ under the constraint $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$. Later, using these algorithms for adjustment sets, the constraint given by the set $\mathbf{R}$ corresponds to the nodes forbidden by the condition (a) of the criterion (Definition 4). The constraint given by $\mathbf{I}$ helps to enumerate all such sets.

A single d-separator can be found using a closed form solution that can be constructed in time $\mathcal{O}(n+m)$:

Lemma 20. *Let $\mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$ be sets of nodes with $\mathbf{I} \subseteq \mathbf{R}$, $\mathbf{R} \cap (\mathbf{X} \cup \mathbf{Y}) = \emptyset$. If there exists a d-separator $\mathbf{Z}_0$, with $\mathbf{I} \subseteq \mathbf{Z}_0 \subseteq \mathbf{R}$ then $\mathbf{Z} = possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \cap \mathbf{R}$ is a d-separator.*

Observing certain variables can be very expensive, so it is desirable to not just find any d-separator, but a d-separator $\mathbf{Z}$ that contains a minimal number of nodes, in the sense that no subset $\mathbf{Z}' \subset \mathbf{Z}$ is a d-separator. The above lemma implies:

Corollary 21. *Let $\mathbf{X}, \mathbf{Y}, \mathbf{I}$ be sets of nodes. Every minimal set over all d-separators containing $\mathbf{I}$ is a subset of $\mathbf{Z} = possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$.*

This means that every minimal d-separator $\mathbf{Z}$ is a vertex cut in the moral graph $(\mathcal{G}_{possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})})^m$. Because this moral graph is independent of $\mathbf{Z}$, it is sufficient to search a standard vertex cut within this undirected graph. van der Zander, Liśkiewicz, and Textor (2014) provide the necessary algorithms, and describe an $\mathcal{O}(n^2)$ algorithm for testing and finding a minimal d-separator by searching nodes that are reachable from $\mathbf{X}$ as well as $\mathbf{Y}$ in the moral graph. Finding a d-separator that is not just a minimal d-separator, but a minimum d-separator with a minimum cost according to some linear cost function that assigns a certain weight to every node can be done in $\mathcal{O}(n^3)$ using a max-flow algorithm.

After finding a single d-separator, it is also interesting to know which other d-separators exist and enumerate all of them. For this task the algorithms of van der Zander, Liśkiewicz, and Textor can also be used, since they can enumerate any class of sets given a test for the existence of a set $\mathbf{Z}$ in the class with $\mathbf{I} \subseteq \mathbf{Z} \subseteq \mathbf{R}$ by enumerating all sets and aborting branches in the search graph that will not lead to a solution. The runtime has a delay linear to the maximal set size and complexity of the test, i.e. between every found d-separator $\mathcal{O}(n(n+m))$ time passes and $\mathcal{O}(n^3)$ between every minimal d-separator.

Testing, Computing, and Enumerating Adjustment Sets. Now we are ready to describe algorithms to find, test and enumerate arbitrary, minimal and minimum adjustment sets. For each problem such an algorithm calculates the set $PCP(\mathbf{X}, \mathbf{Y})$, constructs the proper back-door graph in linear time and solves the corresponding separator problem restricted to $\mathbf{R}' = \mathbf{R} \setminus possibleDe(PCP(\mathbf{X}, \mathbf{Y}))$. The algorithm has a runtime that is the same as the runtime of the corresponding algorithm for d-separation and even the runtime of the corresponding algorithm for DAGs.

For the testing problems this means, we test if $\mathbf{Z} \cap possibleDe(PCP(\mathbf{X}, \mathbf{Y})) = \emptyset$. If this is not true, $\mathbf{Z}$ is not an adjustment set, otherwise it is an adjustment set, if and only if it is a d-separator in the back-door graph.

For a singleton $\mathbf{X}$ the d-separation algorithms can be used directly. For sets $\mathbf{X}$ with more than one element, it is also necessary to test if the back-door graph $\mathcal{G}^{pbd}_{\mathbf{XY}}$ is an RCG. If not, no adjustment set exists. This test can be done as described in Section 8, but it is faster to test in $\mathcal{O}(n+m)$ if $\mathbf{Z} = possibleAn(\mathbf{X} \cup \mathbf{Y}) \setminus possibleDe(PCP(\mathbf{X}, \mathbf{Y}))$ is an adjustment set, i.e. a d-separator in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. This can be tested with the Bayes-Ball-like search, which will work in any chain graph, not just RCGs. We know from Lemma 19

that if $\mathbf{Z}$ is an adjustment set, the graph is an RCG. If $\mathbf{Z}$ is not an adjustment set, no adjustment set exists and any further search can be aborted.

This also completes the proof of Proposition 8.

9 Discussion

We have introduced restricted chain graphs as a new graph class which includes DAGs and CPDAGs and still has an algorithmic simple notion of d-separation. For these RCGs we give a constructive back-door criterion that reduces problems related to adjustment sets to problems involving d-separation. This leads to efficient algorithms to find, test and enumerate adjustment sets as well as minimal and minimum adjustment sets in chain graphs. The algorithms are easily implementable and our software is accessible online at `http://dagitty.net`. It remains an open problem to extend our methods to arbitrary mixed graphs.

If a given graph is a CPDAG or an arbitrary RCG, our algorithms run in linear time. It is interesting that the problems involving adjustment sets for such graphs are not harder than for DAGs.

