A Fixed-Parameter Tractable Algorithm for Counting Markov Equivalence Classes with the Same Skeleton

Vidya Sagar Sharma
School of Technology and Computer Science
Tata Institute of Fundamental Research, Mumbai
vidyasagartifr@gmail.com

Abstract

Causal DAGs (also known as Bayesian networks) are a popular tool for encoding conditional dependencies between random variables. In a causal DAG, the random variables are modeled as vertices in the DAG, and it is stipulated that every random variable is independent of its non-descendants conditioned on its parents. It is possible, however, for two different causal DAGs on the same set of random variables to encode exactly the same set of conditional dependencies. Such causal DAGs are said to be *Markov equivalent*, and equivalence classes of Markov equivalent DAGs are known as *Markov Equivalent Classes* (MECs). Beautiful combinatorial characterizations of MECs have been developed in the past few decades, and it is known, in particular that all DAGs in the same MEC must have the same "skeleton" (underlying undirected graph) and v-structures (induced subgraph of the form $a \to b \leftarrow c$).

These combinatorial characterizations also suggest several natural algorithmic questions. One of these is: given an undirected graph G as input, how many distinct Markov equivalence classes have the skeleton G? Much work has been devoted in the last few years to this and other closely related problems. However, to the best of our knowledge, a polynomial time algorithm for the problem remains unknown.

In this paper, we make progress towards this goal by giving a fixed parameter tractable algorithm for the above problem, with the parameters being the treewidth and the maximum degree of the input graph G. The main technical ingredient in our work is a construction we refer to as *shadow*, which lets us create a "local description" of long-range constraints imposed by the combinatorial characterizations of MECs.

Introduction

Graphical models provide frameworks for concisely describing conditional independence constraints between subsystems of a larger system. Due to their generality, they find applications in several areas, and have also been studied extensively from a theoretical point of view.

In this paper, our focus is on a specific type of graphical model: namely directed acyclic graphical (DAG) models or Bayesian networks. Here, a probability distribution over a set of random variables V is said to satisfy a DAG G with vertices V if and only if for every w in V, w is independent of the set of all its non-descendants conditional on the set of its parents (x is a parent of y in G if $x \to y$ is in G). The set of probability distributions that satisfy G is denoted Markov(G). Further, for disjoint $A, B, S \subseteq V$, G is said to *entail* that A is independent of B given S (written $G \models A \perp B|S$) if and only if A is independent of B given S in every probability measure in Markov(G). Two DAGs are said to be *Markov equivalent* if both entail the same set of such conditional independence relations. Verma and Pearl (1990) gave an elegant graphical characterization of this equivalence: two DAGs are Markov equivalent if and only if they have the same *skeleton* (underlying undirected graph) and the same set of v-structures (induced subgraphs of the form $a \to b \leftarrow c$). DAGs that are Markov equivalent to each other are said to belong to the same *Markov equivalence class* (MEC) (see fig. 1).

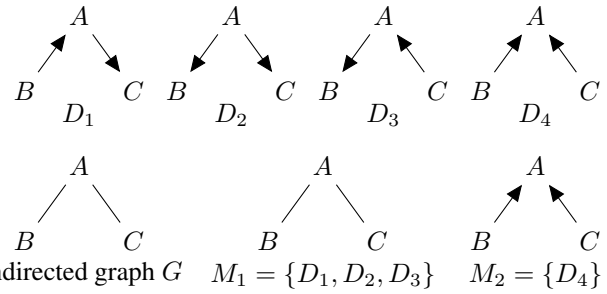

Figure 1: Markov equivalent DAGs and MECs with the same skeleton. D_1, D_2, and D_3 are Markov equivalent, while D_4 is not equivalent to D_1, D_2, and D_3. The MEC $M_1 = \{D_1, D_2, D_3\}$ contains D_1, D_2 and D_3, and its graphical representation is the union of D_1, D_2, and D_3. The MEC M_2 contains only D_4, and its graphical representation matches D_4. The MECs M_1 and M_2 share G as their skeleton, and in fact are the only MECs with skeleton G. Both M_1 and M_2 entail a conditional independence relation of the form $B \perp C \mid S$, where in M_1, $S = \{A\}$, and in M_2, $S = \emptyset$.

As an MEC consists of Markov equivalent DAGs, it uniquely represents the set of conditional independence relations represented by the DAGs it contains and is graphically represented by a partially directed graph which is the graph-

ical union of the DAGs it contains. We treat an MEC and its graphical representation as the same. Since all the DAGs that belong to an MEC have the same skeleton and the same set of v-structures, an MEC is determined by its skeleton and the set of v-structures it contains. Andersson, Madigan, and Perlman (1997) gave a necessary and sufficient condition for a partially directed graph to be an MEC (see Theorem 1 below). Meek (1995) formulated rules to construct an MEC based on knowledge about the conditional independence relations among random variables. For a set of conditional independence relations involving a set of random variables V and represented by an MEC M, Meek (1995, p. 3) also showed that two random variables A and B are non-adjacent in the skeleton of M if and only if there exists $S \subseteq V \setminus \{A, B\}$ such that M entails that $A \perp B \mid S$.

This implies that two MECs M_1 and M_2 sharing the same skeleton entail sets of conditional independence relations $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively, which have the following important relation: $\mathcal{M}_1$ contains a conditional independent relation indicating that A is independent of B given some $S_1 \subseteq V \setminus \{A, B\}$ if and only if $\mathcal{M}_2$ also contains a conditional independence relation signifying that A is independent of B given some $S_2 \subseteq V \setminus \{A, B\}$ (S_1 are S_2 may not be same). In other words, MECs with the same skeleton can be related by an equivalence relation that has not just a graphical representation (i.e., that they have the same underlying undirected graph), but also a natural statistical one (the one given above).

This connection between MECs with the same skeleton motivates the problem of understanding this class. In particular, a natural question to ask is: how many MECs are in this class?

