def `GetCPDAG` (*Undirected graph* $\mathcal{U}$*, DSep*) **:**
  for *every unshielded* A—C—$B \in \mathcal{U}$ **do**
    if $C \notin DSep(A, B)$ **then**
      Orient $A \rightarrow C \leftarrow B$;
  end
  CPDAG $\mathcal{G} \leftarrow$ ApplyMeekRules($\mathcal{U}$);
  return $\mathcal{G}$;

def `PCTest` (*Undirected graph* $\mathcal{U}(\mathbf{V}, \mathbf{E})$) **:**
  $s \leftarrow 0$;
  while $\exists (A$—$B) \in \mathbf{E}$ *s.t.* $|Ne(A) \setminus \{B\}| \geq s$ **do**
    for $\mathbf{S} \subseteq Ne(A) \setminus \{B\}$ *s.t.* $|\mathbf{S}| = s$ **do**
      if $A \perp\!\!\!\perp B | \mathbf{S}$ **then**
        $\mathcal{U}$.removeEdge(A—B);
        yield $(A \perp\!\!\!\perp B | \mathbf{S})$;
        break;
    end
    $s \leftarrow s + 1$;
  end

Completely connected undirected graph $\mathcal{U}(\mathbf{V}, \mathbf{E})$;
$\forall A, B \in \mathbf{V}$, DSep$(A, B) \leftarrow$ null;
for $(A \perp\!\!\!\perp B | \mathbf{S}) \in PCTest(\mathcal{U})$ **do**
  DSep$(A, B) \leftarrow \mathbf{S}$;
end
CPDAG $\mathcal{G} \leftarrow$ GetCPDAG($\mathcal{U}$, DSep);
Output: $\mathcal{G}$, DSep

Figure 12: The PC algorithm (Spirtes et al., 2000, Sec. 5.4.2).

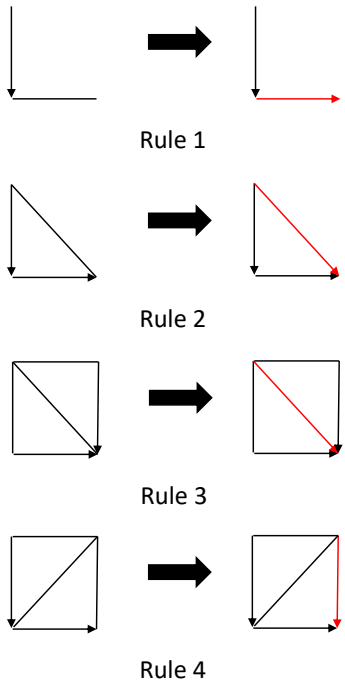


Figure 13: Meek's orientation rules.

Appendix A. Additional details on the PC algorithm

Definition 26 (CPDAG (Maathuis and Colombo, 2015, Pg. 5)) *A set of DAGs that entail the same set of CIs form an MEC. This MEC can be uniquely represented using a CPDAG. A CPDAG is a graph with the same skeleton as each DAG in the MEC and contains both directed (→) and undirected (—) edges. A directed edge $A \rightarrow B$ means that the $A \rightarrow B$ is present in every DAG in the MEC. An undirected edge A—B means that there is at least one DAG in the MEC with an $A \rightarrow B$ edge and at least one DAG with the $B \rightarrow A$ edge.*

The PC algorithm (Fig. 12) starts with a fully connected skeleton and runs CI tests to remove edges. For each pair of nodes (A, B) that are adjacent in the skeleton, we run CIs of size s—starting with $s = 0$ and then increasing it by one in each subsequent iteration—until the edge is removed or the number of nodes adjacent to both A and B is less than s. Once the skeleton is found, UCs are detected and then additional edges are oriented by repeatedly applying Meek's rules (Fig. 13) until

Input: Node X.
MB$(X) \leftarrow \emptyset$;
// Forward Pass.
while *MB*(X) *has changed* **do**
  for $V \in \mathbf{V} \setminus (MB(X) \cup \{X\})$ **do**
    if $V \not\perp\!\!\!\perp X | MB(X)$ **then** MB(X).add(V) ;
  end
end
// Backward Pass.
for $V \in MB(X)$ **do**
  if $V \perp\!\!\!\perp X | (MB(X) \setminus \{X\})$ **then** MB(X).remove(V) ;
end
Output: MB(X)

Figure 14: The IAMB algorithm.

def Nbrs(*CPDAG* $\mathcal{G}$, *Target* X)**:**
  parents $\leftarrow \emptyset$;
  children $\leftarrow \emptyset$;
  unoriented $\leftarrow \emptyset$;
  for $V \in Ne_{\mathcal{G}}(X)$ **do**
    if $X \leftarrow V \in \mathcal{G}$ **then**
      parents.add(V);
    else if $X \rightarrow V \in \mathcal{G}$ **then**
      children.add(V);
    else
      unoriented.add(V);
  end
  return parents, children, unoriented;

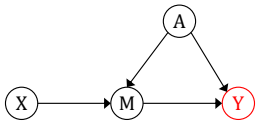
Figure 15: The *Nbrs* subroutine used by SD.

no additional edges can be oriented. Under faithfulness, with access to a CI oracle, the output of PC is a CPDAG encoding the MEC of the true DAG $\mathcal{G}^*$.

Appendix B. Additional details for Section 4

In the example below, we demonstrate that there exist causal graphs with nodes for which mns does not exist.

Example 6 (MNS does not exist) *In the following graph, for node $Y \in Desc(X)$, mns_X does not exist because there is no subset of $Ne(X)$ that d-separates Y from X.*



B.1. Omitted Proofs for Section 4

Proposition 2 *For any node $V \notin (Desc(X) \cup Ne^+(X))$, $mns_X(V)$ exists and $mns_X(V) \subseteq Pa(X)$.*

Proof Let $\mathbf{Q} = \text{Desc}(X) \cup \text{Ne}^+(X)$. Since Pa$(X)$ blocks all backdoor paths from X, for every $V \notin \mathbf{Q}$, we have $V \perp\!\!\!\perp X | \text{Pa}(X)$. Therefore, for every $V \notin \mathbf{Q}$, there exists some subset $\mathbf{S} \subseteq \text{Pa}(X)$ such that $V \perp\!\!\!\perp X | \mathbf{S}$. ■

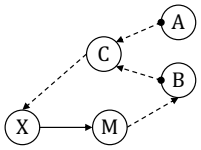
Proposition 3 *[Uniqueness of MNS] For nodes V s.t. $mns_X(V)$ exists, it is unique.*

Proof We will prove this by contradiction. Consider a node V with two MNSs: $\mathbf{S}_1 \subseteq \text{Ne}(X)$ and $\mathbf{S}_2 \subseteq \text{Ne}(X)$ with $\mathbf{S}_1 \neq \mathbf{S}_2$. If $\mathbf{S}_1 \subset \mathbf{S}_2$ or $\mathbf{S}_2 \subset \mathbf{S}_1$, then minimality is violated. Hence, going

forward we will only consider the case where $\mathbf{S}_1 \setminus \mathbf{S}_2 \neq \emptyset$ and $\mathbf{S}_2 \setminus \mathbf{S}_1 \neq \emptyset$. Consider any node $A \in \mathbf{S}_1 \setminus \mathbf{S}_2$. For $\mathbf{S}_2$ to be a valid MNS, some nodes in $\mathbf{S}_2 \setminus \mathbf{S}_1$ must block all paths from V to X that contain A (this is because if this path were to be only be blocked by some nodes in $\mathbf{S}_1$, then minimality of $\mathbf{S}_1$ will be violated as $\mathbf{S}_1 \setminus \{A\}$ would also have been a valid MNS). This means that there is a path from V to X through some nodes in $\mathbf{S}_2 \setminus \mathbf{S}_1$ that cannot be blocked by $\mathbf{S}_1$ (else these nodes in $\mathbf{S}_1$ would have blocked the paths from V to A violating minimality of $\mathbf{S}_1$). This contradicts the fact that $\mathbf{S}_1$ is a valid MNS. Therefore, we must have $\mathbf{S}_1 = \mathbf{S}_2$. ■

Proposition 4 *[Eager Collider Check] For nodes $A, B \in \mathbf{V} \setminus Ne^+(X)$, any $\mathbf{S} \subset \mathbf{V} \setminus \{A, B, X\}$, if (i) $A \perp\!\!\!\perp B|\mathbf{S}$; and (ii) $A \not\perp\!\!\!\perp B|\mathbf{S} \cup \{X\}$; then $A, B \notin Desc(X)$ and $mns_X(A), mns_X(B) \subseteq Pa(X)$.*

Proof We prove this by contradiction. Let's say there is a child M of X such that $M \in \text{mns}_X(B)$ or $M \in \text{mns}_X(A)$. First, note that if Conditions (i, ii) hold, then there is a path of the form $A\bullet \rightarrow C$ and $B\bullet \rightarrow C$ and $C \rightarrow \ldots X$, where $\bullet$ means that there can be either an arrowhead or tail (i.e., there can be either a directed path $A \rightarrow \ldots \rightarrow C$ or a backdoor path $A \leftarrow \ldots \rightarrow C$ and likewise for B) with $C \notin \mathbf{S}$. W.l.o.g., let's say that $M \in \text{mns}_X(B)$ (the argument for node A follows similarly). Then there is a directed path from X to B through M (i.e., $X \rightarrow M \rightarrow \ldots \rightarrow B$). There cannot be a path $B \rightarrow \ldots \rightarrow M$ because then M will be a collider and therefore we will have $M \notin \text{mns}_X(B)$. These components are illustrated in the figure below:



We now show that $B \notin \text{Desc}(X)$ (the argument for node A is the same). There cannot be a directed path $B \rightarrow \ldots \rightarrow C$ because otherwise a cycle $B \rightarrow \ldots \rightarrow C \rightarrow \ldots \rightarrow X \rightarrow M \rightarrow \ldots \rightarrow B$ gets created. Thus the path from B to C must be of the form $B \leftarrow \ldots \rightarrow C$. Note that there is an active path between A and B through X ($A\bullet \rightarrow \ldots \rightarrow C \rightarrow \ldots \rightarrow X \rightarrow M \rightarrow \ldots \rightarrow B$). Since $A \perp\!\!\!\perp B|\mathbf{S}$, there are two possibilities: (i) $\mathbf{S}$ contains X to block this path which contradicts the definition of $\mathbf{S}$ (where $X \notin \mathbf{S}$); or (ii) $\mathbf{S}$ blocks all paths between A and X or between B and X in which case A and B cannot become dependent when additionally conditioned on X thereby violating Condition (ii). Therefore, we have that $A, B \notin \text{Desc}(X)$ and by Prop. 2, $\text{mns}_X(A)$ and $\text{mns}_X(B)$ will be valid and only contain parents of X. ■

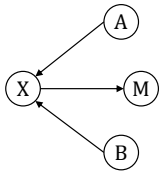
Claassen and Heskes (2012) use minimal (in)dependencies to construct three logical rules which are sound and complete for performing causal discovery. While their algorithm cannot directly be used for local causal discovery, we show below that Lemma 3 in their paper can be used to simplify the proof of Eager Collider Check:

Proof [Alternative proof of ECC] Since we have $A \perp\!\!\!\perp B|\mathbf{S}$ and $A \not\perp\!\!\!\perp B|\mathbf{S} \cup \{X\}$, by Claassen and Heskes (2012, Lemma 3), we have $A, B \notin \text{Desc}(X)$. Therefore, by Prop. 2, $\text{mns}_X(A)$ and $\text{mns}_X(B)$ will be valid and only contain parents of X. ■

B.1.1. PROOF OF CORRECTNESS OF LDECC UNDER THE CFA

Lemma 27 *Consider a DAG $\mathcal{G}(\mathbf{V}, \mathbf{E})$ and a node $X \in \mathbf{V}$. Let $A, B \in Pa(X)$ be two parents of X such that A—$B \notin \mathbf{E}$. Then, for any $\mathbf{S} \subseteq \mathbf{V} \setminus \{A, B, X\}$ such that $A \perp\!\!\!\perp B|\mathbf{S}$, we have $Ch(X) \cap \mathbf{S} = \emptyset$.*

Proof We prove this by contradiction. Let's say that there is child $M \in \mathbf{S}$ (the relevant component of the graph is shown in the figure below).



Since M is a child of X, conditioning on M opens up the path $A \to X \leftarrow B$ rendering A and B dependent conditioned on $\mathbf{S}$. Therefore, we must have $M \notin \mathbf{S}$. ■

Theorem 5 *[Correctness] Under the CFA and with access to a CI oracle, we have $\Theta_{LDECC} \overset{set}{=} \Theta^*$.*

Proof We will prove the correctness of LDECC by showing that (i) every orientable neighbor of the treatment X will get oriented correctly by LDECC; and (ii) every unorientable neighbor of the treatment will remain unoriented.

We assume that the function *FindMarkovBlanket* finds the Markov blanket correctly under the CFA. The IAMB algorithm, which we use in our experiments, has this property. Additionally, the function *RunLocalPC* will also return the correct $\text{Ne}(X)$ under the CFA and with a CI oracle.

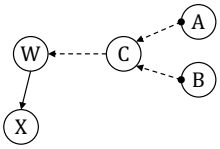
Parents are oriented correctly. In PC, edges get oriented using UCs and then additional orientations are propagated via the application of Meek's rules (Figure 13).

The simplest case is where two parents form a UC at X. Consider parents W_1 and W_2 that get oriented because they form a UC $W_1 \to X \leftarrow W_2$. Lines 7,8 will mark W_1 and W_2 as parents.

We will now consider parents that get oriented due to each of the four Meek rules and show that LDECC orients parents for each of the four cases.

Meek Rule 1*:*

Consider a parent W that gets oriented due to the application of Meek's rule 1. This can only happen due to some UC $A \to C \leftarrow B$ from which these orientations have been propagated (relevant components of the graph are illustrated in the figure below).

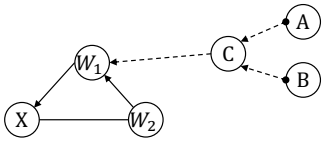


Thus there is a directed path $C \to \ldots \to W \to X$. This would mean that $W \in \text{mns}_X(A)$ and $W \in \text{mns}_X(B)$. Thus Line 16 will mark W as a parent.

Meek Rule 2*:*

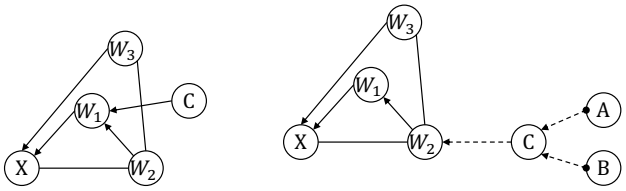
Consider a parent W_2 that gets oriented due to the application of Meek's rule 2. In this case, we have an oriented path $W_2 \to W_1 \to X$ but the edge W_2—X is unoriented (and Meek Rule 2 must be applied to orient it).

The first possibility is that the $X \leftarrow W_1$ was oriented due to some UC $A \rightarrow C \leftarrow B$ with a path $C \rightarrow \ldots \rightarrow W_1$ (relevant components of the graph are illustrated in the figure below):



In this case, there would be a collider at W_1: $C \rightarrow \ldots \rightarrow W_1 \leftarrow W_2$. Thus if $W_1 \in \text{mns}_X(A)$, then $W_2 \in \text{mns}_X(A)$ and thus LDECC will mark W_2 as a parent in Line 16.

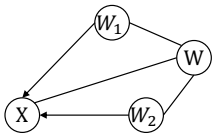
The other possibility is that $X \leftarrow W_1$ was oriented due to a UC like $W_3 \rightarrow X \leftarrow W_1$ but there is an edge W_3—W_2 which causes the collider $W_2 \rightarrow X \leftarrow W_3$ to be shielded and due to this, the W_2—X remained unoriented. However, by definition of the Meek rule, the edge $W_2 \rightarrow W_1$ is oriented. Thus, (i) either there is a UC of the form $W_2 \rightarrow W_1 \leftarrow C$; or (ii) there is a UC from which the $W_2 \rightarrow W_1$ orientation was propagated. The relevant components of the graph for these two cases are illustrated in the figures below.



For Case (i), $W_2 \in \text{mns}_X(C)$ and for Case (ii), $W_2 \in \text{mns}_X(A)$ and $W_2 \in \text{mns}_X(B)$. In both cases, LDECC will mark W_2 as a parent in Lines 16.

Meek Rule 3*:*

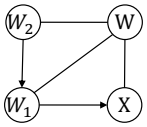
Consider a parent W that gets oriented due to the application of Meek's rule 3 (relevant component of the graph is shown in the figure below).



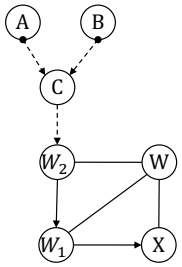
By definition of the Meek rule, W_1—W—W_2 is a non-collider (because if it were a collider, the edges would have been oriented since this triple is unshielded) and therefore for any $\mathbf{S} \subseteq \mathbf{V} \setminus \{W_1, W_2\}$ such that $W_1 \perp\!\!\!\perp W_2 | \mathbf{S}$, we have $W \in \mathbf{S}$. Thus Line 9 will mark W as a parent.

Meek Rule 4*:*

Consider a parent W that gets oriented due to the application of Meek's rule 4. The relevant component of the graph is shown in the figure below.

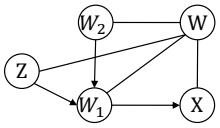


The first possibility is that the orientations $W_2 \rightarrow W_1 \rightarrow X$ were propagated from a UC $A \rightarrow C \leftarrow B$ with a path $C \rightarrow \ldots \rightarrow W_2$ (the relevant components of the graph are shown in the figure below).



In this case, due to the non-collider W_2—W—X (because if it were a collider, the edges would have been oriented since this triple is unshielded), we have $W \in \text{mns}_X(A)$ and thus W will be marked as a parent in Line 16.

The second possibility (similar to the Meek rule 2 case) is that $W_2 \to W_1$ was oriented due to a UC like $Z \to W_1 \leftarrow W_2$ but there is an edge Z—W which shields the $Z \to W_1$—W causing the W_1—W edge to remain unoriented (the relevant components of the graph are shown in the figure below).



In this case, we would have $W \in \text{mns}_X(Z)$ and thus W gets marked as a parent in Line 16.

Children are oriented correctly. We now similarly show that children of X get oriented correctly.

The simplest case is when there is a UC of the form $X \to M \leftarrow V$. Since $\text{MB}(X)$ and $\text{Ne}(X)$ are correct, the function *GetUCChildren* (Fig. 4) will mark M as a child.

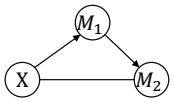
Now, we consider each Meek rule one at a time and show that LDECC will orient children for each rule.

Meek Rule 1*:*

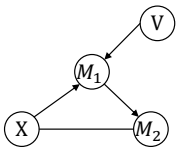
Consider a child M that gets oriented due to the application of Meek's rule 1. This can only happen if there is some parent W that gets oriented and W—X—M forms an unshielded non-collider. In this case, Line 19 will mark M as a child.

Meek Rule 2*:*

Consider a child M_2 that gets oriented due to the application of Meek's rule 2: there is an oriented path $X \to M_1 \to M_2$ but the X—M_2 edge is still unoriented (the relevant component of the graph is shown in the figure below).



One possibility is that there is UC of the form $V \to M_1 \to X$ which orients the $X \to M_1$ edge and V—M_1—M_2 is a non-collider which orients the $M_1 \to M_2$ edge (the relevant components of the graph are shown in the figure below).



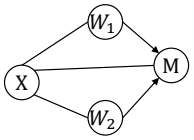
In this case, since V is a spouse (i.e., parent of child) of X, the function *GetUCChildren* (Fig. 4) will mark M_2 as a child.

