Counting and Sampling from Markov Equivalent DAGs Using Clique Trees

AmirEmad Ghassami,[1] Saber Salehkaleybar,[2] Negar Kiyavash,[3] Kun Zhang[4]
[1]Department of ECE, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[2]Electrical Engineering Department, Sharif University of Technology, Tehran, Iran
[3]Departments of ECE and ISE, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[4]Department of Philosophy, Carnegie Mellon University, Pittsburgh, USA
[1]ghassam2@illinois.edu, [2]saleh@sharif.edu, [3]kiyavash@illinois.edu, [4]kunz1@cmu.edu

Abstract

A directed acyclic graph (DAG) is the most common graphical model for representing causal relationships among a set of variables. When restricted to using only observational data, the structure of the ground truth DAG is identifiable only up to Markov equivalence, based on conditional independence relations among the variables. Therefore, the number of DAGs equivalent to the ground truth DAG is an indicator of the causal complexity of the underlying structure–roughly speaking, it shows how many interventions or how much additional information is further needed to recover the underlying DAG. In this paper, we propose a new technique for counting the number of DAGs in a Markov equivalence class. Our approach is based on the clique tree representation of chordal graphs. We show that in the case of bounded degree graphs, the proposed algorithm is polynomial time. We further demonstrate that this technique can be utilized for uniform sampling from a Markov equivalence class, which provides a stochastic way to enumerate DAGs in the equivalence class and may be needed for finding the best DAG or for causal inference given the equivalence class as input. We also extend our counting and sampling method to the case where prior knowledge about the underlying DAG is available, and present applications of this extension in causal experiment design and estimating the causal effect of joint interventions.

1 Introduction

A directed acyclic graph (DAG) is a commonly used graphical model to represent causal relationships among a set of variables (Pearl 2009). In a DAG representation, a directed edge $X \rightarrow Y$ indicates that variable X is a direct cause of variable Y relative to the considered variable set. Such a representation has numerous applications in fields ranging from biology (Sachs et al. 2005) and genetics (Zhang et al. 2013) to machine learning (Peters, Janzing, and Schölkopf 2017; Koller and Friedman 2009).

The general approach to learning a causal structure is to use statistical data from variables to find a DAG, which is maximally consistent with the conditional independencies estimated from data. This is due to the fact that under Markov property and faithfulness assumption, d-separation of variables in a DAG is equivalent to conditional independencies of the variables in the underlying joint probability distribution (Spirtes, Glymour, and Scheines 2000; Pearl 2009). However, a DAG representation of a set of conditional independencies in most cases is not unique. This restricts the learning of the structure to the Markov equivalences of the underlying DAG. The set of Markov equivalent DAGs is referred to as a Markov equivalence class (MEC). A MEC is commonly represented by a mixed graph called essential graph, which can contain both directed and undirected edges (Spirtes, Glymour, and Scheines 2000).

In general, there is no preference amongst the elements of MEC, as they all represent the same set of conditional independencies. Therefore, for a given dataset (or a joint probability distribution), the size of the MEC, i.e., the number of its elements, can be seen as a metric for the causal complexity of the underlying structure–this complexity indicates how many interventional experiments or how much additional information (e.g., knowledge about the causal mechanisms) is needed to further fully learn the DAG structure, which was only partially recovered from mere observation.

For the general problem of enumerating MECs, Steinsky proposed recursive enumeration formulas for the number of labelled essential graphs, in which the enumeration parameters are the number of vertices, chain components, and cliques (Steinsky 2013). This approach is not focused on a certain given MEC. For the problem of finding the size of a given MEC, an existing solution is to use Markov chain methods (He et al. 2013; Bernstein and Tetali 2017). According to this method, a Markov chain is constructed over the elements of the MEC whose properties ensure that the stationary distribution is uniform over all the elements. The rate of convergence and computational issues hinders practical application of Markov chain methods. Recently, an exact solution for finding the size of a given MEC was proposed (He, Jia, and Yu 2015), in which the main idea was noting that subclasses of the MEC with a fixed unique root variable partition the class. The authors show that there are five types of MECs whose sizes can be calculated with five formulas, and for any other MEC, it can be partitioned recursively into smaller subclasses until the sizes of all subclasses can be calculated from the five formulas. An accelerated version of the method in (He, Jia, and Yu 2015) is proposed in (He and Yu 2016), which is based on the concept of core graphs.