Our Contributions We now formalize the problem outlined above. Our input consists of a connected undirected graph G with n nodes. Our objective is to determine the count of MECs that have G as their skeleton. The primary contribution of this paper is the introduction of a fixed-parameter tractable (FPT) algorithm. This algorithm, when given an undirected graph G, computes the number of MECs whose skeletons match G. The algorithm's parameters are the degree and the treewidth of the input undirected graph. (An algorithm is said to be FPT with respect to a parameter if there is a constant c and a computable function f such that the algorithm's runtime on instances of size n for which the value of the parameter is k is bounded above by $f(k) \cdot n^c$: the crucial point here is that the degree of the polynomial in n *does not* depend upon the parameter k (Cygan et al. 2015).) Our main result presents an algorithm capable of counting the MECs associated with an input undirected graph G with n nodes and having a degree of δ and a treewidth of k. This counting can be achieved in $O(n(2^{O(k^4\delta^4)} + n^2))$ time. Importantly, the runtime of our algorithm remains polynomially bounded when the parameters δ and k are both bounded above by constants. As an illustrative example, our algorithm demonstrates polynomial runtime for tree graphs with bounded degrees.

As of now, we do not know the precise computational complexity of this problem, and we consider our result as a first step towards a complete resolution of this question. This mirrors the situation of the problem of counting DAGs of an MEC, where initially an algorithm that was exponential in the degree of the graph was given by Ghassami et al. (2019) and then improved by Talvitie and Koivisto (2019) who gave a fixed parameter tractable algorithm for the problem. Finally, a polynomial algorithm was provided by Wienöbst, Bannach, and Liśkiewicz (2021). We hope that the techniques introduced in our paper will be useful in the further study for the problem.

Related Work The problem of counting MECs with a given number of nodes (instead of a skeleton) has received extensive attention in the literature. Gillispie and Perlman (2001) developed a computer program for computing the number of MECs with n nodes. Gillispie and Perlman (2002) created a computer program to enumerate Markov equivalence classes, studying class size distributions and the number of edges for graphs up to 10 vertices. They also observed that the ratio of DAGs to the number of MECs seems to asymptotically converge to around 3.7. Steinsky (2003) presented a recursive formula for counting Markov equivalence classes of size 1. Gillispie (2006) provided a recursive algorithm for counting MECs of any size. He, Jia, and Yu (2013) analyze the set of MECs with n nodes by constructing a Markov chain on the space of MECs and show that most edges of an MECs are directed. More recently, Schmid and Sly (2022) show that the expected ratio of the number of DAGs and the number of MECs approaches a positive constant when the number of nodes goes to infinity.

All of the previous results focused on the class of MECs with a given number of nodes, and not with a given skeleton. Radhakrishnan, Solus, and Uhler (2016) considered instead the problem of counting MECs with a given skeleton. They classified MECs based on the number of v-structures present and derived a generating function for counting MECs. They experimentally demonstrated that the generating function varies for graphs with the same number of vertices. In subsequent work, Radhakrishnan, Solus, and Uhler (2018) delved further into the problem of counting MECs with the same skeleton. They explored generating functions for specific graph structures (e.g., path graphs, cycle graphs, star graphs, and bi-star graphs) and provided tight lower and upper bounds for the number of MECs in any tree.

We are not aware of any progress on the problem of counting MECs for general graphs, and to the best of our knowledge, this paper is the first to introduce a fixed-parameter tractable algorithm for this problem.

Preliminaries

In this paper, we adopt mostly the terminology and notation introduced by Andersson, Madigan, and Perlman (1997) and Diestel (2005). We defer details of standard terminology about chain graphs and tree decompositions to the supplementary material[1]. Additionally, all omitted proofs can be found in the supplementary material .

[1]The supplementary material is available at https://arxiv.org/abs/2310.04218.

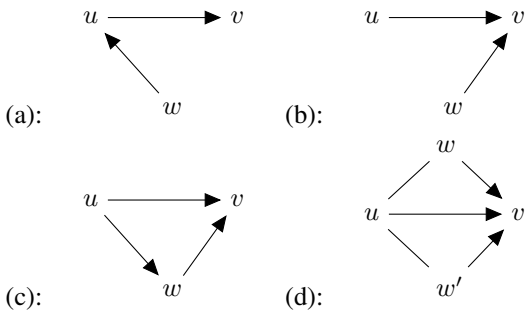

Figure 2: Strongly protected $u \to v$.

Markov Equivalence Class. A causal DAG encodes a set of conditional independence relations between random variables represented by its vertices. Two DAGs are said to belong to the same *Markov equivalence class* (MEC) if both encode the same set of conditional independence relations. Verma and Pearl (1990) showed that two DAGs are in the same MEC if, and only if, both have (i) the same skeleton and (ii) the same set of v-structures. An MEC can be represented by the graph union of all DAGs in it. Andersson, Madigan, and Perlman (1997) give the following necessary and sufficient conditions for a graph to be an MEC.

Theorem 1. *(Andersson, Madigan, and Perlman 1997) A graph G is an MEC if, and only if,*

1. *G is a chain graph.*
2. *For every chain component τ of G, G_τ is chordal, i.e., every undirected connected component of G is chordal.*
3. *The configuration $a \to b - c$ does not occur as an induced subgraph of G.*
4. *Every directed edge $u \to v \in G$ is strongly protected in G, i.e., $u \to v$ is a part of at least one of the subgraphs of G as shown in fig. 2.*

With a slight abuse of terminology, we equate the chain graph with chordal components that represent an MEC with the MEC itself, and refer to both as an "MEC".

Definition 2 (Partial MEC). A graph M is identified as a *partial MEC* if it meets these conditions: it is a chain graph, its undirected connected components are chordal, and it lacks subgraphs in the form of $u \to v - w$. In essence, a partial MEC follows conditions 1, 2, and 3 of Theorem 1. It's worth noting that every MEC is a partial MEC, and a partial MEC adhering to condition 4 is an MEC.

The notion of partial MEC has been studied earlier, for example, by Van Der Zander and Liskiewicz (2016).

Notation 3. *Let $G = (V, E)$ be a graph, where V denotes the vertex set and $E \subseteq V \times V$ is the edge set. For any vertices u and v in V, if both (u, v) and (v, u) are in E, we denote an undirected edge between u and v as $u - v$. If only (u, v) is in E (and (v, u) is not), we indicate a directed edge from u to v as $u \to v$. We use V_G to represent the set of vertices in graph G and E_G as its edge set. We say that an MEC M is an MEC of an undirected graph G if skeleton$(M) = G$. Similarly, we say a partial MEC M is a partial MEC of an undirected graph G if skeleton$(M) = G$. For an undirected graph G, we denote by MEC(G) the set of MECs of G, and by* PMEC(G) *the set of partial MECs of G. We denote the collection of v-structures in G as $\mathcal{V}(G)$. For an undirected graph G, we denote by $N(X, G)$ the set of neighbors of nodes in X in G.*

