An Efficient Algorithm for Counting Markov Equivalent DAGs

Robert Ganian,[1] **Thekla Hamm,**[1] **Topi Talvitie**[2]
[1]Vienna University of Technology, Austria, Email: rganian@gmail.com, thamm@ac.tuwien.ac.at
[2]University of Helsinki, Finland, Email: topi.talvitie@helsinki.fi

Abstract

We consider the problem of counting the number of DAGs which are Markov-equivalent, i.e., which encode the same conditional independencies between random variables. The problem has been studied, among others, in the context of causal discovery, and it is known that it reduces to counting the number of so-called moral acyclic orientations of certain undirected graphs, notably chordal graphs.

Our main empirical contribution is a new algorithm which outperforms previously known exact algorithms for the considered problem by a significant margin. On the theoretical side, we show that our algorithm is guaranteed to run in polynomial time on a broad class of chordal graphs, including interval graphs.

1 Introduction

A key task in causal discovery concerns the learning of directed acyclic graphs (DAGs) that encode the direct causal relationships within a set of random variables of interest. This task is complicated by the fact that different DAGs may be Markov equivalent, which means that they can represent exactly the same set of conditional independencies between the random variables. Every Markov equivalence class of DAGs can be uniquely represented by an *essential graph*, which is a partially directed graph in which edges whose directions are not fixed by the class are represented as undirected edges (Andersson, Madigan, and Perlman 1997). If we have only observational data from the joint distribution of the random variables of interest, we can identify the essential graph, but not necessarily the correct DAG within the equivalence class represented by the essential graph.

There has been extensive research devoted to exploring the DAGs within a given equivalence class: for instance, Maathuis, Kalisch and Bühlmann (2009) estimated causal effects between pairs of variables when only the essential graph is known, while Ghassami, Salehkaleybar, Kiyavash and Bareinboim (2018) considered the problem of discovering the directions of as many undirected edges in the essential graph as possible using given number of interventional experiments. The problems of counting and sampling DAGs in a Markov equivalence class are particularly central in the area of these exploration problems, because approaches to the other problems typically reduce to them (Radhakrishnan, Solus, and Uhler 2017; 2018). For this reason, various algorithms for sampling and counting DAGs in a Markov equivalence class have been proposed (Talvitie and Koivisto 2019; He, Jia, and Yu 2015; Ghassami et al. 2019; He and Yu 2016; Bernstein and Tetali 2017).

The problem of counting and sampling DAGs in an equivalence class efficiently and directly reduces to the restricted case where the essential graph is an undirected chordal graph[1] (Gillispie and Perlman 2002). In this case, the DAGs in the Markov equivalence class are exactly the acyclic orientations of the edges of the chordal graph which do not contain an *immorality*, that is, a situation in which a vertex has two parents that are not connected by an edge. For this reason, the DAGs are known as *moral acyclic orientations* (MAOs). We denote the problem of computing the number of MAOs in chordal graphs as "#MAO".

Related Work. In their previous work, He, Jia and Yu (2015) formulated the so-called *RootPicking algorithm* for #MAO: a recurrent procedure which branches over all choices of the unique source vertex of a MAO, fixes the orientations of edges which are forced by the given choice of the source, and proceeds on each connected component. RootPicking lies at the heart of the following algorithmic approaches for #MAO: He and Yu (2016) enhanced RootPicking algorithm by identifying and extracting so-called *core graphs* to handle dense subinstances better. Ghassami et al. (2019) showed that the root-picking algorithm admits a runtime upper bound of $n^{\Delta+\mathcal{O}(1)}$ on n-vertex graphs of maximum degree Δ. Using dynamic programming on a structural decomposition of the graph, namely its clique-tree, rather than building on RootPicking, Talvitie and Koivisto (2019) showed that if the size of the largest clique in a chordal graph is k, one can count its MAOs in $O(2^k k! k^2 n)$ time.

These counting methods can also be adapted to solve the sampling problem such that after running the counting algorithm, we can obtain uniform samples from the set of MAOs in polynomial time per sample by retracing the computation that gave the number of MAOs. In particular, we can sam-

[1]A graph is chordal if it admits a special tree-like representation called a *clique-tree*.

ple a MAO of H by first sampling its root with probability proportional to the number of MAOs of H with that root, then inferring the orientations arising from this, and finally recursively sampling a MAO for each of the remaining undirected components. To determine the appropriate probabilities, we can invoke GetPartSol, in an analogous manner as in previous related works (see, e.g., the work of Ghassami et al. (2019)).

Contribution. Our main contribution is a new algorithm, AnonMAO, which solves #MAO. AnonMAO enhances RootPicking with a *vertex anonymisation* technique that reduces the number of generated subinstances which need to be solved throughout the recursion. We remark that, like the algorithms mentioned as Related Work, AnonMAO can be adapted to solve the sampling problem. Specifically, if we use the *alias method* (Vose 1991; Walker 1977) we can uniformly sample MAOs in linear time; see also the work of Ghassami et al. (2019).

On the empirical side, we show that AnonMAO vastly outperforms previously known exact algorithms for #MAO.

As our main theoretical contribution, we show that AnonMAO can solve all instances of #MAO where the clique-tree of the instance has a polynomially bounded number of subtrees in polynomial time. The techniques used here are based on an in-depth and non-trivial analysis of the structure of subinstances generated by RootPicking, notably with respect to a fixed clique-tree. In particular, for the most part this analysis is not specific to the formulation of AnonMAO but may also prove useful when considering the complexity of future algorithms for #MAO based on RootPicking.

The property of having a clique-tree with polynomially many subtrees is satisfied by a wide range of chordal graphs. We also show that the polynomial bound on the number of subtrees can be equivalently expressed as a more simple condition on the degree of the nodes in the clique-tree. One special and very well-studied subclass of chordal graphs that admit clique-trees with polynomially many subtrees is the class of *interval graphs* (Brandstädt, Le, and Spinrad 1999); consequently, as a corollary we obtain that #MAO is polynomial-time solvable on interval graphs.