In this paper, we propose a new counting approach, in which the counting is performed on the clique tree representation of the graph. Compared to (He, Jia, and Yu 2015), our method provides us with a more systematic way for finding the orientation of the edges in a rooted subclass, and enables us to use the memory in an efficient way in the implementation of the algorithm. Also, using clique tree representation enables us to divide the graph into smaller pieces, and perform counting on each piece separately (Theorem 2). We will show that for bounded degree graphs, the proposed solution is capable of computing the size of the MEC in polynomial time. The counting technique can be utilized for two main goals: **(a) Uniform sampling**, and **(b) Applying prior knowledge**. As will be explained, for these goals, it is essential in our approach to have the ability of explicitly controlling the performed orientations in the given essential graph. Therefore, neither the aforementioned five formulas presented in (He, Jia, and Yu 2015), nor the accelerated technique in (He and Yu 2016) are suitable for these purposes.

(a) Uniform sampling: In Section 4, we show that our counting technique can be used to uniformly sample from a given MEC. This can be utilized in many scenarios; followings are two examples. 1. *Evaluating effect of an action in a causal system.* For instance, in a network of users, one may be interested in finding out conveying a news to which user leads to the maximum spread of the news. This question could be answered if the causal structure was known, but we often do not have the exact causal DAG. Instead, we can resort to uniformly sampling from the corresponding MEC and evaluate the effect of the action on the samples. 2. Given a DAG from a MEC, there are simple algorithms for generating the essential graph corresponding to the MEC (Meek 1995; Andersson, Madigan, and Perlman 1997; Chickering 2002). However, for MECs with large size, it is not computationally feasible to form every DAG represented by the essential graph. In (Hoyer and Hyttinen 2009) the authors require to evaluate the score of all DAGs in MEC to find the one with the highest score. Evaluating scores on uniform samples provides an estimate of the maximum score.

(b) Applying prior knowledge: In Section 5, we further extend our counting and sampling techniques to the case that prior knowledge regarding the orientation of a subset of the edges in the structure is available. Such prior knowledge may come from imposing a partial ordering of variables before conducting causal structure learning from both observational and interventional data (Scheines et al. 1998; Hauser and Bühlmann 2012; Wang et al. 2017), or from certain model restrictions (Hoyer et al. 2012; Rothenhäusler, Ernest, and Bühlmann 2018; Eigenmann, Nandy, and Maathuis 2017). We will show that rooted subclasses of the MEC allow us to understand whether a set of edges with unknown directions can be oriented to agree with the prior, and if so, how many DAGs in the MEC are consistent with such an orientation. We present the following two applications: 1. The authors in (Nandy, Maathuis, and Richardson 2017) proposed a method for estimating the causal effect of joint interventions from observational data. Their method requires extracting possible valid parent sets of the intervention nodes from the essential graph, along with their multiplicity. They proposed the joint-IDA method for this goal, which is exponential in the size of the chain component of the essential graph, and hence, becomes infeasible for large components. In Section 5, we show that counting with prior knowledge can solve the issue of extracting possible parent sets. 2. In Section 6, we provide another specific scenario in which our proposed methodology is useful. This application is concerned with finding the best set of variables to intervene on when we are restricted to a certain budget for the number of interventions (Ghassami et al. 2018a).

2 Definitions and Problem Description

For the definitions in this section, we mainly follow (Andersson, Madigan, and Perlman 1997). A graph G is a pair $(V(G), E(G))$, where $V(G)$ is a finite set of vertices and $E(G)$, the set of edges, is a subset of $(V \times V) \setminus \{(a, a) : a \in V\}$. An edge $(a, b) \in E$ whose opposite $(b, a) \in E$ is called an undirected edge and we write $a - b \in G$. An edge $(a, b) \in E$ whose opposite $(b, a) \notin E$ is called a directed edge, and we write $a \to b \in G$. A graph is called a chain graph if it contains no *partially* directed cycles. After removing all directed edges of a chain graph, the components of the remaining undirected graph are called the chain components of the chain graph. A v-structure of G is a triple (a, b, c), with induced subgraph $a \to b \leftarrow c$. Under Markov condition and faithfulness assumptions, a directed acyclic graph (DAG) represents the conditional independences of a distribution on variables corresponding to its vertices. Two DAGs are called Markov equivalent if they represent the same set of conditional independence relations. The following result due to (Verma and Pearl 1990) provides a graphical test for Markov equivalence.