Paths. For a graph G, a sequence $P = (u_1, u_2, \ldots, u_{l-1}, u_l)$ of distinct vertices u_1 to u_l is said to be a ***path*** from u_1 to u_l if, for all $1 \leq i < l$, either $u_i - u_{i+1} \in E_G$ or $u_i \to u_{i+1} \in E_G$. The number of edges in the path is referred to as the ***length of the path*** (the length of P above is $l - 1$). P is considered an ***undirected path*** of G if, for all $1 \leq i < l$, $u_i - u_{i+1} \in E_G$. If the path contains directed edges, it is termed a *directed path* of G. A path $P = (u_1, u_2, \ldots, u_{l-1}, u_l)$ is considered a ***chordal path*** of G if there is no edge in G between any two non-adjacent nodes of P. In other words, for $1 \leq i < j \leq l$, if $j \neq i + 1$, then neither $u_i - u_j \in E_G$, nor $u_i \to u_j \in E_G$, nor $u_j \to u_i \in E_G$. For a graph G, a path $P = (u_1, u_2, \ldots, u_{l-1}, u_l)$ is defined as a ***triangle-free path*** of G if there is no edge between the two adjacent nodes of any node u_i in P. More formally, for any $2 \leq i \leq l - 1$, neither $u_{i-1} - u_{i+1} \in E_G$, nor $u_{i-1} \to u_{i+1} \in E_G$, nor $u_{i+1} \to u_{i-1} \in E_G$, i.e., $u_{i-1} - u_{i+1} \notin$ skeleton(G). We say that $P = (u_1, u_2, \ldots, u_{l-1}, u_l)$ is a ***path from*** (u, v) ***to*** (x, y) if the first two nodes of P are u and v (i.e., $u_1 = u$ and $u_2 = v$), and the last two nodes of P are x and y (i.e., $u_{l-1} = x$ and $u_l = y$). Similarly, if the first two nodes of P are u and v, and the last node of P is w, then we say P is a ***path from*** (u, v) ***to*** w.

The notion of triangle-free paths does not seem to have been used much in the literature, but turns out to be crucial for our purposes. It is clear that every chordless path is also a triangle-free path. The utility of triangle-free paths for us stems partly from the following two results.

Proposition 4. *Let G be an undirected chordal graph, and P be a triangle-free path in G. Then, P is also a chordless path.*

The following observation shows the transitive nature of triangle-free paths in a chain graph.

Proposition 5 (Concatenation of Triangle-Free Paths). *Consider a chain graph G with chordal undirected components. Let $u, v, x, y, w \in V_G$ be (possibly non-distinct) vertices of G. Suppose $P_1 = (a_1 = u, a_2 = v, \ldots, a_{l-1} = x, a_l = y)$ and $P_2 = (b_1 = x, b_2 = y, \ldots, b_{m-1}, b_m = w)$ are triangle-free paths in G from (u, v) to (x, y) and from (x, y) to w, respectively. Then, their concatenation $P = (a_1 = u, a_2 = v, \ldots, a_{l-1} = x, a_l = y, b_3, b_4, \ldots, b_{m-1}, b_m = w)$ is a triangle-free path in G from (u, v) to w.*

We will also need the standard notion of a tree decomposition (see, e.g., Blair and Peyton (1993)).

Definition 6 (Tree decomposition). Given an undirected graph $G = (V, E)$ a *tree decomposition* of G is a tuple $(X = \{X_1, X_2, X_3, \ldots, X_l\}, T)$, where each $X_i \subseteq V$ and T is a tree with vertex set X satisfying the conditions that (i) for every edge $u - v \in E$, there exists $X_i \in X$ such

that $u, v \in X_i$, and (ii) for any $v \in V$, the set of all X_j that contain v is connected in T. If the size of the largest element of X is $w + 1$, then the tree decomposition is said to have *width* w. The *treewidth* of G is the smallest possible width of a tree decomposition of G.

Main Results

We start with a formal description of the problem.

Problem 1 (Counting MECs of an undirected graph). ***Input:*** *An undirected graph G.*
Output: $|\text{MEC}(G)|$.

To compute $|\text{MEC}(G)|$, we first assign a unique identifier, referred to as a "shadow" to each MEC of G. We then proceed to count the MECs of G that have the same shadow S, for each possible shadow S. The sum of these counts, over all possible shadows S, yields $|\text{MEC}(G)|$ (Lemma 10). We now define the concept of a shadow for an MEC.

Definition 7 (Shadow of an MEC). Let M be an MEC of G, $Y \subseteq V_G$, $O \in \text{PMEC}(G[Y])$ (i.e., O is a partial MEC with skeleton $G[Y]$), and $P_1 : E_O \times E_O \to \{0, 1\}$ and $P_2 : E_O \times V_O \to \{0, 1\}$ be two functions. We define (O, P_1, P_2) as a *shadow* of M on Y, denoted as $(O, P_1, P_2) = \text{shadow}(M, Y)$ if the following conditions are met:

1. $M[Y] = O$, meaning that O is an induced subgraph of M on Y.
2. For $(u, v), (x, y) \in E_O$, $P_1((u, v), (x, y)) = 1$ if $(u, v) \neq (x, y)$ and there exists a triangle-free path from (u, v) to (x, y) in M, otherwise $P_1((u, v), (x, y)) = 0$.
3. For $(u, v) \in E_O$ and $w \in V_O$, $P_2((u, v), w) = 1$, if $v \neq w$ and there exists a triangle-free path from (u, v) to w in M, otherwise $P_2((u, v), w) = 0$.

With a slight abuse of notation, we refer to (O, P_1, P_2) as a *shadow of an undirected graph* G, denoted as $(O, P_1, P_2) \in \text{shadow}(G)$, where $O \in \text{PMEC}(G)$, and $P_1 : E_O \times E_O \to \{0, 1\}$ and $P_2 : E_O \times V_O \to \{0, 1\}$ are two functions.

Remark 8 (P_1 does not determine P_2). We emphasize that P_2 is not necessarily determined by P_1. Further details can be found in the supplementary material.

The following definition is used to partition the set $\text{MEC}(G)$.

Definition 9 ($\text{MEC}(G, O, P_1, P_2)$). Let G be an undirected graph, $Y \subseteq V_G$, and $(O, P_1, P_2) \in \text{shadow}(G[Y])$. We define $\text{MEC}(G, O, P_1, P_2)$ to be the set of MECs M of G for which (O, P_1, P_2) is a shadow of M on Y. More formally, $\text{MEC}(G, O, P_1, P_2) \coloneqq \{M : M \in \text{MEC}(G) \text{ and } (O, P_1, P_2) = \text{shadow}(M, V_O)\}$.