References

Acid, S., and De Campos, L. M. 1996. An algorithm for finding minimum d-separating sets in belief networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, 3–10. Morgan Kaufmann Publishers Inc.

Andersson, S. A.; Madigan, D.; Perlman, M. D.; et al. 1997. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics* 25(2):505–541.

Andersson, S. A.; Madigan, D.; and Perlman, M. D. 1997. On the Markov equivalence of chain graphs, undirected graphs, and acyclic digraphs. *Scandinavian Journal of Statistics* 24(1):81–102.

Bouckaert, R. R., and Studenỳ, M. 1995. Chain graphs: semantics and expressiveness. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*. Springer. 69–76.

Chickering, D. M. 1995. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*, 87–98. Morgan Kaufmann.

Chickering, D. M. 2002. Learning equivalence classes of Bayesian-network structures. *The Journal of Machine Learning Research* 2:445–498.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms, Second Edition*. The MIT Press, 2nd edition.

Frydenberg, M. 1990. The chain graph Markov property. *Scandinavian Journal of Statistics* 17:333–353.

Lauritzen, S., and Wermuth, N. 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics* 17:31–57.

Le Gall, F. 2014. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, 296–303. ACM.

Meek, C. 1995. Causal inference and causal explanation with background knowledge. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*, 403–410. Morgan Kaufmann.

Pearl, J. 1995. Causal diagrams for empirical research. *Biometrika* 82(4):669–688.

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

Perković, E.; Textor, J.; Kalisch, M.; and Maathuis, M. 2015. A complete generalized adjustment criterion. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, 682–691. AUAI Press.

Rose, D. J.; Tarjan, R. E.; and Lueker, G. S. 1976. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.* 5(2):266–283.

Shachter, R. D. 1998. Bayes-ball: The rational pastime. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 480–487. Morgan Kaufmann.

Shpitser, I.; VanderWeele, T.; and Robins, J. 2010. On the validity of covariate adjustment for estimating causal effects. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 527–536. AUAI Press.

Shpitser, I. 2012. Appendum to on the validity of covariate adjustment for estimating causal effects. unpublished manuscript.

Takata, K. 2010. Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph. *Discrete Applied Mathematics* 158:1660–1667.

Textor, J., and Liśkiewicz, M. 2011. Adjustment criteria in causal diagrams: An algorithmic perspective. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 681–688. AUAI Press.

Tian, J.; Paz, A.; and Pearl, J. 1998. Finding minimal d-separators. Technical Report R-254, University of California, Los Angeles.

van der Zander, B.; Liśkiewicz, M.; and Textor, J. 2014. Constructing separators and adjustment sets in ancestral graphs. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 907–916. AUAI Press.

Verma, T., and Pearl, J. 1990. Equivalence and synthesis of causal models. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence*, 255–270. Elsevier.

Verma, T., and Pearl, J. 1992. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proceedings of the 8th Annual Conference on Uncertainty in Artificial Intelligence*, 323–330. Morgan Kaufmann.

Zhang, J. 2008. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research* 9:1437–1474.

A APPENDIX

A.1 Further notation

First we need to define additional graph preliminaries: The neighbors of a node V are all the nodes connected to V by an undirected edge. The parents of a node V are all the nodes connected to V by a directed edge pointing to V. We use $Ne(\mathbf{V})$ or $Pa(\mathbf{V})$ to refer to the set of neighbors or parents of a node set $\mathbf{V}$.

We abbreviate a directed path from A to B as $A \overset{*}{\to} B$, and as $A \overset{+}{\to} B$ if it has at least one edge, i.e. if $A \neq B$. A path in the opposite direction, from B to A, is written as $A \overset{*}{\leftarrow} B$, and as $A \overset{+}{\leftarrow} B$, if $A \neq B$. An undirected path is similarly written as $A \overset{+}{—} B$ or $A \overset{*}{—} B$. A possible directed path is written as $A \ (\to \overset{*}{|} -) \ B$ or $A \ (\to \overset{+}{|} -) \ B$. A single edge of unknown type is written as $\sim$, and a path of unknown type as $A \overset{*}{\sim} B$ or $A \overset{+}{\sim} B$. Paths written in this combination can be arbitrarily combined, e.g. $A \to B \leftarrow C$ for a collider; or $A \overset{*}{—} \overset{+}{\to} B$ for a path that starts with (possible empty) undirected part and ends with a directed path.

A.2 Generalized paths

The main paper gives a definition for d-connected paths which is the definition usually used by the causality community, but it is not clear, why this definition was chosen. What happens if a path is allowed to visit the same node multiple times? Or if colliders are also opened by possible descendants? In this section we examine these variations of paths and show that almost all of them define the same connectivities. This greatly simplifies the later proofs, since we only need to show the existence of one kind of path, and then can assume we have a path of a different kind, e.g. turning an almost definite status path into a definite status path.

A *walk* is defined like a path, but it is allowed to visit the same node multiple times. Walks are commonly used and often d-connectedness of walks is defined such that a walk is connected by a set $\mathbf{Z}$, if every collider is in $\mathbf{Z}$ and every non-collider is not in $\mathbf{Z}$. We will call such connected walks *d-in-*connected. The main paper defines d-connected paths, which requires that colliders are ancestors of nodes in $\mathbf{Z}$, here we will call such paths *d-An-*connected. Correspondingly, if all colliders are possible ancestors of nodes in $\mathbf{Z}$ (and non-colliders are not in $\mathbf{Z}$), the path is *d-possibleAn-*connected. All d-·-connected definitions apply to walks and paths. We will abbreviate definite status as ds- and almost definite status as ads-.