2 Preliminaries

Graphs and Orientations. A graph G is defined by a finite set of *vertices* $V(G)$ and a binary relation on it, i.e., a subset of $V(G) \times V(G)$, describing a set of *edges* $E(G)$. We distinguish between *undirected* edges $(v, w) \in E(G)$, for which also $(w, v) \in E(G)$; and *directed* edges $(v, w) \in E(G)$, for which $(w, v) \notin E(G)$. Graphs all of whose edges are undirected are called *undirected* themselves. Similarly graphs all of whose edges are directed are called *directed*. For ease of presentation we denote an edge in an undirected graph containing two vertices u and v by $u - v$ and an edge in a directed graph containing two vertices u and v in that order by $u \to v$ or $v \leftarrow u$. (Note that with this notation $u - v = v - u$, but $u \to v \neq v \to u$.) A directed graph G' is an *orientation* of a graph G if for every edge $u - v$ of G, G' contains either edge $u \to v$ or $u \leftarrow v$ (but not both), and for every edge $u \to v$ of G', G contains the edge $u - v$ or the edge $u \to v$.

A *(graph) isomorphism* is a bijection between the vertex sets of two graphs under which the edge relation is invariant. Graphs between which an isomorphism exists are *isomorphic*. An *induced subgraph* $G[X]$ of a graph G is a graph defined by a subset $X \subseteq V(G)$ and the edge relation $E(G)$ restricted to X. A *subgraph* of G is a graph that is given by a subset $X \subseteq V(G)$ and a subset $F \subseteq E(G[X])$. If H is isomorphic to a subgraph of G, we say G *contains* H.

An *(undirected) path* is a graph $v_1 - v_2 - \ldots - v_\ell$ where the v_i are pairwise different. More specifically the given path is also called v_1-v_ℓ*-path* or *path from* v_1 *to* v_ℓ or *path between* v_1 *and* v_ℓ. We extend this notion to paths between a vertex and a vertex subset and paths between two vertex subsets: For X, Y a v-X-path is a v-x-path with $x \in X$; an X-Y-path is a x-Y-path with $x \in X$. A subgraph of a path that is itself a path is called *subpath*. Given a graph G and vertices $v, w \in V(G)$, the *distance* $\mathrm{dist}_G(v, w)$ between v and w in G is defined as $\min\{\ell - 1 \mid G \text{ contains a } v\text{-}w\text{-path with } \ell \text{ vertices}\}$. For $X, Y \subseteq V(G)$, $\mathrm{dist}_G(v, X)$ and $\mathrm{dist}_G(X, Y)$ are defined analogously. We define the set of *neighbours* of a vertex (set) in G as $N_G(v) = \{w \in V(G) \mid \mathrm{dist}_G(v, w) = 1\}$ (and $N_G(X) = \{w \in V(G) \mid \mathrm{dist}_G(w, X) = 1\}$). A vertex is *dominating* if it is adjacent to all other vertices in the graph.

A graph G is *connected* if for all $v, w \in V(G)$, G contains a v-w-path. A maximal (in terms of inclusion of the vertex sets) connected induced subgraph of a graph is also called a *connected component* of it.

An *(undirected) cycle* is a graph $v_1 - v_2 - \ldots v_\ell - v_1$ where the v_i are pairwise different. An undirected graph is a *tree* if it is connected and contains no cycle. An (induced) subgraph of a tree that is itself a tree is called *(induced) subtree*. An undirected graph is *chordal* if, for any cycle with more than three vertices it contains, the subgraph induced by the vertices of this cycle contains a cycle with at most three vertices.

A *directed cycle* is a graph $v_1 \to v_2 \to \ldots v_\ell \to v_1$ where the v_i are pairwise different. An orientation is *acyclic* if it contains no directed cycle.

Problem Statement. An immorality is a graph that is isomorphic to $a \to v \leftarrow b$. An orientation is *moral* if it contains no immorality as an induced subgraph.

A Markov equivalence class can be set into one-to-one correspondence with a partially directed graph (its so-called *essential graph* (Verma and Pearl 1990)) in such a way that the DAGs in the Markov equivalence class are exactly the acyclic orientations of this graph which do not create new immoralities. What is more, such a graph is efficiently computable from any DAG in the Markov equivalence class (Meek 1995). It is known that removing all the directed edges of the graph results in an undirected chordal graph (Andersson, Madigan, and Perlman 1997), and each component of this graph may be oriented independently (Gillispie and Perlman 2002). Thus the problem of determining the size of a Markov equivalence class reduces to the following problem (see also the recent work of Talvitie and Koivisto (2019)):

#MAO: Given an undirected connected chordal graph (*UCCG*) G, count the number of MAOs of G.
Note that #MAO is invariant under graph isomorphism.

The RootPicking Algorithm. He, Jia and Yu (2015) proposed a recursive algorithm for #MAO which has been used as a foundation in almost all approaches in the literature thus far. It uses the fact that any MAO of a UCCG contains a unique vertex v (called *root*) such that there is no edge $u \to v$ in the MAO. By this observation the number of MAOs is equal to the sum, over all vertices v of the UCCG, of the number of MAOs that have v as root. Now the key is that the number of MAOs of the UCCG with v as root can be computed from the number of MAOs of certain UCCGs that are subgraphs of the original UCCG. An exact description of the procedure (without additional optimisation) which we refer to simply as *RootPicking* is given in Algorithm 1.

Algorithm 1: RootPicking

Input: An UCCG G
Output: The number of MAOs of G
1 **if** $|V(G)| = 1$ **then**
2 **return** 1
3 $\texttt{Sol} = 0$
4 **for** $r \in V(G)$ **do**
5 Let G' be a copy of G
6 **for** *$a - b$ an edge in G with* $\mathrm{dist}_G(r, a) < \mathrm{dist}_G(r, b)$ **do**
7 Replace $a - b$ by $a \to b$ in G'
8 **while** *G' contains an induced subgraph $a \to b - c$* **do**
9 Replace $b - c$ by $b \to c$ in G'
10 Remove all directed edges of G'
11 $\texttt{Sol} \mathrel{+}= \prod_{\mathcal{C} \text{ connected component of } G'} \text{RootPicking}(\mathcal{C})$
12 **return** Sol

It is easy to see and has already been remarked by He, Jia and Yu (2015) that, except for the edges incident to the root r, all edges replaced in Line 7 of RootPicking, would also be replaced in Line 9 of RootPicking. Moreover, edges that contain the root would not be replaced in Line 9. This means it is equivalent to replace all edges containing the root $r - x$ by $r \to x$ and then successively replace $b - c$ by $b \to c$ for all induced $a \to b - c$. In this vein we will distinguish *root orientations*, meaning orientations of edges containing the root, and *morally propagating orientations*, meaning orientations arising from Lines 8 and 9 of the algorithm.