Lemma 10 uses the partitioning of the set of MECs of G based on their shadow to compute $|\text{MEC}(G)|$.

Lemma 10. *Let G be an undirected graph. Then, for any $Y \subseteq V_G$, $|\text{MEC}(G)| = \sum_{(O,P_1,P_2) \in \text{shadow}(G[Y])} |\text{MEC}(G, O, P_1, P_2)|$.*

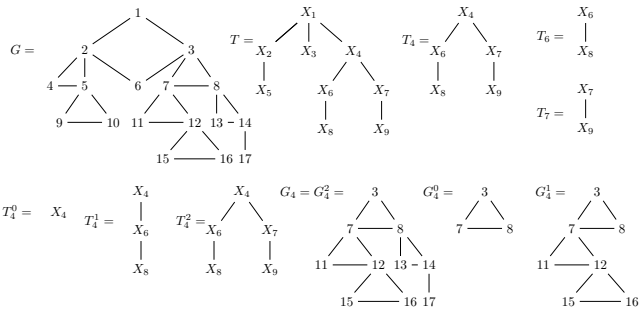

Figure 3: Example: G is an undirected graph, T is a tree decomposition of G, where $X_1 = \{1, 2, 3\}, X_2 = \{2, 4, 5\}, X_3 = \{2, 3, 6\}, X_4 = \{3, 7, 8\}, X_5 = \{5, 9, 10\}, X_6 = \{7, 11, 12\}, X_7 = \{8, 13, 14\}, X_8 = \{12, 15, 16\}$, and $X_9 = \{14, 17\}$.

Projection of an MEC

Lemma 10 shows that if we possess knowledge about $|\text{MEC}(G, O, P_1, P_2)|$ for every shadow $(O, P_1, P_2) \in \text{shadow}(G[Y])$ for some $Y \subseteq V_G$, then we can compute $|\text{MEC}(G)|$. Our goal now is to construct a recursive algorithm to compute $|\text{MEC}(G, O, P_1, P_2)|$. At a high level, the recursive structure of this algorithm is similar to many other algorithms that employ tree decompositions, the novelty being in the method for implementing each step of the recursion. We therefore give only a short description of this structure here and defer a detailed description to the supplementary material.

Let $(X = \{X_1, X_2, \ldots, X_k\}, T)$ be a tree decomposition of G, with the root node being X_1. For each node X_i in T, the subtree rooted at X_i is denoted as T_i. The induced subgraph of G corresponding to T_i is denoted as G_i. Thus, our goal is to to count the MECs of G_1.

Let the children of X_i in the subtree T_i be denoted as $(X_{i_1}, X_{i_2}, \ldots, X_{i_l})$. For $0 \leq j \leq l$, we use T_i^j to represent the induced subtree of T_i containing node X_i and the nodes of $T_{i_1}, T_{i_2}, \ldots, T_{i_j}$. Specifically, T_i^0 consists of only the node X_i, and $T_i^l = T_i$. The corresponding induced subgraph of G for T_i^j is denoted as G_i^j. When the number of children of X_1 is denoted as p, it holds that $T_1^p = T_1 = T$, and correspondingly, $G_1^p = G_1 = G$. In this case, our goal is to count the MECs of G_1^p (see fig. 3).

Standard arguments show (see the supplementary material) that for $j \geq 1$, if we cut the edge $X_i - X_{i_j}$ in the tree T_i^j, then the two subtrees T_i^{j-1} and T_{i_j} of T_i we obtain are tree decompositions of the corresponding subgraphs G_i^{j-1} and G_{i_j} of G_i^j. The recursive structure of the algorithm is now as follows. The base case is of graphs $G_i^0 = G[X_i]$, for $1 \leq i \leq k$. For each shadow $(O, P_1, P_2) \in \text{shadow}(G[X_i])$, we compute $|\text{MEC}(G_i^0, O, P_1, P_2)|$ using brute force approach (which can be done, since each X_i is of size at most one more than the width of the tree decomposition).

We now turn to the recursive case. This is the main technical content of the paper. For $1 \leq i \leq k$, and $j \geq 1$, we give relations between MECs of G_i^j, G_i^{j-1}, and G_{i_j}, such that if

we have knowledge about (a) $|\text{MEC}(G_i^{j-1}, O_1, P_{11}, P_{12})|$ for each shadow $(O_1, P_{11}, P_{12}) \in \text{shadow}(G_i^{j-1}[X_i \cup N(X_i, G_i^{j-1})])$, and (b) $|\text{MEC}(G_{i_j}, O_2, P_{21}, P_{22})|$ for each shadow $(O_2, P_{21}, P_{22}) \in \text{shadow}(G_{i_j}[X_{i_j} \cup N(X_{i_j}, G_{i_j})])$, then we can compute $|\text{MEC}(G_i^j, O, P_1, P_2)|$ for each shadow $(O, P_1, P_2) \in \text{shadow}(G_i^j[X_i \cup N(X_i, G_i^j)])$. This gives us a recursive method to compute $|\text{MEC}(G_1^p, O, P_1, P_2)|$ for each shadow (O, P_1, P_2) of G_1^p on $X_1 \cup N(X_1, G_1^p)$. Further, using Lemma 10, we can compute $|\text{MEC}(G = G_1^p)|$.

It turns out that the problem of establishing these relations can be abstracted out in simpler terms. We consider an undirected graph H, and let H_1 and H_2 be two induced subgraphs of H such that $H = H_1 \cup H_2$, and $I = V_{H_1} \cap V_{H_2}$ is a vertex separator of H that separates $V_{H_1} \setminus I$ and $V_{H_2} \setminus I$. Let $S_1 \subseteq V_{H_1}$ and $S_2 \subseteq V_{H_2}$ such that $S_1 \cap S_2 = I$. To connect to the previous discussion, H plays the role of G_i^j, H_1 that of G_i^{j-1}, H_2 that of G_{i_j}, S_1 plays the role of X_i, and S_2 plays the role of X_{i_j} ($X_i \cap X_{i_j} = V_{G_i^{j-1}} \cap V_{G_{i_j}} = I$, is a vertex separator, comes from the tree decomposition properties). Many forthcoming definitions, observations, lemmas, and theorem, will use the set-up of H, H_1, H_2, S_1, S_2 and I stated above.