Lemma 1. *(Verma and Pearl 1990) Two DAGs are Markov equivalent iff they have the same skeleton and v-structures.*

We denote the Markov equivalence class (MEC) containing DAG D by $[D]$. A MEC can be represented by a graph G^*, called essential graph, which is defined as $G^* = \cup(D : D \in [D])$. We denote the MEC corresponding to essential graph G^* by $MEC(G^*)$. Essential graphs are also referred to as completed partially directed acyclic graphs (CPDAGs) (Chickering 2002), and maximally oriented graphs (Meek 1995). Authors in (Andersson, Madigan, and Perlman 1997) proposed a graphical criterion for characterizing an essential graph. They showed that an essential graph is a chain graph in which every chain component is chordal. As a corollary of Lemma 1, no DAG in a MEC can contain a v-structure in the subgraphs corresponding to chain components.

We refer to the number of elements of a MEC as the size of the MEC, and we denote the size of $MEC(G^*)$ by $Size(G^*)$. Let $\{G_1, ..., G_c\}$ denote the chain components of G^*. $Size(G^*)$ can be calculated from the size of chain components using the following equation (Gillispie and Perlman 2002; He and Geng 2008):

$$Size(G^*) = \prod_{i=1}^{c} Size(G_i). \tag{1}$$

Therefore, counting can be done separately in each chain component. Hence, without loss of generality, we can focus on the problem of finding the size of a MEC for which the essential graph is a chain component, i.e., an undirected connected chordal graph (UCCG), in which none of the members of the MEC are allowed to contain any v-structures.

In order to solve this problem, the authors of (He, Jia, and Yu 2015) showed that there are five types of MECs whose sizes can be calculated with five formulas, and that for any other MEC, it can be partitioned recursively into smaller subclasses until the sizes of all subclasses can be calculated from the five formulas. In the following we explain the partitioning method as it is relevant to this work as well. Let G be a UCCG and D be a DAG in *MEC*(G). A vertex $v \in V(G)$ is the root in D if its in-degree is zero.

Definition 1. *Let G be a UCCG. The v-rooted subclass of MEC(G) is the set of all v-rooted DAGs in MEC(G). This subclass can be represented by the v-rooted essential graph $G^{(v)} = \cup(D : D \in v\text{-rooted subclass})$.*

For instance, for UCCG G in Figure $1(a)$, $G^{(v_1)}$ and $G^{(v_2)}$ are depicted in Figures $1(c)$ and $1(g)$, respectively.

Lemma 2. *(He, Jia, and Yu 2015) Let G be a UCCG. For any $v \in V(G)$, the v-rooted subclass is not empty and the set of all v-rooted subclasses partitions MEC(G).*

From Lemma 2 we have

$$\mathit{Size}(G) = \sum_{v \in V(G)} \mathit{Size}(G^{(v)}). \qquad (2)$$

Therefore, using equations (1) and (2), we have $\mathit{Size}(G^*) = \prod_{i=1}^{c} \sum_{v \in V(G_i)} \mathit{Size}(G_i^{(v)})$. He et al. showed that $G^{(v)}$ is a chain graph with chordal chain components, and introduced an algorithm for generating this essential graph. Therefore, using the last equation, $\mathit{Size}(G^*)$ can be obtained recursively. The authors of (He, Jia, and Yu 2015) did not characterize the complexity, but reported that their experiments suggested that when the number of vertices is small or the graph is sparse, the proposed approach is efficient.

3 Calculating the size of a MEC

In this section, we present our proposed method for calculating the size of the MEC corresponding to a given UCCG. We first introduce some machinery required in our approach for representing chordal graphs via clique trees. The definitions and propositions are mostly sourced from (Blair and Peyton 1993; Vandenberghe and Andersen 2015).