We will refer to the subgraphs $\mathcal{C}$ of G that are recursed on in Line 11 together with the full history which led to this recursive call. Each subinstance $\mathcal{C}$ is recursed on after a specific sequence of choices of roots in the nested executions of Line 4 leading to the recursive call on $\mathcal{C}$ in Line 11. We denote the set of these chosen roots by $R_{\mathcal{C}}$.

Optimisations and Extensions of RootPicking. Talvitie and Koivisto (2019) observed that each UCCG considered in the RootPicking algorithm is an induced subgraph of the original UCCG. By memoizing the results, they reduced the time complexity to $\tilde{O}(2^n)$ and also achieved speedups in practice (Talvitie and Koivisto 2019).

He and Yu (2016) extended the RootPicking algorithm to better handle dense graphs. They gave a recursive algorithm that, given a *core graph* (an UCCG without dominating vertices), computes a polynomial $P(m)$ such that by adding m dominating vertices, the number of MAOs is $P(m)m!$.

Clique-Trees. We will use the clique-tree characterisation of UCCGs to argue about the performance of our new algorithm. A *clique* is a graph C in which for any two vertices $v \neq w \in V(C)$, $u - v$ is an edge in C. A *clique-tree* of a graph G is a pair $(\mathcal{T}, \chi)$ of a tree $\mathcal{T}$ (whose vertices we call *nodes*) and a map χ from $V(\mathcal{T})$ to subsets of $V(G)$ such that

- for each node $t \in V(\mathcal{T})$, $G[\chi(t)]$ is a clique;
- for each vertex $v \in V(G)$, $\mathcal{T}[\{t \in V(\mathcal{T}) \mid v \in \chi(t)\}]$ is a connected subgraph of $\mathcal{T}$ with at least one vertex; and
- for every maximal $X \subseteq V(G)$ such that $G[X]$ is a clique, there is exactly one $t \in V(\mathcal{T})$ such that $\chi(t) = X$.

A clique-tree of a chordal graph can be constructed in linear time (Blair and Peyton 1993).

3 The Algorithm: AnonMAO

Algorithm 2: AnonMAO

Input: A UCCG G
Output: The number of MAOs of G
1 Let $\texttt{Sol}[\cdot, \cdot]$ be a storage indexed by $2^{V(G)} \times \{0, \ldots, |V(G)|\}$
2 Initialise all $\texttt{Sol}[\cdot, \cdot] = \textsc{Null}$
3 **return** GetPartSol(G)

Subroutine: GetPartSol

Input: A connected induced subgraph H of G
Output: The number of MAOs of H
1 **if** $|V(H)| = 1$ **then**
2 **return** 1
3 Let D be the set of dominating vertices in H
4 Let $S = V(H) \setminus D$ and $d = |D|$
5 **if** $\texttt{Sol}[S, d] = \textsc{Null}$ **then**
6 $\texttt{Sol}[S, d] = 0$
7 **for** $r \in V(H)$ **do**
8 Let H' be a copy of H
9 **for** *$a - b$ an edge in H with* $\mathrm{dist}_H(r, a) < \mathrm{dist}_H(r, b)$ **do**
10 Replace $a - b$ by $a \to b$ in H'
11 **while** *H' contains $a \to b - c$ as induced subgraph* **do**
12 Replace $b - c$ by $b \to c$ in H'
13 Remove all directed edges of H'
14 $\texttt{Sol}[S, d] \mathrel{+}= \prod_{\mathcal{C} \text{ conn. comp. of } H'} \text{GetPartSol}(\mathcal{C})$
15 **return** $\texttt{Sol}[S, d]$

Our main contribution is a new algorithm which extends RootPicking by combining the notion of *cores* (He and Yu 2016) with dynamic records. In this section, we present the algorithm and prove correctness. In the next section, we will then prove that the algorithm in fact runs in polynomial time on a broad subclass of chordal graphs, among others implying that #MAO is polynomial-time tractable on all interval graphs. We begin by recalling the definition of cores.

Definition 1. *The* core *(or* core graph*) of a graph G is obtained from G by removing all dominating vertices, i.e., is given by $G[V(G) \setminus \{v \in V(G) \mid v \text{ is dominating in } G\}]$.*

From this definition it is obvious that the core of a graph can be computed in polynomial time from the graph. Conversely, G can be reconstructed (up to isomorphism) from its core graph and the number of dominating vertices. This easy observation lies at the heart of the presented dynamic programming algorithm (Algorithm 2 along with its Subroutine GetPartSol).

Theorem 2. AnonMAO *correctly solves* #MAO.

Proof. The algorithm proceeds just as RootPicking whenever $\texttt{Sol}[S, d] = \text{NULL}$ in Line 5 of GetPartSol. By initialisation of Sol this is the case for all entries of Sol at the beginning of the algorithm. Because of this and Lines 3 and 4 of GetPartSol we can assume that an entry $\texttt{Sol}[S, d] \neq \text{NULL}$ contains the number of MAOs of some UUCG H that is an induced subgraph of G, has d dominating vertices D and $V(H) \setminus D = S$. This same number is returned when encountering an induced subgraph H' of G which has d dominating vertices D' and $V(H') \setminus D' = S$. H and H' can be easily seen to be isomorphic by mapping D to D' arbitrarily and S to S with the identity mapping. This means GetPartSol returns the number of MAOs of H which is by isomorphism the same as the number of MAOs of H'. □

4 Polynomially Tractable Instances of #MAO

In this section, we will prove that AnonMAO solves #MAO in polynomial time whenever there is a polynomial bound on the number of subtrees of the clique-tree. For the following considerations, we fix an UCCG G together with an associated clique-tree $(\mathcal{T}, \chi)$. We also introduce the following notation: for a subtree T of $\mathcal{T}$ and $v \in V(G)$, we let $T^{(v)} = \{t \in V(T) \mid v \in \chi(t)\}$.