The chief technical contribution of this paper is to construct tools for establishing relations between shadows of the MECs of H, H_1, and H_2. As a first step towards this, we define a relation between MECs of H and MECs of H_1 and H_2 in the following definition.

Definition 11 (**Projection**)**.** Let G be an undirected graph, $X \subseteq V_G$, $G' = G[X]$ be an induced subgraph of G, M be an MEC of G, and M' be an MEC of G'. We say that M' is a *projection* of M onto X, denoted as $\mathcal{P}(M, X) = M'$, if $\mathcal{V}(M') = \mathcal{V}(M[X])$ (Recall from Notation 3 that $\mathcal{V}(G)$ is the set of v-structures of G).

With slight overloading of the function $\mathcal{P}$, for two subsets $X_1, X_2 \subseteq V_G$, we define (M_1, M_2) to be the projection of M onto (X_1, X_2), denoted as: $\mathcal{P}(M, X_1, X_2) = (M_1, M_2)$, if $M_1 = \mathcal{P}(M, X_1)$, and $M_2 = \mathcal{P}(M, X_2)$.

To further motivate the notion of projection, we quote here a lemma, on the structural similarity between an MEC M and its projection M', which turns out to be important for our proofs given in the supplementary material.

Lemma 12. *Let G be a graph, $X \subseteq V_G$, $M \in \text{MEC}(G)$, and $\mathcal{P}(M, X) = M'$. If $u \to v \in M'$, then $u \to v \in M$.*

The structural resemblance between an MEC and its projection further implies a relationship between their shadows. This relationship will be used to recursively compute $|\text{MEC}(H, O, P_1, P_2)|$ for any shadow $(O, P_1, P_2) \in \text{shadow}(H[S_1 \cup N(S_1, H)])$. To establish this relationship, we provide the following definition.

Definition 13 (Derived Path Function)**.** Let $(O, P_1, P_2) \in \text{shadow}(H[S_1 \cup S_2 \cup N(S_1 \cup S_2, H)])$, and for $a \in \{1, 2\}$, $(O_a, P_{a1}, P_{a2}) \in \text{shadow}(H_a[S_a \cup N(S_a, H_a)])$. We say (P_1, P_2) as the *derived path function* of $(O, P_{11}, P_{12}, P_{21}, P_{22})$, denoted as $(P_1, P_2) = \text{DPF}(O, P_{11}, P_{12}, P_{21}, P_{22})$, if P_1 and P_2 are computed through the following steps:

1. Step 1 (Initialization):
   (a) $\forall((u, v), (x, y)) \in E_O \times E_O$, if $(u, v) \neq (x, y)$ and there exists a triangle-free path from (u, v) to (x, y) in O, then $P_1((u, v), (x, y)) = 1$, otherwise $P_1((u, v), (x, y)) = 0$.
   (b) $\forall((u, v), w) \in E_O \times V_O$, if $v \neq w$ and there exists a triangle-free path from (u, v) to w in O, then $P_2((u, v), w) = 1$, otherwise $P_2((u, v), w) = 0$.
2. Step 2 (Update 1):
   (a) $\forall((u, v), (x, y)) \in E_O \times E_O$ such that $(u, v) \neq (x, y)$ and $P_1((u, v), (x, y)) = 0$, if for some $a \in \{1, 2\}$, $P_{a1}((u, v), (x, y)) = 1$, then update P_1 with $P_1((u, v), (x, y)) = 1$.
   (b) $\forall((u, v), w) \in E_O \times V_O$ such that $v \neq w$ and $P_2((u, v), w) = 0$, if for any $a \in \{1, 2\}$, $P_{a2}((u, v), w) = 1$, then update P_2 with $P_2((u, v), w) = 1$.
3. Step 3 (Update 2: adding transitivity):
   (a) $\forall((u, v), (x, y)) \in E_O \times E_O$ such that $(u, v) \neq (x, y)$ and $P_1((u, v), (x, y)) = 0$, if there exists $(z_1, z_2) \in E_O$ such that $P_1((u, v), (z_1, z_2)) = P_1((z_1, z_2), (x, y)) = 1$, then update P_1 with $P_1((u, v), (x, y)) = 1$.
   (b) $\forall((u, v), w) \in E_O \times V_O$, such that $v \neq w$ and $P_2((u, v), w) = 0$, if there exists $(z_1, z_2) \in E_O$ such that $P_1((u, v), (z_1, z_2)) = P_2((z_1, z_2), w) = 1$, then update P_2 with $P_2((u, v), w) = 1$.

We say that a derived path function (P_1, P_2) of $(O, P_{11}, P_{12}, P_{21}, P_{22})$ is a *valid derived path function* if for all $(u, v), (x, y) \in E_O$, if $(u, v) \neq (x, y)$ and $P_1((u, v), (x, y)) = 1$, then $P_1((x, y), (u, v)) = 0$.

We now establish a relationship between shadows of H, H_1 and H_2.

Definition 14 (Extension)**.** For $a \in \{1, 2\}$, let $(O_a, P_{a1}, P_{a2}) \in \text{shadow}(H_a[S_a \cup N(S_a, H_a)])$. We define $O \in PMEC(H[S_1 \cup S_2 \cup N(S_1 \cup S_2, H)])$ to be an *extension* of (O_1, P_{11}, P_{12}) and (O_2, P_{21}, P_{22}), denoted as $O \in \mathcal{E}(O_1, P_{11}, P_{12}, O_2, P_{21}, P_{22})$, if

1. For $a \in \{1, 2\}$, for $u, v \in S_a \cup N(S_a, H_a)$, if $u \to v \in O_a$, then $u \to v \in O$.
2. For $a \in \{1, 2\}$, $\mathcal{V}(O_a) = \mathcal{V}(O[S_a \cup N(S_a, H_a)])$.
3. For $a \in \{1, 2\}$, for $u - v \in O_a$, $u \to v \in O$ if, and only if, at least one of the following occurs:
   (a) $u \to v$ is strongly protected in O.
   (b) $\exists x - y \in O_a$ such that $x \to y \in O$, and $P_{a1}((x, y), (u, v)) = 1$.
   (c) $\exists x - y \in O_a$ such that $x \to y \in O$, $P_{a2}((x, y), v) = 1$, and $P_{a2}((v, u), x) = 1$.
4. $\text{DPF}(O, P_{11}, P_{12}, P_{21}, P_{22})$ is valid.