Note that if the $X \to M_1$ was oriented due to a UC upstream of X via the application of Meek rule 1, this would also cause the $X \to M_2$ edge to be oriented (and thus Meek rule 2 would not apply).

The other possibility is that there might be a UC of the form $M_1 \to M_2 \leftarrow Z$ that can orient $M_1 \to M_2$. However, for the $X \to M_1$ edge to remain unoriented, there must be an edge Z—X to shield the X—M_2—Z collider. If this happens, Meek rule 3 would apply (which we handle separately as shown next).

Meek Rule 3*:*

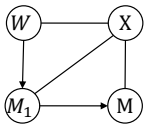
Consider a child M that gets oriented due to the application of Meek's rule 3 (the relevant component of the graph is shown in the figure below).



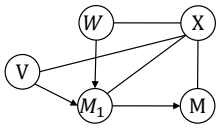
By definition of the Meek rule, W_1—X—W_2 is a non-collider (because if it were a collider, the edges would have been oriented since this triple is unshielded) and since $W_1 \to M \leftarrow W_2$ forms a collider, we have $W_1 \not\perp\!\!\!\perp W_2 | \mathbf{S} \cup \{M\}$ for any $\mathbf{S}$ s.t. $W_1 \perp\!\!\!\perp W_2 | \mathbf{S}$. Thus Line 13 will mark M as a child.

Meek Rule 4*:*

Consider a child M that gets oriented due to the application of Meek's rule 4 (the relevant component of the graph is shown in the figure below).

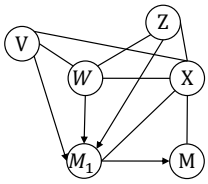


One possibility such that the $W \to M_1$ gets oriented leaving the edges W—X, X—M, and X—M_1 unoriented is if there is a UC of the form $V \to M_1 \leftarrow W$ where there is an edge V—X to shield the X—M_1 edge (the relevant component of the graph is shown in the figure below).



Here the triple W—X—V must be a non-collider to keep the X—M edge unoriented (otherwise applying Meek rule 1 from the UC $W \to X \leftarrow V$ would orient $X \to M$). So for any $\mathbf{S}$ such that $V \perp\!\!\!\perp W | \mathbf{S}$, we must have $X \in \mathbf{S}$ and $V \not\perp\!\!\!\perp W | \mathbf{S} \cup \{M\}$. Therefore, Line 13 will mark M as a child.

The other possibility is that the $W \to M_1$ gets oriented due to Meek rule 3 (the relevant component of the graph is shown in the figure below).



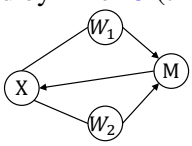
Here the Z—X and V—X must be present to keep the X—M_1 edge unoriented (because otherwise an unshielded collider would be created). Furthermore, the triple Z—X—V must be a non-collider in order to keep the X—M edge unoriented (otherwise applying Meek rule 1 from the UC $Z \to X \leftarrow V$ would orient $X \to M$). So for any $\mathbf{S}$ such that $V \perp\!\!\!\perp Z|\mathbf{S}$, we must have $X \in \mathbf{S}$ and $V \not\perp\!\!\!\perp Z|\mathbf{S} \cup \{M\}$. Therefore, Line 13 will mark M as a child.

No spurious orientations. Now we prove that nodes are never oriented the wrong way by LDECC.

We show that *GetUCChildren* (Fig. 4) will never orient a parent as a child. We prove this by contradiction. Let's say there is a parent W that is oriented as a child by *GetUCChildren*. This will happen if there is a node $D \in \text{MB}(X) \setminus \text{Ne}(X)$ s.t. $D \not\perp\!\!\!\perp W|\text{DSep}(D, X)$ with $W \notin \text{DSep}(D, X)$. Thus, there would a path D—W. Since W is a parent, there would be a path D—$W \to X$ leading to $W \in \text{DSep}(D, X)$ which is a contradiction.

Line 8 can never mark a child as a parent since otherwise the CFA would be violated.

Line 13 will not mark a parent as a child. Consider a parent M that incorrectly gets marked as a child by Line 13 (the relevant component of the graph is shown in the figure below).



For Line 13 to be reached, the if-condition in Line 10 must be *True*. This will happen if $W1$—X—$W2$ is a non-collider. Thus at least one of W_1 or W_2 is a child. W.l.o.g., let's assume that W_1 is a child. If that happens, a cycle gets created: $X \to W_1 \to M \to X$. Therefore M can never be oriented by Line 13.

Line 16 cannot mark a child as a parent because of the correctness of the ECC (Prop. 4).

Line 9 will not mark a child as a parent. Both A and B from Line 7 are parents of X. By Lemma 27, the set $\mathbf{S}$ in Line 9 cannot contain a child.

Line 19 will not add a child as a parent because otherwise the CFA would be violated. ■

B.2. Omitted proofs for computational requirements (Sec. 4.1)

Proposition 6 *[PC vs LDECC] We have* $T_{LDECC} \leq T_{PC} + \mathcal{O}(|\mathbf{V}|^2) + \mathcal{O}\left(|\mathbf{V}| \cdot 2^{|Ne(X)|}\right)$.

Proof LDECC performs $\mathcal{O}\left(|\mathbf{V}| \cdot 2^{|\text{Ne}(X)|}\right)$ CI tests to find $\text{Ne}(X)$. After that LDECC runs CI tests like PC. The *GetMNS* function requires $\mathcal{O}(2^{|\text{Ne}(X)|})$ CI tests. The $\mathcal{O}(|\mathbf{V}|^2)$ term accounts for the extra CI tests of the form $A \not\perp\!\!\!\perp B|\mathbf{S} \cup \{X\}$ we might run for ECCs. ■

Proposition 11 *[LDECC upper bound] For a locally orientable DAG* $\mathcal{G}^*$*, let* $D = \max_{V \in MB^+(X)} |Ne(V)|$ *and* $S = \max_{P \in Pa(X)} \min_{\alpha \in POC(P)} sep(\alpha)$*. Then* $T_{LDECC} \leq \mathcal{O}(|\mathbf{V}|^{\max\{S,D\}})$.

Proof Since the graph is locally orientable, all neighbors of X will get oriented. The complexity of discovering the neighbors of X is upper bounded by $\mathcal{O}(|\mathbf{V}|^{|\text{MB}^+(X)|})$; that of discovering any non-colliders of the form A—X—B is $\mathcal{O}(|\mathbf{V}|^D)$; and in order to unshield the colliders that orient the parents, LDECC runs $\mathcal{O}(|\mathbf{V}|^S)$ tests. Thus the total number of CI tests is $\mathcal{O}(|\mathbf{V}|^{\max\{S,D\}})$. ■

Proposition 12 *[SD upper bound] For a locally orientable DAG $\mathcal{G}^*$, let $\pi : \mathbf{V} \to \mathbb{N}$ be the order in which nodes are processed by SD (Line 4 of SD). For $P \in Pa(X)$, let $CUC(P) = \text{argmin}_{\alpha \in POC(P)} \pi(\alpha)$ denote the closest UC to P. Let $C = \max_{P \in Pa(X)} sep(CUC(P))$, $D = \max_{V \in MB^+(X)} |Ne(V)|$, and $E = \max_{\{V : \pi(V) < \pi(CUC(P))\}} |Ne(V)|$. Then $T_{SD} \leq \mathcal{O}(|\mathbf{V}|^{\max\{C,D,E\}})$.*

Proof Like LDECC, the complexity of discovering the neighbors of X and non-colliders of the form A—X—B is at most $\mathcal{O}(|\mathbf{V}|^D)$ tests. Then, in order to sequentially reach the closest UCs, SD runs $\mathcal{O}(|\mathbf{V}|^E)$ tests. Once the collider is reached, SD runs $\mathcal{O}(|\mathbf{V}|^C)$ CI tests to unshield them. ■

B.3. Omitted proofs for faithfulness requirements (Sec. 4.2)

Proposition 17 *PC will identify the MEC of $\mathcal{G}^*$ if LF holds for all nodes.*

Proof It is known that the PC algorithm correctly identifies the MEC of $\mathcal{G}^*$ if AF and OF hold for all nodes (see e.g., Zhang and Spirtes (2008)). AF for all nodes ensures that the skeleton is recovered correctly. OF for all unshielded triples ensures that UCs are detected correctly and that the orientations propagated via Meek's rules are correct. ■

Proposition 18 *[Faithfulness for PC and SD] PC and SD will identify Θ^* if (i) LF holds $\forall V \in MB^+(X)$; (ii) $\forall (A \to C \leftarrow B) \in J^*$, (a) LF holds for A, B, and C, and (b) LF holds for each node on all paths $C \to \ldots \to V \in \mathcal{G}^*$ s.t. $V \in Ne(X)$; (iii) For every edge A—$B \notin \mathcal{G}^*$, $\exists \mathbf{S} \subseteq (Ne_{\mathcal{U}}(A) \cup Ne_{\mathcal{U}}(B))$ s.t. $A \perp\!\!\!\perp B | \mathbf{S}$; and (iv) OF holds for all unshielded triples in $\mathcal{G}^*$.*

Proof Similar to the proof of Thm. 5, we will prove this by showing that all neighbors of X get oriented correctly and the unorientable neighbors remain unoriented.

The key ideas of the proof are as follows: (1) Condition (i) guarantees that the structure inside $\text{MB}^+(X)$ is discovered correctly which further ensures that Meek rules 2-4 work correctly (since, as shown in the proof of Thm. 5, they are only applied inside $\text{MB}^+(X)$); (2) Condition (ii)(a) guarantees that each UC in $\mathcal{G}^*$ is detected and unshielded; (3) Condition (ii)(b) guarantees that orientations from each UC in $\mathcal{G}^*$ are propagated correctly to X; (4) Condition (iii) guarantees that the undirected skeleton discovered by PC and SD is a subgraph of the skeleton of $\mathcal{G}^*$: This is because PC and SD remove an edge A—B by running CI tests by conditioning on neighbors of A and B in the current undirected skeleton; and (5) Condition (iv) guarantees that in the skeleton recovered by PC and SD, there are no incorrectly detected UCs.