Sub-blocks. A crucial notion that will help us understand the behavior of AnonMAO is that of *sub-blocks*. The definition of sub-blocks is motivated by the following insight into the behavior of the RootPicking algorithm.

Lemma 3. *Let $r \neq u \to v$ be an edge of G that is oriented during the* RootPicking *algorithm when considering some root $r \in V(G)$. Then $\text{dist}_{\mathcal{T}}(\mathcal{T}^{(u)}, \mathcal{T}^{(r)}) < \text{dist}_{\mathcal{T}}(\mathcal{T}^{(v)}, \mathcal{T}^{(r)})$. Moreover, there is a node $s \in V(\mathcal{T})$ on the path from $\mathcal{T}^{(r)}$ to $\mathcal{T}^{(v)}$ such that $u \in \chi(s)$ and $v \notin \chi(s)$.*

Proof. Because $u \neq r$, the orientation $u \to v$ results from a non-empty sequence of morally propagating orientations following an orientation of an edge incident to the root. This describes a path in G that starts in r and ends in u. We proceed by induction along the length of a shortest such path, i.e., a shortest path of orientations from the root to u.

In base case, when the sequence consists of just one morally propagating orientation, i.e., $r - u$ is an edge in G, there has to be $s \in V(\mathcal{T})$ such that $r, u \in \chi(s)$. Similarly because $u - v$ is an edge in G there is some $t \in V(\mathcal{T})$ such that $u, v \in \chi(t)$. As $u - v$ must be replaced due to a morally propagating orientation, $r - v$ is not an edge in G and thus $v \notin \chi(s)$. By the properties of the clique-tree, s lies on the path from $\mathcal{T}^{(r)}$ to $\mathcal{T}^{(v)}$ which proves the base case.

For the inductive step, assume that the claim holds whenever we have an induced subgraph $w \to u - v$ of G where the orientation of $w \to u$ can be explained by a sequence of propagating orientations of length $i - 1$. Consider an induced subgraph $w \to u - v$ of G where the orientation of $u \to v$ follows from a shortest path of orientations from r to u (through w) such that this path has length $i1$. Now, consider the unique bag s in $\mathcal{T}^{(u)}$ that is closest to $\mathcal{T}^{(r)}$. By our inductive assumption about the orientation of the edge $w \to u$, $w \in \chi(s)$. But then s cannot contain v, since v is not adjacent to w. Since $\mathcal{T}^{(v)}$ does intersect $\mathcal{T}^{(u)}$ and s is the bag in $\mathcal{T}^{(u)}$ closest to $\mathcal{T}^{(r)}$, s must lie on the path from $\mathcal{T}^{(v)}$ to $\mathcal{T}^{(r)}$—in particular, it satisfies the conditions of the lemma. □

Informally, Lemma 3 shows that every morally propagating orientation (i.e., all orientations of edges not incident to the current root) $u \to v$ implies the existence of a "distinguishing" bag that appears in the root-direction inside the clique-tree. This motivates the following definition, which identifies "sub-blocks" in a clique-tree which are not distinguished by such a bag (i.e., where all the bags have the same intersection with the neighbouring bag in the root direction).

Definition 4. *A* clique-tree sub-block *is a tuple $(T, \texttt{Out})$ where T is a subtree of $\mathcal{T}$ and $\texttt{Out} \subseteq N_{\mathcal{T}}(V(T))$.*

The body *$B_{T,\texttt{Out}}$ of a clique-tree sub-block $(T, \texttt{Out})$ is the vertex set of G given by $\bigcup_{t \in V(T)} \chi(t) \setminus \bigcup_{t \in \texttt{Out}} \chi(t)$.*

A clique-tree sub-block is called propagation identical *with respect to some $r \in V(G)$ if either $\forall t \in V(T)\ r \in \chi(t)$, or $\forall t \in \texttt{Out}\ \forall s, s' \in V(T)\ \chi(s) \cap \chi(t) = \chi(s') \cap \chi(t)$.*

A clique-tree sub-block is propagation identical w.r.t. a set R of roots if it is a propagation identical w.r.t. each vertex in R. Intuitively, by Lemma 3, a propagation identical sub-block (or PIS, in short) $(T, \texttt{Out})$ provides a template for specifying a vertex set of G, namely its body, between whose vertices no morally propagating orientations take place. We will use propagation identicality to show that each subinstance considered by RootPicking is the body of some clique-tree sub-block, where each root picked up to that point is either a dominating vertex in the subinstance or lies in the direction of some node $t \in \texttt{Out}$. In combination with the following observation, this will provide us with an upper bound on the number of subinstances considered by AnonMAO.

Observation 5. *The number of clique-tree sub-blocks is at most ρ^2, where ρ is the number of subtrees of $\mathcal{T}$.*

To formalise and prove the desired claim, we first consider subinstances containing the latest root picked separately in Lemma 6. We use this as a base for proving the general case in Lemma 7 and Lemma 8.

Lemma 6. *Consider a subinstance C that is recursed on during the* RootPicking *algorithm which arises from another subinstance C' after picking a root $r \in V(C')$ such that $r \in N_G(V(C))$. Then C is a connected component of $G[\{u \in V(C') \setminus \{r\} \mid \exists t \in V(\mathcal{T}) : u \in \chi(t) \wedge r \in \chi(t)]$.*

Proof Sketch. Since $Z = \{u \in V(C') \setminus \{r\} \mid \exists t \in V(\mathcal{T})\ u \in \chi(t) \wedge r \in \chi(t)\}$ is the set of all vertices at distance 1 from r in C', it obviously contains $V(C)$. The proof follows by showing that no edge in $G[Z]$ can be influenced by a morally propagating orientation. □

For the following lemma, it will be useful to recall that R_C is the set of roots chosen by RootPicking.