With a slight abuse of notation, we say that (O, P_1, P_2) is an extension of $(O_1, P_{11}, P_{12}, O_2, P_{21}, P_{22})$, denoted as $(O, P_1, P_2) \in \mathcal{E}(O_1, P_{11}, P_{12}, O_2, P_{21}, P_{22})$, if

1. O satisfies items 1 to 4 of Definition 14, and

2. $(P_1, P_2) = \text{DPF}(O, P_{11}, P_{12}, P_{21}, P_{22})$.

The following two lemmas, Lemmas 15 and 16, establish a method to count $|\text{MEC}(H, O, P_1, P_2)|$ for any shadow $(O, P_1, P_2) \in \text{shadow}(H[S_1 \cup S_2 \cup N(S_1 \cup S_2, H)])$ when we have knowledge of $|\text{MEC}(H_1, O_1, P_{11}, P_{12})|$ and $|\text{MEC}(H_2, O_2, P_{21}, P_{22})|$ for all shadows $(O_1, P_{11}, P_{12}) \in \text{shadow}(H_1[S_1 \cup N(S_1, H_1)])$ and $(O_2, P_{21}, P_{22}) \in \text{shadow}(H_2[S_2 \cup N(S_2, H_2)])$.

Lemma 15. *Let M, M_1, and M_2 be MECs of H, H_1, and H_2, respectively. Let (O, P_1, P_2) be the* shadow *of M on $S_1 \cup S_2 \cup N(S_1 \cup S_2, H)$, (O_1, P_{11}, P_{12}) be the* shadow *of M_1 on $S_1 \cup N(S_1, H_1)$, and (O_2, P_{21}, P_{22}) be the* shadow *of M_2 on $S_2 \cup N(S_2, H_2)$. If $\mathcal{P}(M, V_{H_1}, V_{H_2}) = (M_1, M_2)$, then $(O, P_1, P_2) \in \mathcal{E}(O_1, P_{11}, P_{12}, O_2, P_{21}, P_{22})$*

The definitions of extensions and derived path function constitute the main conceptual ingredient in the proof of Lemma 15. Given these definitions, the proof is based on a somewhat technical structural induction, that uses ideas that have appeared previously in works on Markov equivalence (Chickering 1995). The detailed proof is provided in the supplementary material.

Lemma 16. *Let M_1 be an MEC of H_1, and M_2 be an MEC of H_2. Let (O_1, P_{11}, P_{12}) be the shadow of M_1 on $S_1 \cup N(S_1, H_1)$, and (O_2, P_{21}, P_{22}) be the shadow of M_2 on $S_2 \cup N(S_2, H_2)$. Let $(O, P_1, P_2) \in \text{shadow}(H[S_1 \cup S_2 \cup N(S_1 \cup S_2, H)])$. If (O, P_1, P_2) is an extension of $(O_1, P_{11}, P_{12}, O_2, P_{21}, P_{22})$, then there exists a unique MEC M of H such that $(O, P_1, P_2) = \text{shadow}(M, S_1 \cup S_2 \cup N(S_1 \cup S_2, H))$, and $\mathcal{P}(M, V_{H_1}, V_{H_2}) = (M_1, M_2)$.*

The proof of this lemma is constructive: in order to show the existence of M, we give an algorithmic process for constructing M given the other data in the lemma (the claim of uniqueness is then relatively easier to establish). At the heart of this algorithmic process is a modified version of the celebrated *lexicographical breadth first search algorithm* (LBFS) of Rose, Tarjan, and Lueker (1976). Interestingly, a different modified version of the LBFS algorithm was also important in the algorithm of Wienöbst, Bannach, and Liśkiewicz (2021) for computing *sizes* of MECs. Our modification, however, is of a different character. While detailed proof and description are provided in the supplementary material, we present here a brief summary.

The usual LBFS algorithm proceeds like a usual BFS, except that it progressively constructs a "lexicographical" partial order on the set of vertices, and each step chooses one of the vertices that are minimal in this order. In particular, it can choose *any* such vertex. For our purposes, we need to further specify the vertex we choose at each step: in particular, we isolate the notion of a so-called *canonical source vertex (CSV)* (see the supplementary material for a definition), and show that, in the context in which we use the LBFS algorithm, the vertex that the LBFS algorithm chooses can always be taken to be such a CSV. The definition of a CSV, and the fact that we can always find such a CSV for the LBFS algorithm to choose in the context in which we use the algorithm, both turn out to be crucial in the proof of Lemma 16 given in the supplementary material.

Lemmas 15 and 16 together give a method for computing $|\text{MEC}(H, O, P_1, P_2)|$ for all $(O, P_1, P_2) \in \text{shadow}(H[S_1 \cup S_2 \cup N(S_1 \cup S_2, H)])$

Lemma 17. *Let $(O, P_1, P_2) \in \text{shadow}(H[S_1 \cup S_2 \cup N(S_1 \cup S_2, H)])$. Then, $|\text{MEC}(H, O, P_1, P_2)| = \sum |\text{MEC}(H_1, O_1, P_{11}, P_{12})| \times |\text{MEC}(H_2, O_2, P_{21}, P_{22})|$, where the summation runs over all $(O_1, P_{11}, P_{12}) \in \text{shadow}(H_1[S_1 \cup N(S_1, H_1)])$ and $(O_2, P_{21}, P_{22}) \in \text{shadow}(H_2[S_2 \cup N(S_2, H_2)])$ such that $(O, P_1, P_2) \in \mathcal{E}(O_1, P_{11}, P_{12}, O_2, P_{21}, P_{22})$.*

We now recall the discussion in the paragraphs preceding Definition 11. As discussed there, H resembles G_i^j, H_1 resembles G_i^{j-1}, H_2 resembles G_{i_j}, and S_1 and S_2 resemble X_i and X_{i_j}. Lemma 17 implies that if for all shadows $(O_1, P_{11}, P_{12}) \in \text{shadow}(G_i^{j-1}[X_i \cup N(X_i, G_i^{j-1})])$, and $(O_2, P_{21}, P_{22}) \in \text{shadow}(G_{i_j}[X_{i_j} \cup N(X_{i_j}, G_{i_j})])$, we have the knowledge about $|\text{MEC}(G_i^{j-1}, O_1, P_{11}, P_{12})|$ and $|\text{MEC}(G_{i_j}, O_2, P_{21}, P_{22})|$, then for all $(O, P_1, P_2) \in \text{shadow}(G_i^j[X_i \cup X_{i_j} \cup N(X_i \cup X_{i_j}, G_i^j)])$, we can compute $|\text{MEC}(G_i^j, O, P_1, P_2)|$. But for the recursion, we need $|\text{MEC}(H, O', P_1', P_2')|$ for all $(O', P_1', P_2') \in \text{shadow}(G_i^j[X_i \cup N(X_i, G_i^j)])$. As (O, P_1, P_2) contains more information than (O', P_1', P_2'), we define the projection of (O, P_1, P_2) to get $|\text{MEC}(H, O', P_1', P_2')|$ for each shadow (O', P_1', P_2').