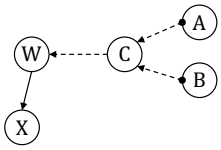
Parents are oriented correctly. In PC and SD, edges get oriented using UCs and then additional orientations are propagated via the application of Meek's rules (Figure 13).

The simplest case is where two parents form a UC at X. Consider parents W_1 and W_2 that get oriented because they form a UC $W_1 \to X \leftarrow W_2$. Condition (i) ensures they are marked as parents.

We will now consider parents that get oriented due to each of the four Meek rules and show that SD and PC orient parents for each of the four cases.

Meek Rule 1*:*

Consider a parent W that gets oriented due to the application of Meek's rule 1. This can only happen due to some UC $A \to C \leftarrow B$ from which these orientations have been propagated (relevant components of the graph are illustrated in the figure below).



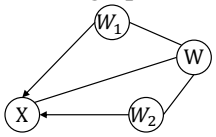
Thus there is a directed path $C \to \ldots \to W \to X$. By Condition (ii)(a), the UC $A \to C \leftarrow B$ will get detected correctly, and by Condition (ii)(b), the orientations will be propagated correctly along this path.

Meek Rule 2*:*

Consider a parent W_2 that gets oriented due to the application of Meek's rule 2. In this case, we have an oriented path $W_2 \to W_1 \to X$ but the edge W_2—X is unoriented (and Meek Rule 2 must be applied to orient it). Condition (i) ensures that this relevant component of the graph is discovered correctly.

Meek Rule 3*:*

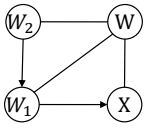
Consider a parent W that gets oriented due to the application of Meek's rule 3 (relevant component of the graph is shown in the figure below).



Condition (i) ensures that this relevant component of the graph is discovered correctly.

Meek Rule 4*:*

Consider a parent W that gets oriented due to the application of Meek's rule 4. The relevant component of the graph is shown in the figure below.



Condition (i) ensures that this relevant component of the graph is discovered correctly.

Children are oriented correctly. We now similarly show that children of X get oriented correctly.

The simplest case is when there is a UC of the form $X \to M \leftarrow V$. Condition (i) ensures that this UC is discovered correctly.

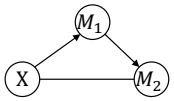
Now, we consider each Meek rule one at a time and show that SD and PC will orient children for each rule.

Meek Rule 1*:*

Consider a child M that gets oriented due to the application of Meek's rule 1. This can only happen if there is some parent W that gets oriented and W—X—M forms an unshielded non-collider. Condition (i) ensures that this unshielded non-collider is discovered correctly.

Meek Rule 2*:*

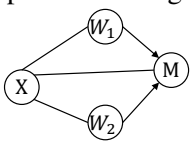
Consider a child M_2 that gets oriented due to the application of Meek's rule 2: there is an oriented path $X \to M_1 \to M_2$ but the X—M_2 edge is still unoriented (the relevant component of the graph is shown in the figure below).



Condition (i) ensures that this relevant component of the graph is discovered correctly.

Meek Rule 3*:*

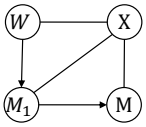
Consider a child M that gets oriented due to the application of Meek's rule 3 (the relevant component of the graph is shown in the figure below).



Condition (i) ensures that this relevant component of the graph is discovered correctly.

Meek Rule 4*:*

Consider a child M that gets oriented due to the application of Meek's rule 4 (the relevant component of the graph is shown in the figure below).



Condition (i) ensures that this relevant component of the graph is discovered correctly.

No spurious orientations. As argued in the preamble of the proof, Conditions (iii, iv) ensure that there are no incorrectly detected UCs. Condition (ii) ensures that no incorrect orientations are propagated from the detected UCs. ■

Proposition 20 *[Faithfulness for LDECC] LDECC will identify Θ^* if (i) LF holds $\forall V \in MB^+(X)$; (ii) $H \subseteq H^*$; (iii) $\forall (A, B, \mathbf{S}) \in H$, MFF holds for $\{A, B\} \setminus Ne(X)$; and (iv) $\forall (A, B, \mathbf{S}) \in H^*$ s.t. there is a UC $(A \to C \leftarrow B) \in \mathcal{G}^*$, we have (a) AF holds for A and B; and (b) $(A, B, \mathbf{S}) \in H$.*

Proof The proof is very similar to that of Theorem 5. The high-level idea is as follows. For any nodes in $\text{Ne}(X)$ that are oriented *without* using ECCs, Condition (i) will ensure they get oriented correctly as they only depend on the structure inside $\text{MB}(X)$. For nodes that get oriented via ECCs, Condition (iv)(a) ensures that each UC eventually gets unshielded. For any UC $A \to C \leftarrow B$ in Condition (iv), since AF holds for both A and B, $\text{Ne}(A)$ and $\text{Ne}(B)$ are detected correctly. Since $\exists \mathbf{S} \subseteq (\text{Ne}(A) \cup \text{Ne}(B))$ s.t. $A \perp\!\!\!\perp B | \mathbf{S}$, we will eventually remove the A—B thereby unshielding this collider. Condition (iv)(b) ensures that we run an ECC for this UC, i.e., the if-block in Line 15 is entered. Next, by Condition (iii), since MFF holds for A and B, the *GetMNS* function will correctly return the parents of X that this UC orients.

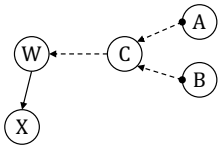
We assume that the function *FindMarkovBlanket* identifies the Markov blanket correctly under Condition (i). The IAMB algorithm, which we use in our experiments, has this property. Additionally, the function *RunLocalPC* will also identify the correct $\text{Ne}(X)$ under Condition (i).

Parents are oriented correctly. The simplest case is where two parents form a UC at X. Consider parents W_1 and W_2 that get oriented because they form a UC $W_1 \to X \leftarrow W_2$. By Condition (i), Lines 7,8 will mark W_1 and W_2 as parents.

We will now consider parents that get oriented due to each of the four Meek rules and show that LDECC orients parents for each of the four cases.

Meek Rule 1*:*

Consider a parent W that gets oriented due to the application of Meek's rule 1. This can only happen due to some UC $A \to C \leftarrow B$ from which these orientations have been propagated (relevant components of the graph are illustrated in the figure below).

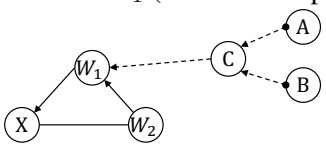


As explained in the preamble, Conditions (iii, iv) ensure that ECCs mark parent correctly and thus Line 16 will mark W as a parent.

Meek Rule 2*:*

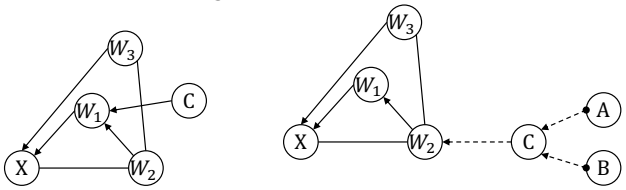
Consider a parent W_2 that gets oriented due to the application of Meek's rule 2. In this case, we have an oriented path $W_2 \to W_1 \to X$ but the edge W_2—X is unoriented (and Meek Rule 2 must be applied to orient it).

The first possibility is that the $X \leftarrow W_1$ was oriented due to some UC $A \to C \leftarrow B$ with a path $C \to \ldots \to W_1$ (relevant components of the graph are illustrated in the figure below):



By Conditions (iii, iv), LDECC will mark W_2 as a parent in Line 16.

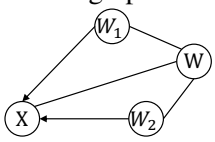
The other possibility is that $X \leftarrow W_1$ was oriented due to a UC like $W_3 \to X \leftarrow W_1$ but there is an edge W_3—W_2 which causes the collider $W_2 \to X \leftarrow W_3$ to be shielded and due to this, the W_2—X remained unoriented. However, by definition of the Meek rule, the edge $W_2 \to W_1$ is oriented. Thus, (i) either there is a UC of the form $W_2 \to W_1 \leftarrow C$; or (ii) there is a UC from which the $W_2 \to W_1$ orientation was propagated. The relevant components of the graph for these two cases are illustrated in the figures below.



In both cases, by Conditions (iii, iv), LDECC will mark W_2 as a parent in Lines 16.

Meek Rule 3*:*

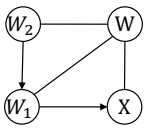
Consider a parent W that gets oriented due to the application of Meek's rule 3 (relevant component of the graph is shown in the figure below).



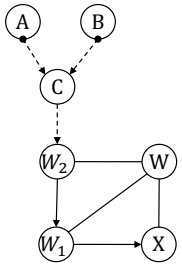
By definition of the Meek rule, W_1—W—W_2 is a non-collider (because if it were a collider, the edges would have been oriented since this triple is unshielded) and therefore for any $\mathbf{S} \subseteq \mathbf{V} \setminus \{W_1, W_2\}$ such that $W_1 \perp\!\!\!\perp W_2 | \mathbf{S}$, by Condition (i), we have $W \in \mathbf{S}$. Thus Line 9 will mark W as a parent.

Meek Rule 4*:*

Consider a parent W that gets oriented due to the application of Meek's rule 4. The relevant component of the graph is shown in the figure below.

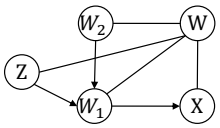


The first possibility is that the orientations $W_2 \to W_1 \to X$ were propagated from a UC $A \to C \leftarrow B$ with a path $C \to \ldots \to W_2$ (the relevant components of the graph are shown in the figure below).



In this case, due to the non-collider W_2—W—X (because if it were a collider, the edges would have been oriented since this triple is unshielded), we have $W \in \text{mns}_X(A)$ and thus by Conditions (iii, iv), W will be marked as a parent in Line 16.