We can show the following relationships between these walks and paths:

Lemma 22. *Given an RCG $\mathcal{G}$, nodes X and Y, and a node set $\mathbf{Z}$ the following statements are equivalent:*

1. *There exists a d-possibleAn-connected ads-walk between X and Y given $\mathbf{Z}$.*
2. *There exists a d-An-connected ads-walk between X and Y given $\mathbf{Z}$.*
3. *There exists a d-in-connected ads-walk between X and Y given $\mathbf{Z}$.*
4. *There exists a d-An-connected ads-path between X and Y given $\mathbf{Z}$.*
5. *There exists a d-possibleAn-connected ads-path between X and Y given $\mathbf{Z}$.*

Proof. We will do a ring proof.

$(1 \to 2)$: Let $\pi : X \overset{+}{\sim} Y$ be a d-*possibleAn*-connected ads-walk. If π is not d-*An*-connected, there is a subwalk $U \to C \leftarrow V$, such that C is a possible ancestor of a node in $\mathbf{Z}$, but not an ancestor. Let Z be the first such node, i.e. the path τ between C and Z does not contain any other node of $\mathbf{Z}$. From Lemma 12 it follows that there are paths $U \overset{+}{\to} Z$ and $V \overset{+}{\to} Z$ that will not contain nodes not in τ. So we can replace $U \to C \leftarrow V$ with $U \overset{+}{\to} Z \overset{+}{\leftarrow} V$ in which the collider is an ancestor of $\mathbf{Z}$. This can be repeated, till every collider is an ancestor.

$(2 \to 3)$: Let $\pi : X \overset{+}{\sim} Y$ be a d-*An*-connected ads-walk. If π is not d-*in*-connected, there is a subwalk $U \to C \leftarrow V$, such that C is an ancestor of a node in $\mathbf{Z}$, but not in $\mathbf{Z}$. Let Z be the first such node, i.e. the path $C \overset{+}{\to} Z$ does not contain any other node of $\mathbf{Z}$. So we can replace $U \to C \leftarrow V$ with $U \overset{+}{\to} Z \overset{+}{\leftarrow} V$. This can be repeated, till every collider is in $\mathbf{Z}$.

$(3 \to 4)$: Let $\pi' : X \overset{+}{\sim} Y$ be the shortest d-*in*-connected ads-walk for which no d-*An*-connected ads-path exists. If it is not a path there is a node V that occurs more than once. If V is X or Y we directly trim this walk to a shorter walk π. Otherwise we can get a shorter walk π by replacing $U \sim V \overset{+}{\sim} V \sim W$ with $U \sim V \sim W$. If this walk is not of almost definite status, there occurs $U \to V - W$ or $U - V \leftarrow W$. We can skip the undirected part of the walk due to Lemma 12 and get a d-*An*-connected ads-walk, because a node adjacent to an undirected edge in π' cannot be in $\mathbf{Z}$ and it cannot introduce a new collider, because π' is of almost definite status.

If π was of almost definite status, V is either an almost definite non-collider or a collider. In former case, it is not in $\mathbf{Z}$, since at least one of the edges of V in π' does not have an arrowhead. In latter case the original walk was $U \to V \overset{+}{\sim} V \leftarrow W$, so there occurs a direction swap $\to Z \leftarrow$ in $V \overset{+}{\sim} V$, and V is an ancestor of some $Z \in \mathbf{Z}$. Thus π is d-*An*-connected.

Either π is a path, has a corresponding path or is a contradiction to π' being the shortest walk for which no path exists.

$(4 \to 5)$: This is trivial, since every ancestor is a possible ancestor.

$(5 \to 1)$: This is trivial, since every path is a walk. □

Corollary 23. *Given an RCG $\mathcal{G}$, nodes X and Y, and a node set $\mathbf{Z}$ the following statements are equivalent:*

1. *There exists a d-possibleAn-connected ads-walk between X and Y given $\mathbf{Z}$.*
2. *There exists a d-An-connected ads-walk between X and Y given $\mathbf{Z}$.*

3. *There exists a d-in-connected ads-walk between X and Y given* $\mathbf{Z}$.
4. *There exists a d-An-connected ads-path between X and Y given* $\mathbf{Z}$.
5. *There exists a d-possibleAn-connected ads-path between X and Y given* $\mathbf{Z}$.
6. *There exists a d-possibleAn-connected ds-walk between X and Y given* $\mathbf{Z}$.
7. *There exists a d-An-connected ds-walk between X and Y given* $\mathbf{Z}$.
8. *There exists a d-in-connected ds-walk between X and Y given* $\mathbf{Z}$.
9. *There exists a d-An-connected ds-path between X and Y given* $\mathbf{Z}$.
10. *There exists a d-possibleAn-connected ds-path between X and Y given* $\mathbf{Z}$.