Lemma 7. *Consider a subinstance C that is recursed on during the* RootPicking *algorithm applied to G. There is a PIS $(T, \texttt{Out})$ with respect to R_C such that C is a connected component of $G[B_{T,\texttt{Out}} \setminus R_C]$.*

Proof Sketch. We prove the lemma by induction on the recursion depth needed to reach C as a subinstance. For the base case, notice that $V(G) = B_{\mathcal{T},\emptyset}$ and $G = G[B_{\mathcal{T},\emptyset} \setminus \emptyset]$.

Now assume that the lemma holds for all instances arising at recursion depth at most i. Let C' be an arbitrary subinstance at recursion depth i, and using the induction hypothesis let $(T', \texttt{Out}')$ be a PIS with respect to $R_{C'}$ such that $G[B_{T',\texttt{Out}'}] \setminus R_{C'}$ has C' as a connected component and let $r \in V(C')$ be an arbitrary vertex of C', i.e., a possible choice for a root when recursing on C'.

We show that for every C which occurs as subinstance at recursion depth $i+1$ arising from C' after picking r as the new root, C is a connected component of $G[B_{T,\texttt{Out}} \setminus R_C]$ for some PIS $(T, \texttt{Out})$ w.r.t. $R_C = R_{C'} \cup \{r\}$. Note that any clique-tree sub-block whose tree is contained in T' is trivially propagation identical w.r.t. $R_{C'} \cap \bigcup_{t \in V(T')} \chi(t)$.

For any C such that $r \in N_G(V(C))$, the claim follows from Lemma 6 by setting $T = T'^{(r)}$ and $\texttt{Out} = \texttt{Out}' \cap N_{\mathcal{T}}(T'^{(r)})$. Now, consider a subinstance C arising from the case where $r \notin N_G(V(C))$. We now proceed in two steps.
1. We show that that $V(C)$ is contained in some $B_{T,\texttt{Out}} \setminus R_C$ for some PIS $(T, \texttt{Out})$. To this end, let T_C be an inclusion-minimal connected subtree T_C of T' such that $V(C) \subseteq \bigcup_{t \in V(T_C)} \chi(t)$. Let $t \in N_{\mathcal{T}}(T_C)$ be a node with minimum $\mathrm{dist}_{\mathcal{T}}(t, \mathcal{T}^{(r)})$.

If $\forall s, s' \in V(T_C)\ \chi(s) \cap \chi(t) = \chi(s') \cap \chi(t)$ and $V(C) \cap \chi(t) = \emptyset$, we set $(T, \texttt{Out})$ to $(T_C, (\texttt{Out}' \cap N_{\mathcal{T}}(T_C)) \cup \{t\})$. Obviously with this choice $V(C) \subseteq B_{T,\texttt{Out}}$ and it remains to show that $(T, \texttt{Out})$ indeed is propagation identical. This is the case, because for $s, s' \in V(T_C) \subseteq V(T')$ it holds that for any $t' \in \texttt{Out}'$ that $\chi(s) \cap \chi(t') = \chi(s') \cap \chi(t')$, and by assumption on t, $\forall s, s' \in V(T_C)\ \chi(s) \cap \chi(t) = \chi(s') \cap \chi(t)$.

Otherwise, we proceed iteratively by extending T_C by t, considering the next $t \in N_{\mathcal{T}}(T_C)$ minimising $\mathrm{dist}_{\mathcal{T}}(t, \mathcal{T}^{(r)})$ for the updated T_C, and checking the condition again. It can be shown that this results in a node t and subtree T_C satisfying $\forall s, s' \in V(T_C)\ \chi(s) \cap \chi(t) = \chi(s') \cap \chi(t)$, and we can proceed as per the previous case.
2. We conclude the proof by showing that no edge in $G[B_{T,\texttt{Out}} \setminus R_C]$ for these $B_{T,\texttt{Out}}$ is directed during RootPicking in the branch of root choices that led to C. This suffices as C is, as a subinstance of RootPicking, a connected component that arises after removing the edges directed due to previous root choices. □

The next lemma provides the core statement we need to argue the desired runtime bounds for AnonMAO. In essence, it shows that every subinstance called by AnonMAO will occur as the body of a clique-tree sub-block.

Lemma 8. *Consider a subinstance C that is recursed on during the* RootPicking *algorithm applied to G. There is a clique-tree sub-block $(T, \texttt{Out})$ such that $C = G[B_{T,\texttt{Out}} \setminus R_C]$.*

Proof Sketch. Due to Lemma 7, there is a PIS $(T', \texttt{Out}')$ w.r.t. R_C such that C is a connected component of $G[B_{T',\texttt{Out}'} \setminus R_C]$. As C is connected in G and a subgraph of $G[B_{T',\texttt{Out}'}]$ we can find an inclusion-maximal subtree T_C of T' such that $V(C) \subseteq \bigcup_{t \in V(T_C)} \chi(t)$ and for all $t \in V(T_C)$, $\chi(t) \cap V(C) \neq \emptyset$. By using the propagation identicality of $(T', \texttt{Out}')$ (which ensures that vertices in T_C intersect with $\texttt{Out}'$ in the same way as T'), it can be shown that setting $\texttt{Out} = (\texttt{Out}' \cap N_{\mathcal{T}}(V(T_C))) \cup N_{T'}(V(T_C))$ yields $C = B_{T_C,\texttt{Out}}$, concluding the proof. □

Picked Roots. As we have seen in the preceding subsection, we can identify subinstances of the RootPicking algorithm by the bodies of clique-tree sub-blocks, with the caveat that we lose information about the precise identities of the picked roots. In this subsection we take care of this issue by fixing choices for the roots that can be seen as locally isomorphic to whichever roots were actually chosen by RootPicking.