For a given UCCG, G, let $\mathcal{K}_G = \{K_1, \cdots, K_m\}$ denote the set containing the maximal cliques of G, and let $T = (\mathcal{K}_G, E(T))$ be a tree on $\mathcal{K}_G$, referred to as a *clique tree*.

Definition 2 (*Clique-intersection property*)**.** *A clique tree T satisfies the clique-intersection property if for every pair of distinct cliques $K, K' \in \mathcal{K}_G$, the set $K \cap K'$ is contained in every clique on the path connecting K and K' in the tree.*

Proposition 1. *(Blair and Peyton 1993) A connected graph G is chordal if and only if there exists a clique tree for G, which satisfies the clique-intersection property.*

Definition 3 (*Induced-subtree property*)**.** *A clique tree $T = (\mathcal{K}_G, E(T))$ satisfies the induced-subtree property if for every vertex $v \in V(G)$, the set of cliques containing v induces a subtree of T, denoted by T_v.*

Proposition 2. *(Blair and Peyton 1993) The clique-intersection and induced-subtree properties are equivalent.*

Since we work with a UCCG, by Propositions 1 and 2, there exists a clique tree on $\mathcal{K}_G$, which satisfies the clique-intersection and induced-subtree properties. Efficient algorithms for generating such a tree is presented in (Blair and Peyton 1993; Vandenberghe and Andersen 2015). In the sequel, whenever we refer to clique tree $T = (\mathcal{K}_G, E(T))$, we assume it satisfies the clique-intersection and induced-subtree properties. Note that such a tree is not necessarily unique, yet we fix one and perform all operations on it. For the given UCCG, similar to (He, Jia, and Yu 2015), we partition the corresponding MEC by its rooted subclasses, and calculate the size of each subclass separately. However, we use the clique tree representation of the UCCG for counting. Our approach enables us to use the memory in the counting process to make the counting more efficient, and provides us with a systematic approach for finding the orientations.

For the r-rooted essential graph $G^{(r)}$, we arbitrarily choose one of the cliques containing r as the root of the tree $T = (\mathcal{K}_G, E(T))$, and denote this rooted clique tree by $T^{(r)}$. Setting a vertex as the root in a tree determines the parent of all vertices of the tree. For clique K in $T^{(r)}$, denote its parent clique by $\mathit{Pa}(K)$. Following (Vandenberghe and Andersen 2015), we can partition each non-root clique into a separator set $\mathit{Sep}(K) = K \cap \mathit{Pa}(K)$, and a residual set $\mathit{Res}(K) = K \setminus \mathit{Sep}(K)$. For the root clique, the convention is to define $\mathit{Sep}(K) = \emptyset$, and $\mathit{Res}(K) = K$. The induced-subtree property implies the following result.

Proposition 3. *(Vandenberghe and Andersen 2015) Let G be the given UCCG, and let $T^{(r)}$ be the rooted clique tree,*
(i) The clique residuals partition $V(G)$,
(ii) For each $u \in V(G)$, denote the clique for which u is in its residual set by K_u, and the induced subtree of cliques containing u by T_u. K_u is the root of T_u. The other vertices of T_u are the cliques that contain u in their separator set.
(iii) The clique separators Sep(K), where K ranges over all non-root cliques, are the minimal vertex separators of G.

Note that sets $\mathit{Sep}(K)$ and $\mathit{Res}(K)$ depend on the choice of root. However, all clique trees have the same vertices (namely, maximal cliques of G) and the same clique separators (namely, the minimal vertex separators of G). Also, since there are at most p cliques in a chordal graph with p vertices, there are at most $p-1$ edges in a clique tree and hence, at most $p-1$ minimal vertex separators.

With a small deviation from the standard convention, in our approach, for the root clique of $T^{(r)}$, we define $\mathit{Sep}(K) = \{r\}$, and $\mathit{Res}(K) = K \setminus \{r\}$. Recall from Proposition 3 that for each $u \in V(G)$, K_u denotes the clique for which u is in its residual set, and this clique is unique. We will need the following results for our orientation approach. All the proofs are provided in (Ghassami et al. 2018b).