Definition 18 (Projection of shadow (O', P_1', P_2'))**.** Let H be an undirected graph. S and S' are two vertex subsets of H such that $S' \subseteq S \subseteq V_H$. Let $(O, P_1, P_2) \in \text{shadow}(G[S])$ and $(O', P_1', P_2') \in \text{shadow}(G[S'])$. We say (O', P_1', P_2') is a projection of (O, P_1, P_2) on S' denoted as $\mathcal{P}(O, P_1, P_2, S') = (O', P_1', P_2')$ if

1. $O[S'] = O'$,
2. for $((u, v), (x, y)) \in E_{G[S']} \times E_{G[S']}$, $P_1'((u, v), (x, y)) = P_1((u, v), (x, y))$, and
3. for $((u, v), w) \in E_{G[S']} \times V_{G[S']}$, $P_2'((u, v), w) = P_2((u, v), w)$.

We now compute $|\text{MEC}(H, O', P_1', P_2')|$.

Lemma 19. *Let H be an undirected graph, S and S' are two vertex subsets of H such that $S' \subseteq S \subseteq V_H$. Let $(O', P_1', P_2') \in \text{shadow}(H[S'])$. Then, $|\text{MEC}(H, O', P_1', P_2')|$ equals $\sum |\text{MEC}(H, O, P_1, P_2))|$, where the sum runs over all $(O, P_1, P_2) \in \text{shadow}(H[S])$ for which $(O', P_1', P_2') = \mathcal{P}(O, P_1, P_2, S')$.*

Combining Lemmas 17 and 19, we obtain the following lemma:

Lemma 20. *For any shadow $(O', P_1', P_2') \in \text{shadow}(H[S_1 \cup N(S_1, H)])$, $|\text{MEC}(H, O', P_1', P_2')|$ equals $\sum |\text{MEC}(H_1, O_1, P_{11}, P_{12})| \times |\text{MEC}(H_2, O_2, P_{21}, P_{22})|$, where the summation runs over all $(O, P_1, P_2) \in \text{shadow}(H[S_1 \cup S_2 \cup N(S_1 \cup S_2, H)])$, $(O_1, P_{11}, P_{12}) \in \text{shadow}(H_1[S_1 \cup N(S_1, H_1)])$, and $(O_2, P_{21}, P_{22}) \in \text{shadow}(H_2[S_2 \cup N(S_2, H_2)])$ such that $(O, P_1, P_2) \in \mathcal{E}(O_1, P_{11}, P_{12}, O_2, P_{21}, P_{22})$ and $(O', P_1', P_2') = \mathcal{P}(O, P_1, P_2, S')$.*

Using Lemma 20, we can compute $|\text{MEC}(G_i^j, O', P_1', P_2')|$ for any partial MEC $(O', P_1', P_2') \in \text{shadow}(G_i^j[X_i \cup N(X_i, G_i^j)])$. This provides us a recursive method to compute the MECs of G.

We now summarise the above solution. To count $|\text{MEC}(G)|$, we first partition $\text{MEC}(G)$ based on the shadows $(O, P_1, P_2) \in \text{shadow}(G[Y])$ for some $Y \in V_G$. From Lemma 10, this reduces our goal to computing $|\text{MEC}(G, O, P_1, P_2)|$ for all the shadows $(O, P_1, P_2) \in \text{shadow}(G[Y])$ for some $Y \in V_G$. To achieve this, we recursively partition G using its tree decomposition, as shown in fig. 3. For the base case, G_i^0, we compute $|\text{MEC}(G_i^0, O, P_1, P_2)|$ for all $(O, P_1, P_2) \in \text{shadow}(G_i^0 = G_i^0[X_i \cup N(X_i, G_i)])$ using the brute force method. We then recursively compute $|\text{MEC}(G_i^j, O', P_1', P_2')|$ for all shadows $(O', P_1', P_2') \in \text{shadow}(G_i^j[X_i \cup N(X_i, G_i^j)])$ using Lemma 20. As $G = G_1^p$, after computing $|\text{MEC}(G_i^p, O, P_1, P_2)|$ for all shadows $(O, P_1, P_2) \in \text{shadow}(G_1^p[X_1 \cup N(X_1, G_1^p)])$, we implement Lemma 10 to compute $|\text{MEC}(G)|$.

Time Complexity To compute $|\text{MEC}(G)|$, we first construct a tree decomposition of G. Let k be the treewidth of G. We use Korhonen (2022)'s algorithm, which constructs a tree decomposition $(\mathcal{X} = \{X_1, X_2, \ldots, X_l\}, T)$ of G of width d where $k \leq d \leq 2k+1$ in time $2^{O(k)} \cdot n$. We need to implement the brute force method for the base case, $G_i^0 = G[X_i]$, for each $X_i \in \mathcal{X}$. Since the number of nodes in a tree decomposition of a graph with n nodes is $O(n)$, therefore, the number of times we have to call the brute force method is $O(n)$. In the base case, G_i^0, $V_{G_i^0} = X_i$. Since the width of the tree decomposition is d, therefore, $|V_{G_i^0}|$ can be at most $d+1$, and $|E_{G_i^0}| = O(d^2)$. For any edge $u - v \in E_{G_i^0}$, in a partial MEC $O \in \text{PMEC}(G_i^0)$, there are three possibilities, either $u \to v \in O$ or $v \to u \in O$ or $u - v \in O$. This implies the number of partial MEC of G_i^j is $O(3^{O(d^2)})$. For each $O \in \text{PMEC}(G_i^0)$, the number of distinct functions $P_1 : E_O \to E_O \to \{0,1\}$ and $P_2 : E_O \to V_O \to \{0,1\}$ is $O(2^{O(d^4)})$. Thus, the number of shadows of G_i^0 is $O(2^{O(d^4)})$. From Definition 2, each MEC is itself a partial MEC. Therefore, in the base case, for any shadow $(O, P_1, P_2) \in \text{shadow}(G_i^0)$, if $M \in \text{MEC}(G_i^0, O, P_1, P_2)$ then $O = M$ and for any $(u,v), (x,y) \in E_O$, $P_1((u,v),(x,y)) = 1$ if there exists a triangle-free path from (u,v) to (x,y) in O else $P_1((u,v),(x,y)) = 0$. Verification of whether a partial MEC is an MEC or not takes $O(d^2)$ time, and verification whether P_1 and P_2 obey the partial MEC O or not takes $O(d^{10})$ time (details are given in supplementary material). Therefore, computing $|\text{MEC}(G_i^0, O, P_1, P_2)|$ for all $(O, P_1, P_2) \in \text{shadow}(G_i^0)$ takes time $O(2^{O(d^4)} d^{10})$. As discussed earlier, $|\mathcal{X}| = O(n)$. Therfore, solving base cases requires time $O(2^{O(d^4)} d^{10} n)$.