Proof. This follows from the combination of lemma 13 and 22: Lemma 22 shows that 1 to 5 are equivalent. Lemma 13 shows that 4 and 9 are equivalent. Since ds-* is a stronger condition than ads-* each of 6. to 10. implies 1. to 5.. The steps $9 \to 10 \to 6$ are trivial. The steps $1 \to 2 \to 3$ in the proof of Lemma 22 do not depend on ads, so they also show $6 \to 7 \to 8$ here. $8 \to 9$ follows from $8 \to 3 \to 4 \to 9$. □

A.3 Proofs

Here we give proofs omitted from the main paper:
Proof of Lemma 11:

Proof. If $\mathcal{G}$ contains a chain component that is not chordal, it has an undirected cycle $V_1 - V_2 - \ldots - V_k - V_1$ with $k > 3$, such that V_i and V_j are not connected by an undirected edge, unless $V_j \in \{V_{i-1}, V_{i+1}\}$ (we set $V_0 = V_k, V_{k+1} = V_1$). No V_i, V_j are connected by a directed edge, otherwise there would be a semi-directed cycle and $\mathcal{G}$ would not be a chain graph. I.e. the subgraph induced by $\{V_1, \ldots, V_k\}$ is exactly the cycle $V_1 - V_2 - \ldots - V_k - V_1$. Thus every orientation of this cycle that does not create a new v-structure creates a directed cycle. So $CE(\mathcal{H}) = \emptyset$. □

Proof of Lemma 16:

Proof. "⇒": Let $\pi : X \stackrel{+}{-} Y$ be the shortest path in the moral graph $\mathcal{G}^m_{possibleAn(\mathbf{X},\mathbf{Y},\mathbf{Z})}$ connecting $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$ and not containing a node of $\mathbf{Z}$. It corresponds to a sequence $\pi' : X \stackrel{+}{\sim} Y$ in the RCG with the same nodes. If π' contains nodes V_i, V_{i+1} that are not connected, they are moralized parents of a node C with $V_i \to C \leftarrow V_{i+1}$ and we can insert C after V_i.

If π' is not a definitive status walk, it contains $U \to V - W, U - V \leftarrow W$ or $U - V - W$ with $U - W$. In every case U and W are connected in $\mathcal{G}$ and $\mathcal{G}^m_{possibleAn(\mathbf{X},\mathbf{Y},\mathbf{Z})}$, thus replacing $U - V - W$ by $U - W$ in π creates a shorter path and contradicts π being the shortest.

No node of π is in $\mathbf{Z}$, so if π' is not d-*possibleAn*-connected, there is a collider $V_i \to C \leftarrow V_{i+1}$, such that $C \notin possibleAn(\mathbf{Z})$. Then $C \in possibleAn(\mathbf{X} \cup \mathbf{Y})$ and there exists a possible directed path from C to $X' \in \mathbf{X}$ or $Y' \in \mathbf{Y}$ that does not intersect $\mathbf{Z}$. From Lemma 12 it follows that there also exists a directed path from V_i and V_{i+1} to X' or Y' not intersecting $\mathbf{Z}$. We can replace the collider C with this path and truncate the walk accordingly.

"⇐": Assume there exists a d-connected definite status path $\pi : X \stackrel{+}{\sim} Y$ in $\mathcal{G}$ given $\mathbf{Z}$ between $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$. Every node on this path is a possible ancestor of a node in $\mathbf{X}$, $\mathbf{Y}$ or $\mathbf{Z}$, thus the path exists in $\mathcal{G}^m_{possibleAn(\mathbf{X},\mathbf{Y},\mathbf{Z})}$. All nodes in $\pi \cap \mathbf{Z}$ are colliders $U \to V \leftarrow W$, which can be replaced in the path with the edge $U - W$ inserted during moralization. □

Corollary 24. *In RCGs d-separation of definite status paths is equivalent to the c-separation of (Bouckaert and Studený 1995).*

Proof of Lemma 18

Proof. If there exists a definite status path between $\mathbf{X}$ and $\mathbf{Y}$, it is also an almost definite status path. On the other hand, if there a exists a d-connected almost definite status path π between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, π is also a d-connected almost definite status path in $\mathcal{G}$ and due to Lemma 13 there exists a d-connected definite status path π' in $\mathcal{G}$. Both paths have the same directed edges and thus π' is also a d-connected definite status path in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. □

Proof of Lemma 19

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

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

According to lemma 12 all non-leading undirected edges can be removed from $B - \pi$, which creates a proper definite status path from $\mathbf{X}$ to $\mathbf{Y}$. Since this path starts with an undirected edge, it exists in the back-door graph as well and must be blocked by every adjustment set. But it can only blocked by nodes in $PCP(\mathbf{X}, \mathbf{Y})$, so no adjustment set exists. □

Proof of Lemma 20:

Proof. First we show that $\mathbf{Z}'_0 = \mathbf{Z}_0 \cap possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$ is a d-separator. Assume it is not, then there exist a proper definite status path π from X to Y with $X \in \mathbf{X}, Y \in \mathbf{Y}$. If π does not contain a collider, every node on π is a possible ancestor of X or Y, and thus $\mathbf{Z}'_0$ blocks π. If π does contain colliders, it is blocked, unless every collider is in $\mathbf{Z}'_0$ and a possible ancestor of $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}$. Then every node on π is in $possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$ and π is blocked by $\mathbf{Z}'_0$. Hence $\mathbf{Z}'_0$ is a vertex cut in the moral graph $\mathcal{G}^m_{possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}_0)}$.