Lemma 3. *If $u \to v \in G^{(r)}$, then $u \in \mathit{Sep}(K_v)$.*

Corollary 1. *If $u \to v \in G^{(r)}$, then $v \notin \mathit{Sep}(K_u)$.*

Lemma 3 states that parents of any vertex v are elements of $\mathit{Sep}(K_v)$. But, not all elements of $\mathit{Sep}(K_v)$ are necessarily

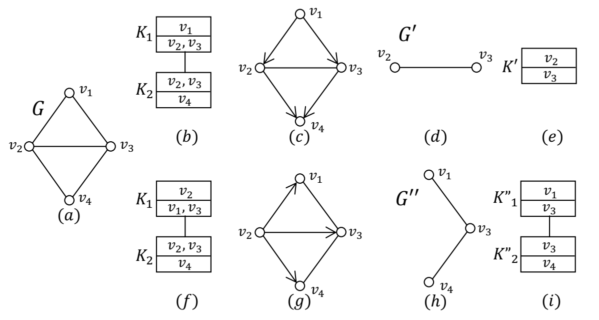

Figure 1: Graphs related to Example 1.

parents of v. Our objective is to find a necessary and sufficient condition to determine the parents of a vertex.

Lemma 4. *If $u \to v \in G^{(r)}$, then for every vertex $w \in$ Res(K_v), $u \to w \in G^{(r)}$.*

Lemma 4 implies that for every clique K, any vertex $u \in$ *Sep*(K) either has directed edges to all elements of *Res*(K) in $G^{(r)}$, or has no directed edges to any of the elements of *Res*(K). For clique K, we define the *emission set*, *Em*(K), as the subset of *Sep*(K), which has directed edges to all elements of *Res*(K) in $G^{(r)}$. Lemmas 3 and 4 lead to the following corollary.

Corollary 2. *Em(K_v) is the set of parents of v in $G^{(r)}$.*

This means that the necessary and sufficient condition for u to be a parent of v is $u \in$ *Em*(K_v). Therefore, for any clique K, we need to characterize its emission set. We will show that vertices in *Em*(K) are the subset of *Sep*(K), which satisfy the following emission condition:

Definition 4 (Emission condition for a vertex)**.** *We say vertex v satisfies the emission condition in clique K if $v \in$ Sep(K), and Em$(K_v) \nsubseteq$ Sep(K).*

As a convention, we assume that the root variable r satisfies the emission condition in all the cliques containing it.

Theorem 1. *$u \to v \in G^{(r)}$ if and only if u satisfies the emission condition in clique K_v in $T^{(r)}$.*

Note that for a clique K, in order to find *Em*(K) using Definition 4, we need to learn the emission set of some cliques on the higher levels of the tree. Hence, the emission sets must be identified on the tree from top to bottom.

After setting vertex r as the root, Theorem 1 allows us to find the orientation of directed edges in $G^{(r)}$ as follows. First, we form $T^{(r)}$. Then, for each $K \in \mathcal{K}_G$, starting from top to bottom of $T^{(r)}$, we identify all vertices which satisfy the emission condition to obtain *Em*(K). Finally, in each clique K, we orient the edges from all the variables in *Em*(K) towards all the variables in *Res*(K).