The second possibility (similar to the Meek rule 2 case) is that $W_2 \to W_1$ was oriented due to a UC like $Z \to W_1 \leftarrow W_2$ but there is an edge Z—W which shields the $Z \to W_1$—W causing the W_1—W edge to remain unoriented (the relevant components of the graph are shown in the figure below).



In this case, we would have $W \in \text{mns}_X(Z)$ and thus by Conditions (iii, iv), W gets marked as a parent in Line 16.

Children are oriented correctly. We now similarly show that children of X get oriented correctly.

The simplest case is when there is a UC of the form $X \to M \leftarrow V$. By Condition (i), since $\text{MB}(X)$ and $\text{Ne}(X)$ are correct, the function *GetUCChildren* (Fig. 4) will mark M as a child.

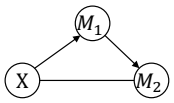
Now, we consider each Meek rule one at a time and show that LDECC will orient children for each rule.

Meek Rule 1*:*

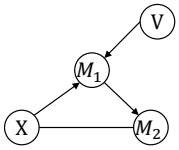
Consider a child M that gets oriented due to the application of Meek's rule 1. This can only happen if there is some parent W that gets oriented and W—X—M forms an unshielded non-collider. By Condition (i), Line 19 will mark M as a child.

Meek Rule 2*:*

Consider a child M_2 that gets oriented due to the application of Meek's rule 2: there is an oriented path $X \to M_1 \to M_2$ but the X—M_2 edge is still unoriented (the relevant component of the graph is shown in the figure below).



One possibility is that there is UC of the form $V \to M_1 \to X$ which orients the $X \to M_1$ edge and V—M_1—M_2 is a non-collider which orients the $M_1 \to M_2$ edge (the relevant components of the graph are shown in the figure below).



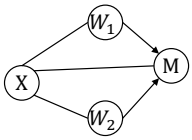
In this case, since V is a spouse (i.e., parent of child) of X, by Condition (i), the function *GetUCChildren* (Fig. 4) will mark M_2 as a child.

Note that if the $X \to M_1$ was oriented due to a UC upstream of X via the application of Meek rule 1, this would also cause the $X \to M_2$ edge to be oriented (and thus Meek rule 2 would not apply).

The other possibility is that there might be a UC of the form $M_1 \to M_2 \leftarrow Z$ that can orient $M_1 \to M_2$. However, for the $X \to M_1$ edge to remain unoriented, there must be an edge Z—X to shield the X—M_2—Z collider. If this happens, Meek rule 3 would apply (which we handle separately as shown next).

Meek Rule 3*:*

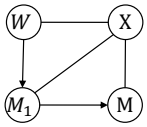
Consider a child M that gets oriented due to the application of Meek's rule 3 (the relevant component of the graph is shown in the figure below).



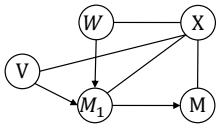
By definition of the Meek rule, W_1—X—W_2 is a non-collider (because if it were a collider, the edges would have been oriented since this triple is unshielded) and since $W_1 \to M \leftarrow W_2$ forms a collider, we have $W_1 \not\perp\!\!\!\perp W_2 | \mathbf{S} \cup \{M\}$ for any $\mathbf{S}$ s.t. $W_1 \perp\!\!\!\perp W_2 | \mathbf{S}$. Thus, by Condition (i), Line 13 will mark M as a child.

Meek Rule 4*:*

Consider a child M that gets oriented due to the application of Meek's rule 4 (the relevant component of the graph is shown in the figure below).

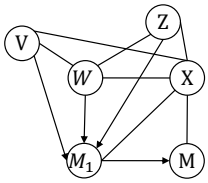


One possibility such that the $W \to M_1$ gets oriented leaving the edges W—X, X—M, and X—M_1 unoriented is if there is a UC of the form $V \to M_1 \leftarrow W$ where there is an edge V—X to shield the X—M_1 edge (the relevant component of the graph is shown in the figure below).



Here the triple W—X—V must be a non-collider to keep the X—M edge unoriented (otherwise applying Meek rule 1 from the UC $W \to X \leftarrow V$ would orient $X \to M$). So for any $\mathbf{S}$ such that $V \perp\!\!\!\perp W|\mathbf{S}$, by Condition (i), we must have $X \in \mathbf{S}$ and $V \not\perp\!\!\!\perp W|\mathbf{S} \cup \{M\}$. Therefore, Line 13 will mark M as a child.

The other possibility is that the $W \to M_1$ gets oriented due to Meek rule 3 (the relevant component of the graph is shown in the figure below).



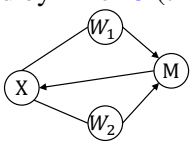
Here the Z—X and V—X must be present to keep the X—M_1 edge unoriented (because otherwise an unshielded collider would be created). Furthermore, the triple Z—X—V must be a non-collider in order to keep the X—M edge unoriented (otherwise applying Meek rule 1 from the UC $Z \to X \leftarrow V$ would orient $X \to M$). So for any $\mathbf{S}$ such that $V \perp\!\!\!\perp Z|\mathbf{S}$, by Condition (i), we must have $X \in \mathbf{S}$ and $V \not\perp\!\!\!\perp Z|\mathbf{S} \cup \{M\}$. Therefore, Line 13 will mark M as a child.

No spurious orientations. Now we prove that nodes are never oriented the wrong way by LDECC.

By Condition (i), *GetUCChildren* (Fig. 4) will never orient a parent as a child.

Line 8 can never mark a child as a parent since otherwise Condition (i) would be violated.

Line 13 will not mark a parent as a child. Consider a parent M that incorrectly gets marked as a child by Line 13 (the relevant component of the graph is shown in the figure below).



For Line 13 to be reached, the if-condition in Line 10 must be *True*. This will happen if $W1$—X—$W2$ is a non-collider. Thus at least one of W_1 or W_2 is a child. W.l.o.g., let's assume that W_1 is a child. If that happens, a cycle gets created: $X \to W_1 \to M \to X$. Therefore, Condition (i) ensures that M can never be oriented by Line 13.

By Condition (ii), every ECC that is run is valid, and thus Line 16 cannot mark a child as a parent because of the correctness of the ECC (Prop. 4).

Similarly, by Condition (i), Line 9 will not mark a child as a parent. Both A and B from Line 7 are parents of X. By Lemma 27, a child cannot d-separate two non-descendants nodes and thus the set $\mathbf{S}$ in Line 9 cannot contain a child.

Line 19 will not add a child as a parent because otherwise Condition (i) would be violated. ■

We now provide sufficient faithfulness conditions for LDECC when *ECCParents*$(A, B, \mathbf{S})$ is run with *check=True*.

For a node $V \notin \text{Ne}^+(X)$, let $\mathbf{Q}(V) = \{\mathbf{S} \subseteq \text{Ne}(X) : V \perp\!\!\!\perp X|\mathbf{S}\}$ and $\mathbf{Q}_{\min}(V) = \{\mathbf{S} \in \mathbf{Q}(V) : |\mathbf{S}| = \min_{\mathbf{S}' \in \mathbf{Q}(V)} |\mathbf{S}'|\}$. With H^* and H as defined in Sec. 4.2, let $H^{(\text{check})} = \{(A, B, \mathbf{S}) \in H : \{A, B\} \cap \text{Ne}(X) = \emptyset$; and $\mathbf{Q}_{\min}(A) \cap \mathbf{Q}_{\min}(B) \neq \emptyset\}$; let $H^{(\text{single})} = \{(A, B, \mathbf{S}) \in H : |\{A, B\} \cap \text{Ne}(X)| = 1\}$; and let $H^{(\text{total})} = H^{(\text{check})} \cup H^{(\text{single})}$.

Lemma 28 *For a true graph $\mathcal{G}^*$, let $(A \to C \leftarrow B)$ be a UC such that $(A, B, \mathbf{S}) \in H^*$ for some subset $\mathbf{S}$. Then, we have that $mns_X(A) = mns_X(B)$.*

Proof Since $(A, B, \mathbf{S}) \in H^*$, by Prop. 4, this UC can be used to orient parents via an ECC. We begin by considering node A. By definition of MNS, we have $A \perp\!\!\!\perp X | \text{mns}_X(A)$. Since the nodes in $\text{mns}_X(A)$ are on the path from $X \leftarrow \ldots \leftarrow C \leftarrow A$, conditioning on $\text{mns}_X(A)$ opens up the $A \rightarrow C \leftarrow B$ path (because the nodes in $\text{mns}_X(A)$ are descendants of C). Therefore, we have $\text{mns}_X(A) \subseteq \text{mns}_X(B)$. We can make a similar argument for node B to show that $\text{mns}_X(B) \subseteq \text{mns}_X(A)$. Combining the two statements, we get $\text{mns}_X(A) = \text{mns}_X(B)$. ■

Proposition 29 (Weaker faithfulness for LDECC) *If* ECCParents$(A, B, \mathbf{S})$ *is run with* check=True*, then LDECC will identify Θ^* if: (i) LF holds for every $V \in Ne^+(X)$; (ii) $H^{(total)} \subseteq H^*$; (iii) $\forall (A, B, \mathbf{S}) \in H^{(total)}$, MFF holds for $\{A, B\} \setminus Ne(X)$; and (iv) $\forall (A, B, \mathbf{S}) \in H^*$ s.t. there is a UC $(A \rightarrow C \leftarrow B) \in \mathcal{G}^*$, we have (a) AF holds for A and B; and (b) $(A, B, \mathbf{S}) \in H^{(total)}$. Futhermore, Conditions (i)-(iv) of this proposition are implied by the conditions of Prop. 20 (i.e., this is a weaker sufficient faithfulness condition for LDECC).*

Proof The can be proved in the same way as Prop. 20 with the crucial difference that the set of ECCs that LDECC now runs is restricted to the set $H^{(\text{total})}$. Condition (ii) now ensures that every ECC that is run by LDECC is a valid ECC. We just have to show that even by restricting the ECCs to $H^{(\text{total})}$, we still run an ECC for the UCs. For this, we leverage Lemma 28 which states that the MNS of nodes A and B for the UC $A \rightarrow C \leftarrow B$ will be the same. Since, by Condition (iii), MFF holds for such nodes A and B, the check *GetMNS*(A) = *GetMNS*(B) will succeed and thus, the ECCs for these UCs will still be run.