Assume that the vertices of G are ordered in an arbitrary way—whenever we speak about vertices being 'smaller' or 'minimal', we understand these terms in regard to this ordering. A clique-tree sub-block $(T, \texttt{Out})$ together with a number $d \in \{0, \ldots, |B_{T,\texttt{Out}}|\}$ induces a subgraph $G(T, \texttt{Out}, d)$ of G in the following way. First, we let $D(T, \texttt{Out}, d) = \{v_1, \ldots, v_d \in B_{T,\texttt{Out}} \mid v_i \text{ is minimal dominating for } B_{T,\texttt{Out}} \setminus \{v_1, \ldots, v_{i-1}\}$ ($D(T, \texttt{Out}, d)$ will serve as our locally isomorphic choice for the previously picked roots). Then we set $V(T, \texttt{Out}, d) = B_{T,\texttt{Out}} \setminus D(T, \texttt{Out}, d)$ and $G(T, \texttt{Out}, d) = G[V(T, \texttt{Out}, d)]$.

Lemma 9. *Consider a subinstance C that is recursed on during the* RootPicking *algorithm applied to G. There is a clique-tree sub-block $(T, \texttt{Out})$ and a number d such that C is isomorphic to $G(T, \texttt{Out}, d)$.*

Proof Sketch. By Lemma 8, there is a clique-tree sub-block $(T, \texttt{Out})$ such that $C = G[B_{T,\texttt{Out}} \setminus R_C]$. Let $\ell = |B_{T,\texttt{Out}} \cap R_C|$. It is not difficult to show that $G[B_{T,\texttt{Out}} \setminus R_C]$ is isomporphic to $G(T, \texttt{Out}, \ell)$, and the lemma follows. □

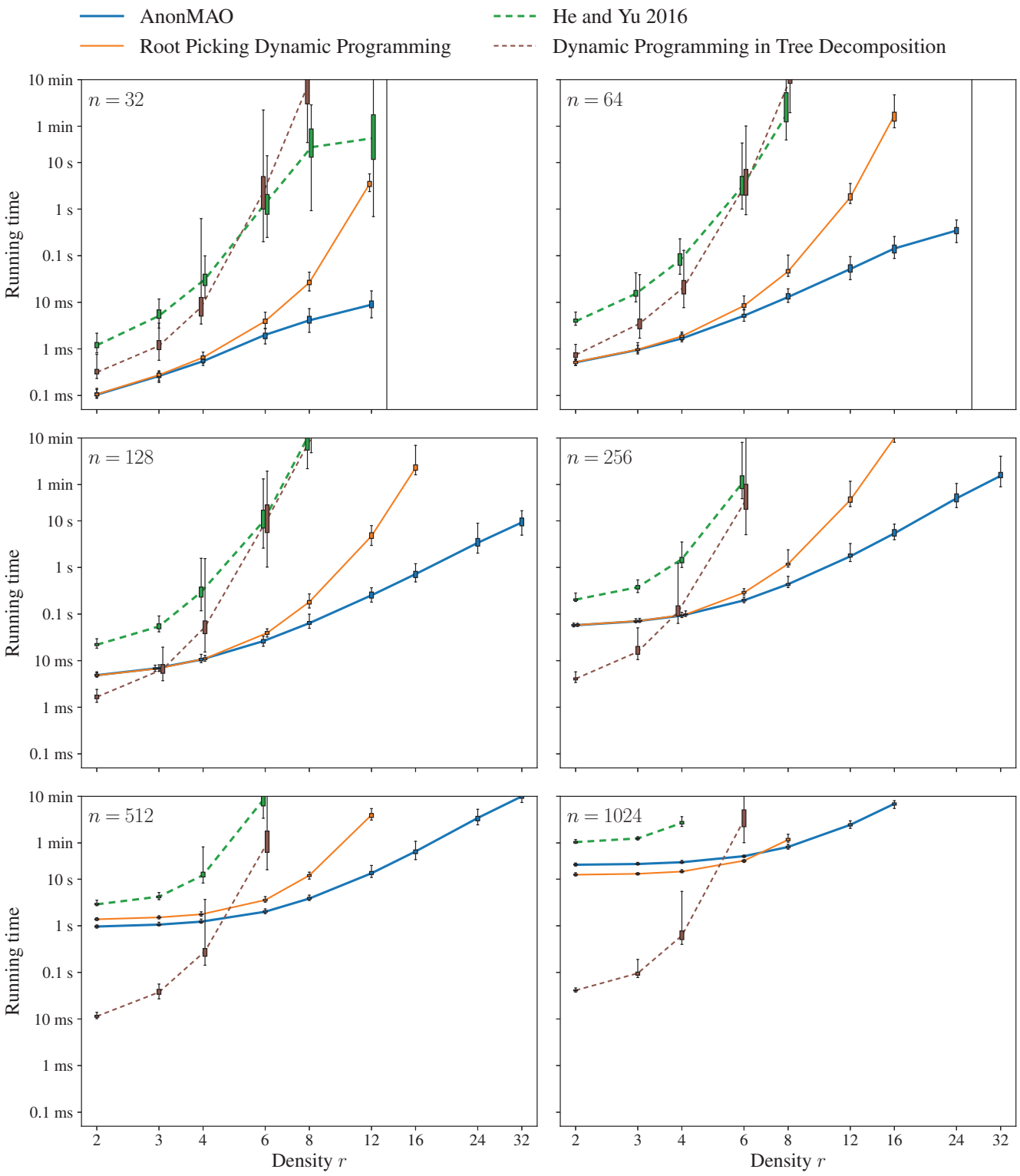

Figure 1: The running times of the algorithms for different numbers of vertices n as functions of the density parameter r. Both axes are logarithmic. The results from 50 repetitions with different inputs for each algorithm, n and r are summarised using a box plot: the box shows the range between the first and third quartile, and the whiskers show the minimum and maximum running times.

Relating $G(T, \texttt{Out}, d)$ and AnonMAO. With all the components in place, now it just remains to formalise the link between the records $G(T, \texttt{Out}, d)$ introduced in this section and the instances considered by AnonMAO.