Example 1. *Assume the UCCG in Figure 1(a) is the given essential graph. Setting vertex v_1 as the root of G (by symmetry, v_4 is similar), the corresponding clique tree $T^{(v_1)}$ is shown in Figure 1(b), where in each clique, the first and the second rows represent the separator and the residual sets, respectively. In this clique tree, we obtain Em$(K_1) = \{v_1\}$, and Em$(K_2) = \{v_2, v_3\}$. Hence, the directed edges are $v_1 \to v_2$, $v_1 \to v_3$, $v_2 \to v_4$, and $v_3 \to v_4$. This results in $G^{(v_1)}$ in Figure 1(c), which is an essential graph with a single chain component G', (Figure 1(d)). Setting vertex v_2 as the root (by symmetry, v_3 is similar), the corresponding clique tree $T'^{(v_2)}$ is shown in Figure 1(e). In this clique tree, Em$(K') = \{v_2\}$ and hence, the directed edge is $v_2 \to v_3$. This results in a directed graph, thus, $Size(G'^{(v_2)}) = 1$. Similarly, $Size(G'^{(v_3)}) = 1$. Therefore, using equation (2), we have $Size(G^{(v_1)}) = Size(G'^{(v_2)}) + Size(G'^{(v_3)}) = 2$. Similarly, we have $Size(G^{(v_4)}) = 2$. With a similar procedure we have $Size(G^{(v_2)}) = Size(G''^{(v_1)}) + Size(G''^{(v_3)}) + Size(G''^{(v_4)}) = 3$. and $Size(G^{(v_3)}) = 3$ (See (Ghassami et al. 2018b) for the detailed explanation). Finally, using equation (2), we obtain that $Size(G) = \sum_i Size(G^{(v_i)}) = 10$.*

Algorithm 1 MEC Size Calculator

```
1:  Input: Essential graph $G^*$, with chain components
2:          $\{G_1, \cdots G_c\}$.
3:  Return: $\prod_{i=1}^{c}$ SIZE$(G_i)$
4:  ________________________________
5:  function SIZE$(G)$
6:      Construct a clique tree $T = (\mathcal{K}_G, E(T))$.
7:      if $T \in$ Memory then
8:          Load $[T, Size_T]$, Return: $Size_T$
9:      else
10:         for $v \in V(G)$ do
11:             Set a clique $K \in T_v$ as the root to form $T^{(v)}$.
12:             $Size_{T^{(v)}} = \mathrm{RS}(G, T^{(v)}, K, Sep(K))$.
13:         end for
14:         Save $[T, \sum_{v \in V} Size_{T^{(v)}}]$, Return $\sum_v Size_{T^{(v)}}$
15:     end if
16: end function
```

3.1 Algorithm

In this subsection, we present an efficient approach for the counting process. In a rooted clique tree $T^{(r)}$, for any clique K, let $T^{(K)}$ be the maximal subtree of $T^{(r)}$ with K as its root, and let *Res*$(T^{(K)}) := \bigcup_{K' \in T^{(K)}}$ *Res*(K'). Also, for vertex sets $S_1, S_2 \subseteq V$, let $[S_1, S_2]$ be the set of edges with one end point in S_1 and the other end point in S_2.

Lemma 5. *For all K, $[$Sep(K), Res$(T^{(K)})]$ is an edge cut.*

We need the following definition in our algorithm.

Definition 5 (Emission condition for a clique)**.** *Clique K satisfies the emission condition if Em$(K) =$ Sep(K).*

Remark 1. *Theorem 1 implies that clique K satisfies the emission condition if and only if all elements in Sep(K) satisfy the emission condition in K.*

In the recursive approach, once the algorithm finds all the directed edges in $G^{(r)}$, it removes all the oriented edges for the next stage and restarts with the undirected components, i.e., the edges that it removes are the directed edges. Therefore, we require an edge cut in which all the edges are directed. This is satisfied by cliques with emission condition:

Function $\text{RS}(G, T, K_{root}, \textit{Sep}(K_{root}))$

```
1:  Initiate: $Size_T = 1$, $Explore = K_{root}$.
2:  Orient from $Sep(K_{root})$ to $Res(K_{root})$ in $G$.
3:  $\tilde{G}$ := Subgraph of $G$ induced on vertices in $T$ minus edges among vertices in $Sep(K_{root})$.
4:  while $Explore \neq \emptyset$ do
5:      for $K \in Ch(Explore)$ do
6:          Form $Em(K)$.
7:          if $K$ satisfies emission condition then
8:              $T = T \backslash T^{(K)}$
9:              Remove $[Sep(K), Res(T^{(K)})]$ and com-
10:             ponents containing $Res(T^{(K)})$ from $\tilde{G}$.
11:             if $(T^{(K)}, Sep(K)) \in$ Memory then
12:                 Load $[(T^{(K)}, Sep(K)), Size_{T^{(K)}}]$
13:                 $Size_T = Size_T \times Size_{T^{(K)}}$
14:             else
15:                 $Size_{T^{(K)}} = \text{RS}(G, T^{(K)}, K, Sep(K))$
16:                 Save $[(T^{(K)}, Sep(K)), Size_{T^{(K)}}]$
17:                 $Size_T = Size_T \times Size_{T^{(K)}}$
18:             end if
19:         else
20:             Orient from $Em(K)$ to $Res(K)$ in $G$ and $\tilde{G}$.
21:         end if
22:     end for
23:     $Explore = Ch(Explore)$
24: end while
25: Return: $Size_T \times \prod_{G' \in \text{undirected components of } \tilde{G}} \text{SIZE}(G')$.
```