Since $\mathbf{I} \subseteq \mathbf{Z}'_0$, we have $possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I}) \subseteq possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}'_0)$. But also $\mathbf{Z}'_0 \subseteq possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{I})$, so $possibleAn(\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}'_0) \subseteq possibleAn(\mathbf{X} \cup$

$\mathbf{Y} \cup \mathbf{I}$). So $\mathbf{Z}'_0$ is also a vertex cut in the moral graph $\mathcal{G}^m_{possibleAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})}$.

Since a vertex cut with added nodes is still a vertex cut and $\mathbf{Z}$ includes every allowed node in the moral graph, $\mathbf{Z}$ is also a d-separator. □

Direct proof of the main result In this section we give a direct proof of the main result by showing that our criterion holds for an RCG $\mathcal{G}$ iff the criterions of (Shpitser 2012) and (van der Zander, Liśkiewicz, and Textor 2014) hold in every DAG $\mathcal{D} \in CE(\mathcal{G})$. This proof does not depend on the GAC of (Perković et al. 2015) and is simpler as it only has to handle a single class of graphs (RCGs) and not multiple graph classes like DAGs, CPDAGs or PAGs.

Lemma 25. *All nodes in a chain component of an RCG have the same set of parents.*

Proof. Assume there exist a node V that has a parent P that is not a parent of a node W which is a neighbor of V in the same component. Then $P \to V - W$ exists in the graph, so according to Lemma 12 there exists an edge $P \to W$ and P is a parent of W.

Since all neighbors have the same parents and the component is connected, all nodes in the component have the same parents. □

Lemma 26. *For every node V in a chain component $\mathcal{C}$ of an RCG $\mathcal{G}$, the edges of $\mathcal{C}$ can be oriented such that for every other node W in $\mathcal{C}$ there exists a directed path $V \overset{+}{\to} W$.*

Proof. Since all nodes in the chain component have the same parents, we can start the MCS orientation algorithm of (Andersson, Madigan, and Perlman 1997) at node V. It returns a sequence $\alpha_1, \ldots, \alpha_{|\mathcal{C}|}$ such that $\alpha_1 = V$ and if every edge $\alpha_i - \alpha_j$ with $i < j$ is oriented to $\alpha_i \to \alpha_j$ the resulting graph is acyclic without immoralities.

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

□

Lemma 27. *A definite non-collider V on a path (walk) π in an RCG $\mathcal{G}$ is not a collider on π in any $\mathcal{D} \in CE(\mathcal{G})$.*

Proof. If V was a collider $\to V \leftarrow$ in $\mathcal{D}$, it would occur as $\to V-$, $-V \leftarrow$ or $-V-$ in $\mathcal{G}$. Only in the last case it could be a definite non-collider in $\mathcal{G}$, but then the adjacent nodes must be different and unconnected. So $\to V \leftarrow$ is not a valid orientation. □

Lemma 28. *If there exists a d-connected given $\mathbf{Z}$ definite status path $X \overset{+}{\sim} Y$ in an RCG $\mathcal{G}$, then there exists a d-connected path $X \overset{+}{\sim} Y$ given $\mathbf{Z}$ in every DAG $\mathcal{D} \in CE(\mathcal{G})$ that contains exactly the same nodes.*

Proof. In $\mathcal{G}$ all nodes on the definite status path are either definite non-colliders not in $\mathbf{Z}$ or colliders in $\mathbf{Z}$, so they are non-colliders not in $\mathbf{Z}$ (due to lemma 27) or colliders in $\mathbf{Z}$ in $\mathcal{D}$. □

The previous lemma does not hold for almost definite status paths, which makes definite status paths an useful definition, despite the much simpler definition of almost definite status paths.

Lemma 29. *Let $\mathcal{G}$ be an RCG and $\mathcal{D} \in CE(\mathcal{G})$. If there exists a d-connected path $\pi : X \overset{+}{\sim} Y$ given $\mathbf{Z}$ in $\mathcal{D}$, there exists a d-connected given $\mathbf{Z}$ almost definite status path $\pi_G : X \overset{+}{\sim} Y$ in $\mathcal{G}$ with the following properties:*

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

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

Proof. Let π be the shortest d-connected path between X and Y given $\mathbf{Z}$ in $\mathcal{D}$. If this path is not an almost definitive status path in $\mathcal{G}$ the path in $\mathcal{G}$ contains $U \to V - W$ or $U - V \leftarrow W$. In the first case there has to exist an edge $U \to W$ in $\mathcal{G}$ due to Lemma 12, so we can replace $U \sim V \sim W$ by $U \to W$ to form a shorter path π' in $\mathcal{D}$. If the edge $V \sim W$ is $V \to W$ in $\mathcal{D}$, the node W is a collider on π' iff it is a collider on π, and π' is a directed path iff π' is one. If the edge is $V \leftarrow W$, V is a collider and an ancestor of $\mathbf{Z}$, so W is an ancestor of $\mathbf{Z}$ in $\mathcal{D}$ and a possible ancestor of $\mathbf{Z}$ in $\mathcal{G}$. After the replacement π' is a d-connected shorter path than π.