For a recursive call, we cut an edge. As the number of edges in the tree decomposition is $O(n)$, therefore, the number of times recursion occurs is $O(n)$. In each recursive call, for G_i^j, we partition G_i^j into G_i^{j-1} and G_{i_j}. The partitioning of G_i^j takes $O(n^2)$ time. The number of nodes in $X_i \cup X_{i_j} \cup N(X_i \cup X_{i_j}, G_i^j)$ can be at most $(d+1)(\delta+1)$, where δ is the degree of G. Thus, the number of shadows in $\text{shadow}(G_i^j[X_i \cup X_{i_j} \cup N(X_i \cup X_{i_j}, G_i^j)])$ is $O(2^{O(d^4\delta^4)})$ (similar to the base case). Finding the value of $|\text{MEC}(G_i^j, O, P_1, P_2)$ for each shadow $(O, P_1, P_2) \in \text{shadow}(G_i^j[X_i \cup N(X_i \cup X_{i_j})])$ using Lemma 20 takes the same time $O(2^{O(d^4\delta^4)})$. Therefore, the time complexity of the whole process is $O(n(2^{O(d^4\delta^4)} + n^2))$. Since $d = O(k)$, therefore, the time complexity of our algorithm is $O(n(2^{O(k^4\delta^4)} + n^2))$, where $n = V_G$, and k and δ are the treewidth and the degree of G, respectively.

Conclusion and Open Problems

We provide a fixed parameter tractable algorithm with runtime $O(n(2^{O(k^4\delta^4)} + n^2))$ for the problem of counting the number of MECs with a given skeleton, where n is the number of nodes in the input graph, and k and δ are the treewidth and the degree of the input graph, respectively.

The main problem left open by this work is to either provide a fully polynomial time algorithm for this problem or else to prove that it is computationally hard in general. We note that our analysis can be strengthened in certain very special cases to provide a much better running time. In particular, it seems that (with some modifications to the algorithm and the analysis) our method can be made to run in polynomial time when G is a tree graph. Similarly, for chordal graphs, it seems that the dependence on the treewidth can be made much better.

An intermediate open problem towards the goal of understanding the complexity fully is to improve the runtime of the fixed-parameter tractable algorithm itself: perhaps using different parameters.

As stated in the introduction, we consider our results to be a first step towards understanding the computational complexity of this problem. However, we hope that the definitions and techniques introduced in this paper will be helpful in further study of the problem.

Acknowledgments

We acknowledge the support from the Department of Atomic Energy, Government of India, under project no. RTI4001. We want to thank Piyush Srivastava for his invaluable suggestions, discussions, and help in the completion of this paper. We thank all the anonymous reviewers for multiple useful suggestions which helped in improving the presentation of this paper.

References

Andersson, S. A.; Madigan, D.; and Perlman, M. D. 1997. A Characterization of Markov Equivalence Classes for Acyclic Digraphs. *Annals of Statistics*, 25(2): 505–541.

Blair, J. R. S.; and Peyton, B. 1993. An introduction to Chordal Graphs and Clique Trees. In *Graph Theory and Sparse Matrix Computation*, 1–29. Springer.

Chickering, D. M. 1995. A Transformational Characterization of Equivalent Bayesian Network Structures. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI 1995)*, 87–98.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer International Publishing.

Diestel, R. 2005. Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33): 12.

Ghassami, A.; Salehkaleybar, S.; Kiyavash, N.; and Zhang, K. 2019. Counting and sampling from Markov equivalent DAGs using clique trees. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3664–3671.

Gillispie, S. B. 2006. Formulas for counting acyclic digraph Markov equivalence classes. *Journal of Statistical Planning and Inference*, 136(4): 1410–1432.

Gillispie, S. B.; and Perlman, M. D. 2001. Enumerating Markov equivalence classes of acyclic digraph models. *arXiv preprint arXiv:1301.2272*.

Gillispie, S. B.; and Perlman, M. D. 2002. The Size Distribution for Markov Equivalence Classes of Acyclic Digraph Models. *Artificial Intelligence*, 141(1-2): 137–155.

He, Y.; Jia, J.; and Yu, B. 2013. Reversible MCMC on Markov equivalence classes of sparse directed acyclic graphs. *The Annals of Statistics*, 41(4): 1742–1779. Publisher: Institute of Mathematical Statistics.

Korhonen, T. 2022. A single-exponential time 2-approximation algorithm for treewidth. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 184–192. IEEE.

Meek, C. 1995. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 403–410.

Radhakrishnan, A.; Solus, L.; and Uhler, C. 2016. Counting Markov Equivalence Classes by Number of Immoralities. *arXiv:1611.07493*.

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

Rose, D. J.; Tarjan, R. E.; and Lueker, G. S. 1976. Algorithmic Aspects of Vertex Elimination on Graphs. *SIAM Journal on Computing*, 5(2): 266–283.

Schmid, D.; and Sly, A. 2022. On the number and size of Markov equivalence classes of random directed acyclic graphs. *arXiv preprint arXiv:2209.04395*.

Steinsky, B. 2003. Enumeration of labelled chain graphs and labelled essential directed acyclic graphs. *Discrete mathematics*, 270(1-3): 267–278.

Talvitie, T.; and Koivisto, M. 2019. Counting and Sampling Markov Equivalent Directed Acyclic Graphs. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, volume 33, 7984–7991.

Van Der Zander, B.; and Liskiewicz, M. 2016. Separators and adjustment sets in Markov equivalent DAGs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1.

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

Wienöbst, M.; Bannach, M.; and Liśkiewicz, M. 2021. Polynomial-Time Algorithms for Counting and Sampling Markov Equivalent DAGs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*, 12198–12206.