Theorem 2. *If clique K satisfies the emission condition, $[Sep(K), Res(T^{(K)})]$ is a directed edge cut.*

Therefore, by Theorem 2, if clique K satisfies the emission condition, in the clique tree, we can learn the orientations in trees $T^{(r)} \setminus T^{(K)}$ and $T^{(K)}$ separately. This property helps us to perform our counting process efficiently by utilizing the memory in the process. More specifically, if rooted clique trees $T^{(r_1)}$ and $T^{(r_2)}$ share a rooted subtree whose root clique satisfies the emission condition, it suffices to perform the counting in this subtree only once. Based on this observation, we propose the counting approach whose pseudocode is presented in Algorithm 1.

The input to Algorithm 1 is an essential graph G^*, and it returns $Size(G^*)$ by computing equation (1), through calling function $\text{SIZE}(\cdot)$ for each chain component of G^*. In Function $\text{SIZE}(\cdot)$, first the clique tree corresponding to the input UCCG is constructed. If this tree has not yet appeared in the memory, for every vertex v of the input UCCG, the function forms $T^{(v)}$ and calculates the number of v-rooted DAGs in the MEC by calling the rooted-size function $\text{RS}(\cdot)$ (lines 10-13). Finally, it saves and returns the sum of the sizes.

The function RS checks whether each clique K of each level of the input rooted tree satisfies the emission condition (lines 7). We assume that all the variables in K_{root} satisfy the emission condition in all the cliques containing them. If emission condition was satisfied, it removes the rooted subtree $T^{(K)}$ from the tree and the corresponding subgraph from the graph (lines 8-10), and checks whether the size of $T^{(K)}$ with its current separator set is already in the memory.

r	p	20	30	40	50	60
0.2	T_1	0.50	2.26	6.65	19.55	55.59
	T_2	0.27	2.61	20.68	219.98	>3600
	T_2/T_1	0.54	1.15	3.11	11.25	>65
0.25	T_1	0.51	2.27	7.56	25.46	59.21
	T_2	0.40	8.77	101.84	1760.21	>3600
	T_2/T_1	0.78	3.86	13.47	69.12	>60

Table 1: Average run time (in seconds).

If so, the function loads it as $Size_{T^{(K)}}$ (lines 12); else, RS calls itself on the rooted clique tree $T^{(K)}$ to obtain $Size_{T^{(K)}}$, and then saves $Size_{T^{(K)}}$ in the memory (lines 15 and 16). If the clique K does not satisfy the emission condition, it simply orients edges from $Em(K)$ to $Res(K)$ (lines 20). Finally, in the resulting essential graph, it calls the function $\text{SIZE}(\cdot)$ for each undirected component (lines 25).

For bounded degree graphs, the proposed approach runs in polynomial time:

Theorem 3. *Let p and Δ be the number of vertices and maximum degree of a graph G. The computational complexity of MEC size calculator on G is in the order of $\mathcal{O}(p^{\Delta+2})$.*

Remark 2. *From Definition 4, it is clear that if $Sep(K_v) \cap Sep(K) = \emptyset$, then $v \in Sep(K)$ satisfies the emission condition in clique K. We can use this property for locally orienting edges without finding emission set $Em(K_v)$.*