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

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

It remains to show that there also exists a *d-connected* almost definite status path with these properties, i.e. such that all colliders are ancestors of $\mathbf{Z}$. Let π be a path that is of almost definite status in $\mathcal{G}$ and a d-connected path between X and Y given $\mathbf{Z}$ in $\mathcal{D}$, chosen such that it is the path with the lowest number of blocked colliders in $\mathcal{G}$. Let V be the first such collider, i.e. π contains $U \to V \leftarrow W$ and V is an ancestor of $\mathbf{Z}$ in $\mathcal{D}$ and not in $\mathcal{G}$. Then there exist paths $U \to V \ (\overset{+}{\to}|-) \ Z$ and $W \to V \ (\overset{+}{\to}|-) \ Z$ with $Z \in \mathbf{Z}$ that do not contain any other node of $\mathbf{Z}$. Due to Lemma 12 there are also d-connected paths $U \overset{+}{\to} Z$ and $W \overset{+}{\to} Z$ in $\mathcal{G}$ and $\mathcal{D}$. These paths intersect in a node V', and we can replace $U \to V \leftarrow W$ by $U \overset{+}{\to} V' \overset{+}{\leftarrow} W$, unless the paths also intersect π in a node $V'' \notin \{U, V, W\}$. If one of them intersects $X \overset{*}{\sim} U$, we can replace the beginning of the path with $X \overset{*}{\sim} V'' \overset{+}{\leftarrow} U$, and if they intersect $W \overset{*}{\sim} Y$, we

replace the end with $V'' \overset{+}{\leftarrow} W \overset{*}{\sim} Y$, reducing the number of blocked colliders.

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

□

Proof of the main theorem 17:

Proof. Let $\mathcal{G}$ be an RCG.

If $\mathcal{G}$ does not contain an undirected edge, i.e. it is a DAG and $CE(\mathcal{G}) = \{\mathcal{G}\}$, the criterion is equivalent to the criterion given in (van der Zander, Liśkiewicz, and Textor 2014): In the absence of undirected edges, the definition of descendants and possible descendants are equal as well as the definition of d-connected almost definite status paths and d-connected paths.

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

Assume $\mathcal{G}$ does not satisfy CBC(a), i.e. there exist nodes $Z \in \mathbf{Z}, W \in PCP_{\mathcal{G}}(\mathbf{X}, \mathbf{Y})$ with paths $\mathbf{X}$ $(\overset{+}{\rightarrow}|-)$ W $(\overset{*}{\rightarrow}|-)$ $\mathbf{Y}$ and W $(\overset{*}{\rightarrow}|-)$ Z. Due to Lemma 12 we can assume these paths have the form $\overset{*}{—}\overset{*}{\rightarrow}$, using possibly another W. The path $X \overset{+}{\sim} W \overset{*}{\sim} Z$ has this form, too, unless the paths are $X \overset{*}{—} \overset{+}{\rightarrow} W \overset{*}{\rightarrow} Y$ and $W \overset{+}{—} \overset{*}{\rightarrow} Z$. If the first path starts with an undirected edge $X - W' \overset{*}{—} \overset{+}{\rightarrow} W \overset{+}{\rightarrow} Y$, we can use W' instead of W and get a path $W' \overset{*}{—} \overset{*}{\rightarrow} Z$ with Lemma 12.

In any case these paths contain only a single undirected subpath, either $X \overset{*}{—}$ as common prefix or $W \overset{*}{—}$. According to lemma 26 there exist a DAG $\mathcal{D} \in CE(\mathcal{G})$ in which this subpath is oriented to a directed path $X \overset{*}{\rightarrow}$ or $W \overset{*}{\rightarrow}$, so Z is also a descendant of $PCP_{\mathcal{D}}(\mathbf{X}, \mathbf{Y})$ in $\mathcal{D}$ and $\mathcal{D}$ violates CBC(a).

Assume $\mathcal{G}$ does not satisfy CBC(b), i.e. there exists a (proper) d-connected almost definite status path π between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}^{pbd}_{\mathbf{XY}}$. Let $\mathcal{D} \in CE(\mathcal{G})$ a DAG that does not violate CBC(a). Due to lemma 13 there exists a definite status path π' between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{G}$ that has the same directed edges and thus exist in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ as well. Due to lemma 27 the path π' exists in $\mathcal{D}$ as d-connected path. If π' starts with $X-$ in $\mathcal{G}$, we can assume it starts with $X \leftarrow$ in $\mathcal{D}$, and the first edge is not removed in $\mathcal{D}^{pbd}_{\mathbf{XY}}$.

Every directed edge of $\mathcal{G}^{pbd}_{\mathbf{XY}}$ also exists in $\mathcal{D}^{pbd}_{\mathbf{XY}}$: Otherwise there would be an edge $X \rightarrow D$ removed in $\mathcal{D}^{pbd}_{\mathbf{XY}}$, which is not removed in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, i.e. a node in $PCP_{\mathcal{D}}$ that is not a node in $PCP_{\mathcal{G}}$, i.e. a proper path $\mathbf{X} \overset{+}{\rightarrow} W \overset{*}{\rightarrow} \mathbf{Y}$ in $\mathcal{D}$ which does not correspond to a proper path $\mathbf{X}$ $(\overset{+}{\rightarrow}|-)$ W $(\overset{*}{\rightarrow}|-)$ $\mathbf{Y}$ in $\mathcal{G}$.

Since π' is proper, no node of $\mathbf{X}$ occurs as a non-endpoint node, so no removed edge occurs in π' and the edges of π' exists (oriented) in $PCP_{\mathcal{D}}$.