Now, we show that the conditions of this proposition are implied by those of Prop. 20. Conditions (i),(iv)(a) of both propositions are the same. Observe that $H^{(\text{total})} \subseteq H$. Therefore, Conditions (ii),(iii) of Prop. 20 imply Conditions (ii),(iii) of this proposition. Finally, we show that Condition (iv)(b) of this proposition is also implied by the conditions of Prop. 20. Consider any $(A, B, \mathbf{S}) \in H^*$ such that the UC $A \rightarrow C \leftarrow B \in \mathcal{G}^*$ and $\{A, B\} \cap \text{Ne}(X) = \emptyset$. If $|\{A, B\} \cap \text{Ne}(X)| = 1$, then $(A, B, \mathbf{S}) \in H^{(\text{single})} \subseteq H^{(\text{total})}$. If $\{A, B\} \cap \text{Ne}(X) = \emptyset$, then by Condition (iii) of Prop. 20, MFF holds for A and B are therefore $Q_{\min}(A) \cap Q_{\min}(B) = \text{mns}_X(A) = \text{mns}_X(B)$. Thus, $(A, B, \mathbf{S}) \in H^{(\text{check})} \subseteq H^{(\text{total})}$. ■

Proposition 23 *[Testing faithfulness for LDECC] Consider running the algorithm in Fig. 8 before invoking* GetMNS*(A) for some node A in LDECC. If the algorithm returns* Fail*, MFF is violated for node A. If the algorithm returns* Unknown*, we could not ascertain if MFF holds for node A.*

Proof The set $\mathbf{Q}$ contains all subsets $\mathbf{S} \subseteq \text{Ne}(X)$ s.t. $A \perp\!\!\!\perp X | \mathbf{S}$ (Lines 1–4). The set $\mathbf{Q}_{\min}$ contains those sets from $\mathbf{Q}$ that are minimal, i.e., for every $\mathbf{S}' \in \mathbf{Q}_{\min}$, there is no subset of $\mathbf{S}'$ in $\mathbf{Q}$. If $|\mathbf{Q}_{\min}| > 1$, then there are multiple possible $\text{mns}_X(A)$ violating the uniqueness of MNS (Prop. 3). Thus we return *Fail* (Example 3 demonstrates this failure case).

Next, if Line 5 is reached, we know that $|\mathbf{Q}_{\min}| = 1$ and $\mathbf{S}$ is the single element from $\mathbf{Q}_{\min}$. Line 7 returns *Fail* if there is a set $\mathbf{S}'$ such that $\mathbf{S} \subset \mathbf{S}'$ and $\mathbf{S}' \in \mathbf{Q}$, and an intermediate set $\mathbf{S}''$ such that $\mathbf{S} \subset \mathbf{S}'' \subset \mathbf{S}'$ and $\mathbf{S}'' \notin \mathbf{Q}$ (Example 4 demonstrates this failure case). Here MFF fails because if $\mathbf{S}$ was the correct $\text{mns}_X(A)$, then we must have $\mathbf{S}'' \in \mathbf{Q}$. This is because, since $\mathbf{S}' \in \mathbf{Q}$, there cannot be an active path from A to X through nodes in $\mathbf{S}' \setminus \mathbf{S}$ (otherwise, we would not have $A \perp\!\!\!\perp X | \mathbf{S}$). Therefore, adding nodes in $\mathbf{S}'' \setminus \mathbf{S}$ to the conditioning set should not violate the independence. But

since we have $\mathbf{S}'' \notin \mathbf{Q}$, there must be a faithfulness violation and $\mathbf{S}$ is not guaranteed to be equal to $\text{mns}_X(A)$.

If Line 9 is reached, we have not been able to detect an MFF violation. However, the algorithm is not *complete*, i.e., a failure to detect a violation does not mean a violation does not exist. So we return *Unknown* which signifies that we were unable to ascertain if MFF was violated for A.

The additional tests are performed in Line 1. Since these tests are performed for every subset $\mathbf{S} \subseteq \text{Ne}(X)$, the number of extra CI tests is $\mathcal{O}(2^{|\text{Ne}(X)|})$. ■

Proposition 24 *[Testing faithfulness for SD] Consider running the algorithm in Fig. 9 with each UC detected by SD. If the algorithm returns* Fail*, then faithfulness is violated for SD. If the algorithm returns* Unknown*, we could not ascertain if the faithfulness assumptions for SD hold.*

Proof In Line 3, $\mathbf{M}$ contains every neighbor of X that gets oriented as a parent due to the input UC $A \to C \leftarrow B$ by SD. If the faithfulness assumptions for SD did hold, then all nodes in $\mathbf{M}$ would actually be parents of X. Thus there should be at least one subset $\mathbf{S} \subseteq \text{Ne}(X)$ such that $A \perp\!\!\!\perp X|\mathbf{S}$ and $\mathbf{M} \subseteq \mathbf{S}$, and likewise for B. If such a subset is not found, this means that one of the nodes that was marked as a parent was actually a child. In this case, Line 6 would return *Fail*. Similarly to the LDECC case, if we are unable to detect a faithfulness violation, we return *Unknown* to indicate that we could not determine if the faithfulness assumptions for SD hold. Since CI tests are performed for every subset $\mathbf{S} \subseteq \text{Ne}(X)$, the number of extra CI tests is $\mathcal{O}(2^{|\text{Ne}(X)|})$. ■

Appendix C. Experiments

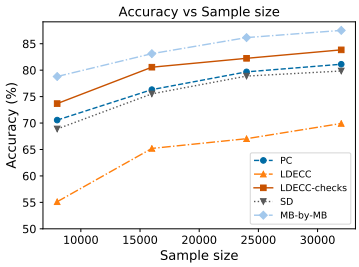
C.1. More details for the synthetic linear graph experiments.

We generate synthetic linear graphs with Gaussian errors, $N_c = 20$ covariates—non-descendants of X and Y with paths to both X and Y—and $N_m = 3$ mediators—nodes on some causal paths from X to Y. We generate edges between the different types of nodes with varying probabilities: (i) We connect the covariates to the treatment with probability p_{cx}; (ii) We connect one covariate to another with probability p_{cc}; (iii) We connect the covariates to the outcome with probability p_{cy}; (iv) We connect the treatment to the mediators with probability p_{mx}; (v) We connect one mediator to another with probability p_{mm}; (vi) We connect the mediators to the outcome with probability p_{my}; (vi) We connect a mediators to a covariate with probability p_{cm}. For our experiments, we have used $p_{cx} = p_{cc} = p_{cy} = p_{mx} = p_{mm} = p_{my} = 0.1$ and $p_{cm} = 0.05$.

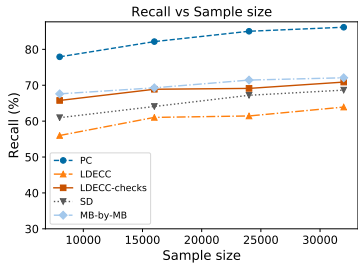
For each node V, we generate data using the following structural equation:

$$v := b_V^\top \text{pa}(v) + \epsilon_V, \;\; \epsilon_V \sim \mathcal{N}(0, \sigma_V^2),$$

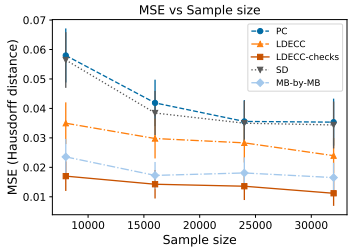
where v and $\text{pa}(v)$ are the realized values of node V and its parents, respectively; $b_V^\top \in \mathbb{R}^{|\text{Pa}(V)|}$ is the vector denoting the edge coefficients; and ϵ_V is an independently sampled noise term. Each element of b_V is sampled uniformly from the interval $[-1, -0.25] \cup [0.25, 1]$ and σ_V^2 is sampled independently from a uniform distribution $U(0.1, 0.2)$.



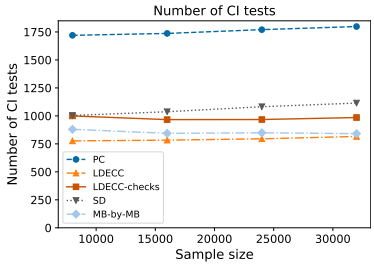
(*a*) Accuracy



(*b*) Recall



(*c*) MSE vs sample size



(*d*) Average number of CI tests

Figure 16: Results on synthetic linear Erdos-Renyi graphs.

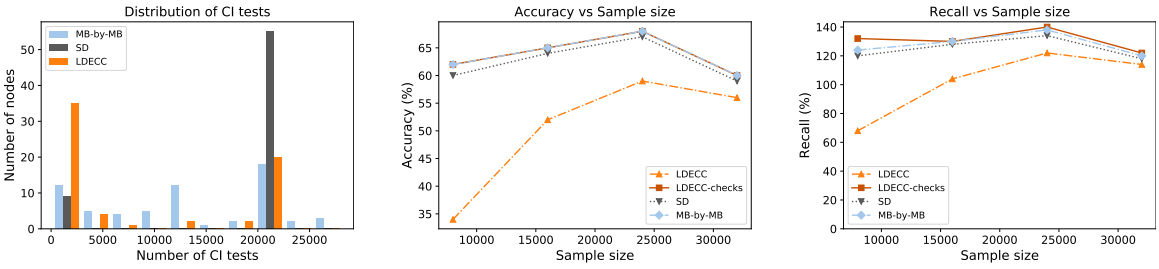
C.2. Results on synthetic linear binomial (Erdos-Renyi) graphs.