Lemma 10. *Let (S, d) be such that there is a subinstance C of* RootPicking *applied to G with d dominating vertices and $G[S]$ as core. Then there is a clique-tree sub-block $(T, \texttt{Out})$ and some $d' \in \{0, \ldots, |B_{T,\texttt{Out}}|\}$ such that $G(T, \texttt{Out}, d')$ has d dominating vertices and $G[S]$ as core.*

Proof. By Lemma 9, there is a clique-tree sub-block $(T, \texttt{Out})$ and some $d' \in \{0, \ldots, |B_{T,\texttt{Out}}|\}$ such that C is isomorphic to $G(T, \texttt{Out}, d')$.

It is easy to see that, because $G(T, \texttt{Out}, d)$ and C are isomorphic, they have the same number of dominating vertices. This means $G(T, \texttt{Out}, d')$ has d dominating vertices.

By choice of $(T, \texttt{Out})$ in the proof of Lemma 9, we can also assume that $C = G[B_{T,\texttt{Out}} \setminus R_C]$. Hence $V(C) = B_{T,\texttt{Out}} \setminus R_C$. By definition $V(G(T, \texttt{Out}, d')) = V(T, \texttt{Out}, d') = B_{T,\texttt{Out}} \setminus D(T, \texttt{Out}, d')$. We show that any $v \in V(C) \setminus V(G(T, \texttt{Out}, d'))$ is dominating in C, and conversely any $v \in V(G(T, \texttt{Out}, d')) \setminus V(C)$ is dominating in $V(G(T, \texttt{Out}, d'))$ which implies the lemma:

Let $v \in V(C) \setminus V(G(T, \texttt{Out}, d')) = (B_{T,\texttt{Out}} \setminus R_C) \cap D(T, \texttt{Out}, d')$. By definition of $D(T, \texttt{Out}, d')$, $v - w$ is an edge in $V(C)$ for every $w \in V(C) \cap (B_{T,\texttt{Out}} \setminus D(T, \texttt{Out}, d')) = V(C) \setminus D(T, \texttt{Out}, d')$. For $w \in V(C) \cap D(T, \texttt{Out}, d')$, either v was included into $D(T, \texttt{Out}, d')$ with a smaller index than w, in which case v dominates w, or vice versa. In any case v is dominating in C.

Conversely let $v \in V(G(T, \texttt{Out}, d')) \setminus V(C) = (B_{T,\texttt{Out}} \setminus D(T, \texttt{Out}, d')) \cap R_C$. By propagation identicity with respect to $R_C \cap \bigcup_{t \in V(T)} \chi(t)$, for $s, s' \in V(T)$, $R_C \cap \chi(s) = R_C \cap \chi(s')$. Thus for all $t \in V(T)$, $v \in \chi(t)$ which implies v is dominating in $G(T, \texttt{Out}, d')$. □

We can now present our main theoretical contribution.

Theorem 11. *On graphs accompanied by a clique-tree such that the tree has polynomially many, say $p(|V(G)|)$, subtrees,* AnonMAO *runs in polynomial time.*

Proof. Let G be such a graph and $(\mathcal{T}, \chi)$ be such a clique-tree of G. Lemma 10 and its proof provide a surjection from the set of clique-tree sub-blocks and natural numbers bounded by the cardinality of their bodies to the set of core-dominating vertex pairs for subinstances considered by AnonMAO. Thus AnonMao distinguishes at most $p(|V(G)|)^2 \cdot |V(G)|$ subinstances using Observation 5. □

Since interval graphs can be characterised as chordal graphs whose clique-tree is a path (Ibarra 2009; Fulkerson and Gross 1965), we obtain the following corollary.

Corollary 12. AnonMAO *runs in polynomial time on interval graphs.*

5 On Clique-Trees with Few Subtrees

We consider now briefly the graph class for which we have shown #MAO to be polynomial time solvable.

One can reformulate the condition of a clique-tree having a polynomially bounded number of subtrees as a more local condition on the clique-tree. Among others, this allows us to check in linear time whether a clique-tree has a desired polynomial bound on the number of subtrees or not.

For a tree T, let h_T denote the number of vertices with degree at least three, which we refer to as *high-degree vertices*, Δ_T denote the maximum degree of a vertex in T, and s_T denote the number of subtrees of T.

Observation 13. *For any tree T, $s_T \leq |V(T)|^{h_T \cdot \Delta_T}$.*

Lemma 14. *For any sequence of $(h_n)_{n \in \mathbb{N}}$ and $(\Delta_n)_{n \in \mathbb{N}}$ such that there is a sequence of trees $(T_n)_{\mathbb{N}}$ such that $|V(T_n)| = n$, $h_{T_n} = h_n$ and $\Delta_{T_n} = \Delta_n$, there is a sequence of such $(T_n)_{n \in \mathbb{N}}$ and $s_{T_n} \geq \max\left\{2^{\Delta_n}, 2^{h_n}, \left(\left\lfloor \frac{n - h_n - 2}{h_n \cdot (\Delta_n - 2)} \right\rfloor + 1\right)^{h_n \cdot (\Delta_n - 2)}\right\}$.*

From this, we can show:

Corollary 15. *For a general class of trees $\mathfrak{T}$, each s_T can be polynomially (in $|V(T)|$) upper-bounded for all $T \in \mathfrak{T}$, if and only if $h_T, \Delta_T < c$ for a constant c for all $T \in \mathfrak{T}$.*

Proof Sketch. The condition is clearly sufficient by Observation 13. It is also necessary: By Lemma 14, $\mathfrak{T}$ may contain a series of trees T with arbitrarily large $|V(T)|$ such that $s_T \geq \max\left\{2^{\Delta_T}, 2^{h_T}, \left(\left\lfloor \frac{|V(T)| - h_T - 2}{h_T \cdot (\Delta_T - 2)} \right\rfloor + 1\right)^{h_T \cdot (\Delta_T - 2)}\right\}$. □

This statement shows that we can express our requirement for polynomial-time solvability (i.e., that the clique-trees have at most polynomially many subtrees) equivalently by requiring that the clique-trees have at most a constant number of high-degree vertices and maximum vertex degree. We remark that one can also show a similar relationship to the number of leaves in the clique-tree.