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

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

Assume there is a DAG $\mathcal{D}$ that does not satisfy CBC(b). Then there exists a proper d-connected path π_D in $\mathcal{D}^{pbd}_{\mathbf{XY}}$ and $\mathcal{D}$, which corresponds to a d-connected definite status path π_G in $\mathcal{G}$. If any other edge of π_G than the first one does not exist in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ π_D is not proper or an directed edge $X \rightarrow \mathit{PCP}(\mathbf{X}, \mathbf{Y})_{\mathcal{G}}$ was inserted in π_G which violates CBC(a) in $\mathcal{D}$ due to lemma 29. If π_G does not start with a directed edge $X \rightarrow \mathit{PCP}(\mathbf{X}, \mathbf{Y})_{\mathcal{G}}$, it also exist in $\mathcal{G}^{pbd}_{\mathbf{XY}}$ and $\mathcal{G}$ violates CBC(b). Every open collider C in $\mathcal{G}$ is also an open collider in $\mathcal{G}^{pbd}_{\mathbf{XY}}$, otherwise there would be a path $C \overset{*}{\rightarrow} X \rightarrow \mathit{PCP}(\mathbf{X}, \mathbf{Y}) \overset{*}{\rightarrow} \mathbf{Z}$ in $\mathcal{G}$, so $\mathcal{G}$ would violate CBC(a).

If the path starts with an edge $X \rightarrow \mathit{PCP}(\mathbf{X}, \mathbf{Y})_{\mathcal{G}}$ and contains a collider, $\mathcal{G}$ violates CBC(a). If it does not contain a collider, π_D is a directed path $\mathbf{X} \overset{+}{\rightarrow} \mathbf{Y}$ or it would contain $\rightarrow -$ and not be of definite status, so it does not exists in $\mathcal{D}^{pbd}_{\mathbf{XY}}$, or π_G in $\mathcal{D}$ is a directed path and one node other than X has a node of $\mathbf{Z}$ as possible descendant due to lemma 29 and $\mathcal{G}$ violates CBC(b).

□

A.4 ALGORITHMS

TRANSFORM a CG to an RCG The first three steps of FINDADJSET convert an arbitrary chain graph $\mathcal{G}$ to an RCG, or abort with $\perp$, if no equivalent RCG exists. The following algorithm describes these steps in greater detail:

function TRANSFORM-TO-RCG($\mathcal{G} = (\mathbf{V}, \mathbf{E})$)
  function PROCESS-EDGE($A \rightarrow B$)
    for C in $Ne(B)$ **do**
      if A and C are unconnected **then**
        for X in $Pa(C)$ **do**
          if B and X are unconnected **then**
            Abort and return $\perp$
        Change $B - C$ to $B \rightarrow C$
        PROCESS-EDGE($B \rightarrow C$)
  for edge $A \rightarrow B \in \mathbf{E}$ **do**
    PROCESS-EDGE($A \rightarrow B$)
  for every connected, undirected component $\mathcal{C}$ **do**
    if $\mathcal{C}$ is not chordal **then**
      Abort and return $\perp$
  Return the changed graph

Algorithm 1: Transform-To-Rcg

The correctness follows from section 5, and the runtime is $O(mn^2) = O(n^4)$, since PROCESS-EDGE is called once for every directed edge and takes $O(n^2)$ time assuming an $O(1)$ connectivity test.

TESTING For a given RCG $\mathcal{G}$ the problem TESTSEP can be solved with a modified Bayes-Ball algorithm in time $\mathcal{O}(n+m)$.

In the Bayes-Ball algorithm a ball jumps along the edges of the RCG. The ball enters a node through an "incoming" edge and can either "pass", i.e. move to all "outgoing" edges of a certain type, or "bounce", i.e. move back along the same incoming edge.

The following table gives the possible combinations of incoming/outgoing edges. The first 4 rows correspond to the standard Bayes Ball algorithm:

incoming	outgoing	action $\notin \mathbf{Z}$	action $\in \mathbf{Z}$
$\rightarrow$	$\rightarrow$	pass	bounce
$\rightarrow$	$\leftarrow$	bounce	pass
$\leftarrow$	$\rightarrow$	pass	bounce
$\leftarrow$	$\leftarrow$	pass	bounce
$\rightarrow$	$-$	bounce	bounce
$\leftarrow$	$-$	pass	bounce
$-$	$\rightarrow$	pass	bounce
$-$	$\leftarrow$	bounce	bounce
$-$	$-$	pass	bounce

Lemma 30. *This modified Bayes-Ball algorithm sends a ball from X to Y given $\mathbf{Z}$ iff a d-in-connected almost definite status walk from X to Y given $\mathbf{Z}$ exists.*

Proof. The edge pairs that can be passed by the ball in the above table correspond exactly to the edge pairs that are allowed in a d-*in*-connected almost definite status walk. □

function TESTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$)
  Run modified Bayes-Ball from $\mathbf{X}$
  return ($\mathbf{Y}$ not reachable)

Algorithm 2: TestSep

The problem TESTMINSEP can be solved using Algorithm 3 TESTMINSEP in $\mathcal{O}(|\mathbf{E}^m_{An}|) = \mathcal{O}(n^2)$ time. The correctness of this algorithm can be shown by generalizing the results presented in (Tian, Paz, and Pearl 1998) for d-separation. 3 TESTMINSEP, runs in $O(|\mathbf{E}^m_{An}|)$ because R_x and R_y can be computed with an ordinary search that aborts when a node in $\mathbf{Z}$ is reached.

function TESTMINSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$)
  if $\mathbf{Z} \setminus possibleAn(\mathbf{X} \cup \mathbf{Y}) \neq \emptyset$ **then return** false
  if not TESTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}$) **then**
    return false
  $\mathcal{G}'^m \leftarrow \mathcal{G}^m_{possibleAn(\mathbf{X}\cup\mathbf{Y})}$
  $R_x \leftarrow \{Z \in \mathbf{Z} \mid \exists$ path $X - Z$ in $\mathcal{G}'^m$ not intersecting $\mathbf{Z} \setminus \{Z\}\}$
  if $Z \notin R_x$ **then return** false
  $R_y \leftarrow \{Z \in \mathbf{Z} \mid \exists$ path $Y - Z$ in $\mathcal{G}'^m$ not intersecting $\mathbf{Z} \setminus \{Z\}\}$
  if $Z \notin R_y$ **then return** false
  return true

Algorithm 3: TestMinSep

FINDING A D-SEPARATOR The problem can be solved using Algorithm 4 FINDSEP in $\mathcal{O}(n+m)$ time. The correctness follows directly from Lemma 20

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

Algorithm 4: FindSep

FINDING A MINIMAL D-SEPARATOR For a given RCG $\mathcal{G}$ the problem FINDMINSEP can be solved with algorithm 5 FINDMINSEP in $O(|\mathbf{E}^m_{An}|) = O(n^2)$ time.

function FINDMINSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$)
  $\mathcal{G}' \leftarrow \mathcal{G}_{possibleAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})}$
  $\mathcal{G}'^m \leftarrow \mathcal{G}^m_{possibleAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})}$
  $\mathbf{Z}' \leftarrow \mathbf{R} \cap possibleAn(\mathbf{X} \cup \mathbf{Y})$
  Remove from $\mathcal{G}'^m$ all nodes of $\mathbf{I}$
  if not TESTSEP($\mathcal{G}', \mathbf{X}, \mathbf{Y}, \mathbf{Z}$) **then**
    return $\perp$
  Run BFS from $\mathbf{X}$. Whenever a node in $\mathbf{Z}'$ is met, mark it, if it is not already marked and do not continue along the path. When BFS stops, let $\mathbf{Z}''$ be the set of all marked nodes. Remove all markings
  Run BFS from $\mathbf{Y}$. Whenever a node in $\mathbf{Z}''$ is met, mark it, if it is not already marked and do not continue along the path. When BFS stops, let $\mathbf{Z}$ be the set of all marked nodes.
  return $\mathbf{Z} \cup \mathbf{I}$

Algorithm 5: FindMinSep

Algorithm 5 FINDMINSEP begins with the separating set $\mathbf{R} \cap possibleAn(\mathbf{X} \cup \mathbf{Y})$ and finds a subset satisfying the conditions tested by algorithm 3 TESTMINSEP.

FINDING A MINIMUM COST D-SEPARATOR The problem MINCOSTSEP can be solved with algorithm 6 FINDMINCOSTSEP in $\mathcal{O}(n^3)$.

function FINDMINCOSTSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}, w$)
$\mathcal{G}'^m \leftarrow \mathcal{G}^m_{possibleAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})}$
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 $\mathcal{G}'^m$.
Change the graph to a flow network as described in (Cormen et al. 2001) and return a minimum cutset $\mathbf{Z}$.

Algorithm 6: FindMinCostSep

The correctness without $\mathbf{I}$ follows from the fact that a minimum set is a minimal set and the minimal cut found in the ancestor moral graph is therefore the minimal separating set. The handling of $\mathbf{I}$ is shown in (Acid and De Campos 1996).

ENUMERATING ALL D-SEPARATORS The problem LISTSEP is solved by the general enumeration algorithm given in (van der Zander, Liśkiewicz, and Textor 2014) and using the RCG FINDSEP algorithm in $O(n(n+m))$.

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($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R} \setminus \{V\}$)

Algorithm 7: ListSep

ENUMERATING ALL MINIMAL D-SEPARATORS The problem LISTMINSEP can be solved with algorithm 8 LISTMINSEP with $O(n^3)$ delay between every outputted $\mathbf{Z}$.

function LISTMINSEP($\mathcal{G}, \mathbf{X}, \mathbf{Y}, \mathbf{I}, \mathbf{R}$)
$\mathcal{G}'^m \leftarrow \mathcal{G}^m_{possibleAn(\mathbf{X}\cup\mathbf{Y}\cup\mathbf{I})}$
Add a node X^m connected to all $\mathbf{X}$ nodes.
Add a node Y^m connected to all $\mathbf{Y}$ nodes.
Remove all nodes of $\mathbf{I}$.
Remove all nodes of $\mathbf{V} \setminus \mathbf{R}$, but insert additional edges connecting the neighbours.
of all removed nodes.
Use the algorithm in (Takata 2010) to list all sets separating X^m and Y^m.

Algorithm 8: ListMinSep

The correctness is shown by (Textor and Liśkiewicz 2011) for adjustment sets and generalizes directly to d-separators, because after moralization, both problems are equivalent to enumerating vertex cuts of an undirected graph. The handling of $\mathbf{I}$ is shown by (Acid and De Campos 1996).