We also test our methods on binomial graphs of size $|\mathbf{V}| = 30$. In binomial graphs, an edge between two nodes is generated with some probability p_e. For our experiments, we use $p_e = 0.8$. We generate the data using linear Gaussian models where the model parameters are sampled in the same way as Sec. C.1. We compare PC, SD, MB-by-MB, and LDECC based on accuracy (Fig. 16(*a*)), recall (Fig. 16(*b*)), MSE (Fig. 16(*c*)), and number of CI tests (Fig. 16(*d*)) across four different sample sizes.

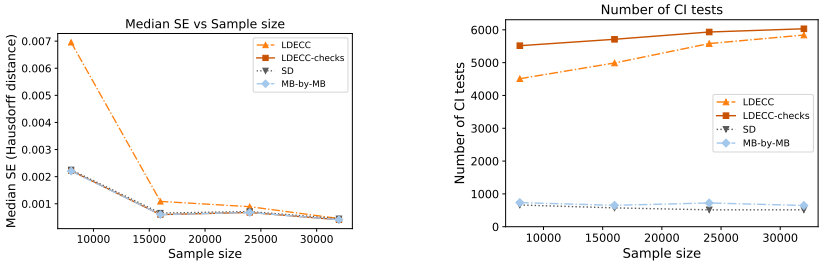
In terms of accuracy, MB-by-MB performs the best with LDECC-checks performing slightly better than SD and PC. In terms of recall, PC performs the best with LDECC-checks, MB-by-MB, and SD doing comparably. We also observe that LDECC-checks has higher accuracy and recall than LDECC. LDECC-checks and MB-by-MB have the lower MSE than PC and SD Both variants of LDECC have lower MSE than PC and SD and all four local causal discovery algorithms perform a comparable number of CI tests (and fewer tests than PC).

C.3. Additional results on semi-synthetic graphs.

We also present results on the linear Gaussian *MAGIC-IRRI* graph from *bnlearn* (Fig. 17). We plot the distribution of CI tests with a CI oracle by repeatedly setting each node as the treatment (capping the maximum number of tests per node to 20000). We see that LDECC and MB-by-MB perform differently across different nodes and outperform SD on most nodes (Fig. 17(*a*)). Next, we designated the nodes *G6003* and *BROWN* as the treatment and outcome, respectively. At four sample sizes, we sample data from the graph 100 times (capping the maximum number of tests run by each algorithm to 7000). In terms of both accuracy (Fig. 17(*b*)) and recall (Fig. 17(*c*)), LDECC-checks, SD, and MB-by-MB perform comparably while LDECC does worse. In terms of Median SE, LDECC-checks,

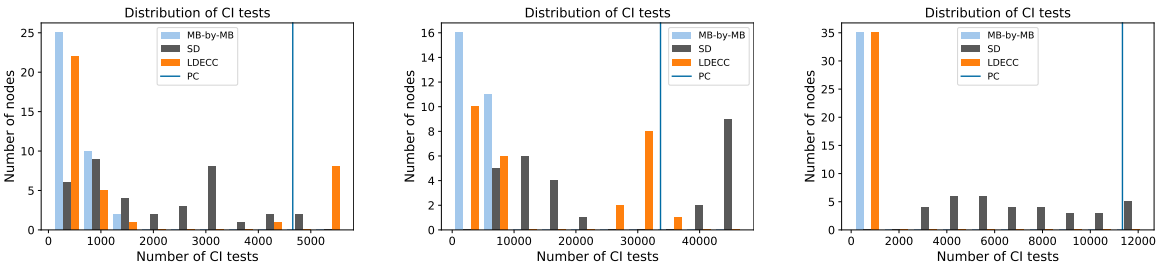


(*a*) Tests with CI oracle (*b*) Accuracy (*c*) Recall



(*d*) Median SE vs sample size (*e*) Average number of CI tests

Figure 17: Results on the semi-synthetic *MAGIC-IRRI* graph.



(*a*) *Alarm* graph. (*b*) *Insurance* graph. (*c*) *Mildew* graph.

Figure 18: Comparison of PC, SD, MB-by-MB, and LDECC based on the number of CI tests (with a CI oracle) on three discrete graphs from *bnlearn*.

SD, and MB-by-MB perform comparably (Fig. 17(*d*)) but LDECC performs substantially more CI tests (on average) than SD and MB-by-MB (Fig. 17(*e*)).

Finally, we compare PC, SD, MB-by-MB, and LDECC based on the number of CI tests (with access to a CI oracle) on three discrete graphs from *bnlearn*: *Alarm* (Fig. 18(*a*)), *Insurance* (Fig. 18(*b*)), and *Mildew* (Fig. 18(*c*)). We plot the distribution of tests for SD, MB-by-MB, and LDECC by setting each node in the graph as the treatment. We see that for most nodes on all three graphs, MB-by-MB and LDECC outperform SD. MB-by-MB has the best performance for all three graphs and performs well across all nodes.

Input: Treatment X, Outcome Y, $\mathbf{Z}$.
$\mathbf{M} \leftarrow \text{Ch}(X) \cup \text{Uo}(X)$;
$\mathbf{I} \leftarrow \{V : \text{GetMNS}(V) = \text{invalid}\}$;
$\text{PossDesc}(X) \leftarrow \mathbf{M} \cup \mathbf{I} \cup \{V \in \mathbf{V} \setminus \mathbf{I} : \mathbf{M} \cap \text{GetMNS}(V) \neq \emptyset\}$;
if $\mathbf{Z} \cap \mathit{PossDesc}(X) \neq \emptyset$ **then return** False ;
for $Q \in (\mathit{Pa}(X) \cup \mathit{Uo}(X)) \setminus \mathbf{Z}$ **do**
  if $Q \not\perp\!\!\!\perp Y | \{X\} \cup \mathbf{Z}$ **then return** False ;
end
return True;

Figure 19: Checking the Generalized Backdoor Criterion.

Appendix D. Covariate adjustment using local information

D.1. Checking the backdoor criterion.

The *backdoor criterion* (Pearl, 2009, Defn. 3.3.1) is a sufficient (but not necessary) condition for a given subset of nodes $\mathbf{Z}$ to be a valid adjustment set with respect to a treatment X and an outcome Y. Maathuis and Colombo (2015) extended this criterion to be applicable to various MECs including CPDAGs. We begin by briefly introducing the existing results in Maathuis and Colombo (2015) and then we prove that it is possible to check the backdoor criterion using only local information around X and $\mathcal{O}(|\mathbf{V}| \cdot 2^{|\text{Ne}(X)|})$ additional CI tests.

For introducing existing results, let the true DAG be $\mathcal{G}^*$ and let the corresponding CPDAG that represents the MEC of $\mathcal{G}^*$ be $\mathcal{C}^*$ (see Defn. 26).

Definition 30 (Possibly causal path) *A path $A — \ldots — B$ is said to be possibly causal if there is at least one DAG in $\mathcal{C}^*$ with a directed path from A to B: $A \to \ldots \to B$.*

Definition 31 (Visible and invisible edges (Maathuis and Colombo, 2015, Defn. 3.1)) *All directed edges in a CDPAG are said to be visible. All undirected edges in a CPDAG are said to be invisible.*

Definition 32 (Backdoor path (Maathuis and Colombo, 2015, Defn. 3.2)) *We say that path between X and Y is a backdoor path if this path does not have a visible edge out of X.*

Definition 33 (Definite non-collider (Maathuis and Colombo, 2015, Defn. 3.3)) *A nonendpoint vertex V_j on a path $< \ldots, V_i, V_j, V_k, \ldots >$ in a CPDAG is a definite non-collider if the triple $< V_i, V_j, V_k >$ is unshielded and the edges V_i—V_j and V_j—V_k are undirected.*

Definition 34 (Definite status path (Maathuis and Colombo, 2015, Defn. 3.4)) *A nonendpoint vertex B on a path p in a CPDAG is said to be of a definite status if it is either a collider or a definite non-collider on p. The path p is said to be of a definite status if all nonendpoint vertices on the path are of a definite status.*

Definition 35 (Possible Descendant) *A node A is a possible descendant of a node V iff there is a possibly causal path from V to A.*

Definition 36 (Generalized backdoor criterion (GBC) (Maathuis and Colombo, 2015, Defn. 3.7)) *A set of variables* $\mathbf{Z}$ *satisfies the backdoor criterion relative to* (X, Y) *in a CPDAG if the following two conditions: (1)* $\mathbf{Z}$ *does not contain any possible descendants of* X*; and (2)* $\mathbf{Z}$ *blocks every definite status backdoor path from* X *to* Y*.*

Intuitively, the GBC checks whether $\mathbf{Z}$ satisfies the backdoor criterion for every DAG in an MEC. Having introduced the GBC, we extend this result and prove that it is possible to check this criterion using only local information. Our procedures are related to existing methods in prior work used for covariate selection (VanderWeele and Shpitser, 2011; Entner et al., 2013) which also use very similar testing strategies. However, these works make slightly stronger assumptions on the known partial orderings (e.g., that all covariates are pre-treatment) whereas our goal is to demonstrate that a similar testing strategy along with the output of a local causal discovery algorithm is also sufficient to determine if a given subset is a valid adjustment set. These prior works also accommodate latent pre-treatment variables whereas we make the assumption of causal sufficiency throughout our work.

Definition 37 (Unoriented nodes) *We define unoriented nodes, denoted by* $Uo(X)$*, as the set of nodes* $V \in Ne(X)$ *such that the edge* X—V *is invisible in* $\mathcal{C}^*$*.*

Importantly, both local discovery procedures, SD and LDECC, find $\text{Uo}(X)$. These nodes are stored in the variable *unoriented* in the algorithms (See Figs. 3,5).