6 Experimental Results

We compared the practical performance of the AnonMAO algorithm to the state-of-the-art algorithms, notably:

- *RootPicking Dynamic Programming*: The root picking algorithm due to He, Jia, and Yu (2015) with the dynamic programming speedup (Talvitie and Koivisto 2019).
- *He and Yu 2016*: The algorithm by He and Yu (2016) based on recursively computing polynomials $P(m)$ for core graphs such that $P(m)m!$ is the number of MAOs when we add m dominating vertices.
- *Dynamic Programming in Tree Decomposition*: The algorithm by Talvitie and Koivisto (2019) based on dynamic programming on the clique-tree with time-complexity in $O(2^k k! k^2 n)$ parameterised by the treewidth k.

We implemented[2] AnonMAO using C++, and for the other algorithms, we used the C++ implementations of Talvitie

[2] github.com/ttalvitie/efficient-markov-equivalent-dag-counting

and Koivisto (2019). All the implementations use only one thread of execution and compute the result exactly.

We use the experimental setup formulated for this problem by He, Jia, and Yu (2015), in which we run the algorithms on randomly generated UCCGs instances with given number of vertices n and given density parameter r. The instances are generated by first generating a tree of n vertices by successively adding each vertex as a neighbour to a randomly chosen previously added vertex, and then, as long as the graph has less than rn edges, repeatedly choosing a random pair of elements and adding an edge between them if the resulting graph is chordal.

For each number of vertices $32 \leq n \leq 1024$ and density parameter $2 \leq r \leq 32$, we generated 50 UCCG instances, and ran each algorithm for each instance. The time limit was set to 10 minutes and the memory limit to 32 gigabytes. The results are shown in Figure 1. From the results we see that AnonMAO can handle much denser instances than the other algorithms, and it typically is the fastest algorithm except in very sparse cases, where the tree decomposition-based dynamic programming algorithm is faster.

7 Conclusion

We presented AnonMAO as a new exact algorithm for #MAO. AnonMAO is based on a simple dynamic programming enhancement of RootPicking in which dominating vertices are anonymised. Our empirical analysis signifies the superiority of AnonMAO compared to previously implemented exact approaches in terms of time performance on most instances. Our theoretical analysis even shows polynomial-time complexity on a large class of graphs. However one can construct a series of instances $(G_n)_{n\in\mathbb{N}}$, for which AnonMAO recurses on an exponential number of subinstances:

- $V(G_n) = \{v_1, \ldots, v_n\}$;
- for $1 \leq i < j \leq n/2$, $v_i - v_j$ is an edge in G; and
- for $n/2 + 1 \leq i \leq n$ and $j \in \{1, \ldots, n/2\} \setminus \{i - n/2\}$, $v_i - v_j$ is an edge in G.

The (for these examples unique) clique-tree of each G_n is given by $\mathcal{T}_n$, where each $\mathcal{T}_n$ consists of $n/2 + 1$ nodes, $t, t_1, \ldots t_{n/2}$ such that $t - t_i$ for $1 \leq i \leq n/2$ is an edge in $\mathcal{T}_n$. Note that each $\mathcal{T}_n$ has $\mathcal{O}(2^{\frac{n}{2}})$ subtrees. One can show that for G_n AnonMAO considers $\mathcal{O}(2^{\frac{n}{2}})$ instances.

The above example shows that the complexity of #MAO on general UCCGs remains an interesting open problem. We believe our analysis of AnonMAO may also provide useful structural insights for investigating the problem on general UCCGs, primarily in the direction of tractability.

Acknowledgments. Robert Ganian and Thekla Hamm acknowledge support from the Austrian Science Fund (FWF, Project P31336: **NFPC**). Thekla Hamm is co-funded by FWF project W1255-N23.

References

Andersson, S. A.; Madigan, D.; and Perlman, M. D. 1997. A characterization of markov equivalence classes for acyclic digraphs. *Ann. Statist.* 25(2):505–541.

Bernstein, M., and Tetali, P. 2017. On sampling graphical Markov models. *ArXiv e-prints 1705.09717*.

Blair, J. R. S., and Peyton, B. 1993. An introduction to chordal graphs and clique trees. *IMA Volumes in Mathematics and its Applications* 56:1–29.

Brandstädt, A.; Le, V. B.; and Spinrad, J. P. 1999. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics.

Fulkerson, D. R., and Gross, O. A. 1965. Incidence matrices and interval graphs. *Pacific J. Math.* 15(3):835–855.

Ghassami, A.; Salehkaleybar, S.; Kiyavash, N.; and Bareinboim, E. 2018. Budgeted experiment design for causal structure learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 1719–1728.

Ghassami, A.; Salehkaleybar, S.; Kiyavash, N.; and Zhang, K. 2019. Counting and sampling from markov equivalent dags using clique trees. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, 3664–3671.

Gillispie, S. B., and Perlman, M. D. 2002. The size distribution for markov equivalence classes of acyclic digraph models. *Artif. Intell.* 141(1/2):137–155.

He, Y., and Yu, B. 2016. Formulas for counting the sizes of Markov equivalence classes of directed acyclic graphs. *ArXiv e-prints 1610.07921*.

He, Y.; Jia, J.; and Yu, B. 2015. Counting and exploring sizes of Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research* 16:2589–2609.

Ibarra, L. 2009. The clique-separator graph for chordal graphs. *Discrete Applied Mathematics* 157(8):1737–1749.

Maathuis, M. H.; Kalisch, M.; and Bühlmann, P. 2009. Estimating high-dimensional intervention effects from observational data. *Ann. Statist.* 37(6A):3133–3164.

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

Radhakrishnan, A.; Solus, L.; and Uhler, C. 2017. Counting markov equivalence classes by number of immoralities. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*.

Radhakrishnan, A.; Solus, L.; and Uhler, C. 2018. Counting markov equivalence classes for DAG models on trees. *Discrete Applied Mathematics* 244:170–185.

Talvitie, T., and Koivisto, M. 2019. Counting and sampling markov equivalent directed acyclic graphs. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, 7984–7991.

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

Vose, M. D. 1991. A linear algorithm for generating random numbers with a given distribution. *IEEE Trans. Software Eng.* 17:972–975.

Walker, A. J. 1977. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.* 3(3):253–256.