Lemma 38 *Let* $\mathbf{M} = Ch(X) \cup Uo(X)$ *and* $\mathbf{I} = \{V : mns_X(V) = invalid\}$*. The possible descendants of a node* X *are* $PossDesc(X) = \mathbf{M} \cup \mathbf{I} \cup \{V \in \mathbf{V} \setminus \mathbf{I} : \mathbf{M} \cap mns_X(V) \neq \emptyset\}$*.*

Proof The nodes in $\mathbf{M}$ are possible descendants of X. By Prop. 2, nodes in $\mathbf{I}$ are also possible descendants. For any possible descendant $V \notin \text{Ne}^+(X) \cup \mathbf{I}$ of X, there must be a path $X \to M \to \ldots \to V$, where $M \in \mathbf{M}$, in at least one DAG. Therefore, it must be the case that for every such node V, we have $\mathbf{M} \cap \text{mns}_X(V) \neq \emptyset$. ■

Proposition 39 (Checking the GBC) *Let* $\mathbf{M} = Ch(X) \cup Uo(X)$*, where* $Uo(X)$ *is defined in Defn. 37. Let* $PossDesc(X) = \mathbf{M} \cup \{V : \mathbf{M} \cap mns_X(V) \neq \emptyset\}$*. Consider a subset of nodes* $\mathbf{Z}$*. Let* $\mathbf{Q} = (Pa(X) \cup Uo(X)) \setminus \mathbf{Z}$*. Then* $\mathbf{Z}$ *satisfies the backdoor criterion for every DAG in the MEC iff: (i)* $\mathbf{Z} \cap PossDesc(X) = \emptyset$*, and (ii)* $\forall Q \in \mathbf{Q}, Q \perp\!\!\!\perp Y | \{X, \mathbf{Z}\}$*. The algorithm for checking the GBC is given in Fig. 19 and it performs* $\mathcal{O}(|\mathbf{V}| \cdot 2^{|Ne(X)|})$ *additional CI tests.*

Proof By Lemma 38, $\text{PossDesc}(X)$ contains the possible descendants of X. Condition (i) is therefore necessary since the descendants of X cannot satisfy the backdoor criterion. Thus, for the rest of the proof, we assume that $\mathbf{Z} \cap \text{PossDesc}(X) = \emptyset$.

We first prove the forward direction: If $\mathbf{Z}$ is a valid adjustment set then $\forall Q \in \mathbf{Q}, Q \perp\!\!\!\perp Y | \{X, \mathbf{Z}\}$. Since $\mathbf{Z}$ is a valid adjustment set, $\mathbf{Z}$ blocks all possibly backdoor paths in every DAG in the MEC. Therefore, we have $\forall Q \in \mathbf{Q}$, $Q \perp\!\!\!\perp Y | \{X, \mathbf{Z}\}$ because otherwise there will at least one DAG where the path $X \leftarrow Q$—$\ldots \to Y$ will be open for some $Q \in \mathbf{Q}$.

Next, we prove the backward direction: if $\forall Q \in \mathbf{Q}, Q \perp\!\!\!\perp Y | \{X, \mathbf{Z}\}$, then $\mathbf{Z}$ is a valid adjustment set. Firstly, for all $P \in \text{Pa}(X) \cap \mathbf{Z}$ (i.e., $P \notin \mathbf{Q}$), all backdoor paths of the form $X \leftarrow P$—$\ldots$—Y are blocked because $P \in \mathbf{Z}$. Since for all $Q \in \mathbf{Q}$, we have $Q \perp\!\!\!\perp Y | \{X, \mathbf{Z}\}$, all possible backdoor

paths $X \leftarrow Q \text{—} \ldots \rightarrow Y$ are blocked by $\mathbf{Z}$ and therefore it is a valid adjustment set in every DAG of the MEC.

Finally, we perform $\mathcal{O}(|\mathbf{V}| \cdot 2^{|\text{Ne}(X)|})$ additional CI tests in Lines 2,3 to find $\text{PossDesc}(X)$ (because for every node, we can find its MNS in $\mathcal{O}(2^{|\text{Ne}(X)|})$ tests). Next, in Line 6, since we only run tests for $N \in (\text{Pa}(X) \cup \text{Uo}(X)) \setminus \mathbf{Z}$, we perform $\mathcal{O}(|\text{Ne}(X)|)$ extra CI tests. ■

D.2. Finding the optimal adjustment set.

A given DAG can have multiple valid adjustment sets. Henckel et al. (2019)[Sec. 3.4] introduce a graphical criterion for linear models for determining the optimal adjustment set, i.e., the set with the lowest asymptotic variance. This criterion was later shown to hold non-parametrically (Rotnitzky and Smucler, 2019). We begin by introducing the existing results and then prove that we can find the optimal adjustment set using only local information and $\mathcal{O}(|\mathbf{V}|)$ additional CI tests (see (Fig. 20)).

Like the previous section, let the true DAG be $\mathcal{G}^*$ and let the corresponding CPDAG that represents the MEC of $\mathcal{G}^*$ be $\mathcal{C}^*$ (see Defn. 26).

Definition 40 (Possible causal nodes (Henckel et al., 2019, Sec. 3.4, Pg. 29)) *The causal nodes relative to (X, Y), denoted by $\mathit{posscn}(X, Y)$, are all nodes on possibly causal paths from X to Y, excluding X.*

Definition 41 (Forbidden nodes (Henckel et al., 2019, Sec. 3.4, Pg. 29)) *The forbidden nodes relative to (X, Y), denoted by $\mathit{forb}(X, Y)$, are defined as*

$$\mathit{forb}(X, Y) = \mathit{PossDesc}(\mathit{posscn}(X, Y)) \cup \{X\}.$$

Definition 42 (Optimal adjustment set (Henckel et al., 2019, Defn. 3.12)) *The optimal adjustment set relative to X, Y is defined as*

$$\mathbf{O}(X, Y, \mathcal{C}^*) = \mathit{Pa}(\mathit{posscn}(X, Y)) \setminus \mathit{forb}(X, Y).$$

We now show that it is possible to find the optimal adjustment set using only local information.

Lemma 43 *Given a CPDAG $\mathcal{C}$, the optimal adjustment set does not contain $\text{PossDesc}(X)$.*

Proof As stated in Defn. 42, the optimal adjustment set is $\mathbf{O}(X, Y, \mathcal{C}^*) = \text{Pa}(\text{posscn}(X, Y)) \setminus \text{forb}(X, Y)$. Therefore, we only need to show that possible descendants of X that are *not* on a causal path from X to Y cannot be in $\mathbf{O}$ (otherwise they will be in $\text{forb}(X, Y)$). Consider a node $V \in \text{Desc}(X)$ not on a causal path from X to Y. This node cannot be a parent of any node in $\text{cn}(X, Y)$. This is because if that happens, then V must also belong to $\text{cn}(X, Y)$ which leads to a contradiction. ■

Corollary 1 *The optimal adjustment set satisfies the GBC.*

Proof By Lemma 43, the optimal adjustment set does not contain any possible descendants of X. Furthermore, by definition, it is a valid adjustment set and therefore blocks all backdoor paths from X to Y. Therefore, it satisfies the GBC. ■

Input: Treatment X, Outcome Y, $\text{Pa}(X)$, $\text{Ch}(X)$, unoriented $\text{Uo}(X)$.

1 $\mathbf{M} \leftarrow \text{Ch}(X) \cup \text{Uo}(X)$;
2 $\mathbf{I} \leftarrow \{V : \text{GetMNS}(V) = \text{invalid}\}$;
3 $\text{PossDesc}(X) \leftarrow \mathbf{M} \cup \mathbf{I} \cup \{V \in \mathbf{V} \setminus \mathbf{I} : \mathbf{M} \cap \text{GetMNS}(V) \neq \emptyset\}$;
4 $\mathbf{Z} \leftarrow (\mathbf{V} \setminus \text{PossDesc}(X))$;
5 $\mathbf{O} \leftarrow \text{HenckelPrune}(X, Y, \mathbf{Z})$ (Henckel et al., 2019, Alg. 1) (Fig. 21) ;
6 **for** $M \in (Pa(X) \cup Uo(X)) \setminus \mathbf{O}$ **do**
7 | **if** $M \not\perp\!\!\!\perp Y | \{X, \mathbf{O}\}$ **then return** *noValidAdj*;
8 **end**

Output: $\mathbf{O}$

Figure 20: Finding the optimal adjustment set.

1 **def** `HenckelPrune` (*Treatment* X, *Outcome* Y, *Subset* $\mathbf{Z}$) **:**
2 | $\mathbf{Z}' \leftarrow \mathbf{Z}$;
3 | **for** $Z \in \mathbf{Z}$ **do**
4 | | **if** $Y \perp\!\!\!\perp Z | \{X\} \cup (\mathbf{Z}' \setminus \{Z\})$ **then** $\mathbf{Z}' \leftarrow \mathbf{Z}' \setminus \{Z\}$;
5 | **end**
6 | **return** $\mathbf{Z}'$;

Figure 21: The *HenckelPrune* function.

Proposition 44 (Optimal adjustment set) *Consider the algorithm in Fig. 20. It performs* $\mathcal{O}(|\mathbf{V}| \cdot 2^{|Ne(X)|})$ *CI tests and (i) returns* `noValidAdj` *if there is no valid adjustment that applies to all DAGs in the MEC; (ii) else, returns the optimal adjustment set that is valid for all DAGs in the MEC (denoted by* $\mathbf{O}$*).*

Proof In Line 4, $\mathbf{Z} = \mathbf{V} \setminus \text{PossDesc}(X)$ represents the largest possible set that could satisfy the GBC, if any such set exists. Therefore, this set $\mathbf{Z}$ would be a superset of the optimal adjustment set, if it exists (by Cor. 1). In Line 5, we invoke the pruning procedure in Henckel et al. (2019, Algorithm 1) which outputs the optimal adjustment when starting from a superset (see Fig. 21). In Line 6, we verify that the pruned set $\mathbf{O}$ is a valid adjustment set (see Prop 39). Henckel et al. (2019, Theorem 3.13(i)) also prove that an optimal adjustment set exists iff there is some valid adjustment set. Thus, if Line 7 is reached, it means that there is no valid adjustment set that applies to every DAG in the MEC.

We perform $\mathcal{O}(|\mathbf{V}| \cdot 2^{|\text{Ne}(X)|})$ CI tests to find $\text{PossDesc}(X)$. The *HenckelPrune* function and Line 6 perform $\mathcal{O}(|\mathbf{V}|)$ additional CI tests. ■