3.2 Simulation Results

We generated 100 random UCCGs of order $p = 20, \cdots, 60$ with $r \times \binom{p}{2}$ as the number of edges based on the procedure proposed in (He, Jia, and Yu 2015), where parameter r controls the graph density. We compared the proposed algorithm with the counting algorithm in (He, Jia, and Yu 2015) in Table 1. Note that, as we mentioned earlier, since the counting methods are utilized for the purpose of sampling and applying prior knowledge, the five formulas in (He, Jia, and Yu 2015) are not implemented in either of the counting algorithms. The parameters T_1 and T_2 denote the average run time (in seconds) of the proposed algorithm and the counting algorithm in (He, Jia, and Yu 2015), respectively. For dense graphs, our algorithm is at least 60 times faster. Furthermore, our experiments showed that for the case of complete graphs or sparse graphs with number of edges of order $\mathcal{O}(p)$, the proposed algorithm runs much faster then the case of graphs with moderate density.

4 Uniform Sampling from a MEC

In this section, we introduce a sampler for generating random DAGs from a MEC. The sampler is based on the counting method presented in Section 3. The main idea is to choose a vertex as the root according to the portion of members of the MEC having that vertex as the root, i.e., in UCCG G, vertex v should be picked as the root with probability $Size(G^{(v)})/Size(G)$.

The pseudocode of our uniform sampler is presented in Algorithm 2, which uses functions $\text{SIZE}(\cdot)$ and $\text{RS}(\cdot)$ of Section 3.1. The input to the sampler is an essential graph G^*, with chain components $\mathcal{G} = \{G_1, \cdots G_c\}$. For each

Algorithm 2 Uniform Sampler

```
Input: Essential graph $G^*$, with chain components
        $\mathcal{G} = \{G_1, \cdots G_c\}$.
while $\mathcal{G} \neq \emptyset$ do
    Pick an element $G \in \mathcal{G}$, and update $\mathcal{G} = \mathcal{G} \setminus G$.
    Run ROOTED($G$).
end while
Return: $G^*$

function ROOTED($G$)
    Construct a clique tree $T = (\mathcal{K}_G, E(T))$.
    Set $v \in V(G)$ as the root with prob. $\frac{\mathrm{RS}(G, T^{(v)}, K, Sep(K))}{\mathrm{SIZE}(G)}$.
    For every clique $K$ in $T^{(v)}$, form $Em(K)$.
    Orient from $Em(K)$ to $Res(K)$ in $G^*$ and $G$.
    $\mathcal{G} = \mathcal{G} \cup \{$chain components of $G\}$.
end function
```

chain component $G \in \mathcal{G}$, we set $v \in V(G)$ as the root with probability $\mathrm{RS}(G, T^{(v)}, K, Sep(K))/\mathrm{SIZE}(G)$, where $K \in T_v$, and then we orient the edges in G^* as in Algorithm 1. We remove G and add the created chain components to $\mathcal{G}$, and repeat until all edges are oriented.

Example 2. *For the UCCG in Figure 1(a), as observed in Example 1, Size*$(G^{(v_1)}) =$ *Size*$(G^{(v_4)}) = 2$, *Size*$(G^{(v_2)}) =$ *Size*$(G^{(v_3)}) = 3$, *and Size*$(G) = 10$. *Therefore, we set vertices v_1, v_2, v_3, and v_4 as the root with probabilities* $2/10$, $3/10$, $3/10$, *and* $2/10$, *respectively. Suppose v_2 is chosen as the root. Then as seen in Example 1, Size*$(G''^{(v_1)}) =$ *Size*$(G''^{(v_3)}) =$ *Size*$(G''^{(v_4)}) = 1$. *Therefore, in G'', we set either of the vertices as the root with equal probability to obtain the final DAG.*

Theorem 4. *The sampler in Algorithm 2 is uniform.*

As a corollary of Theorem 3, for bounded degree graphs, the proposed sampler runs in polynomial time.

Corollary 3. *The computational complexity of the uniform sampler is in the order of* $\mathcal{O}(\Delta p^{\Delta+2})$.

5 Counting and Sampling with Prior Knowledge

Although in structure learning from observational data the orientation of some edges may remain unresolved, in many applications, an expert may have prior knowledge regarding the direction of some of the unresolved edges. In this section, we extend the counting and sampling methods to the case that such prior knowledge about the orientation of a subset of the edges is available. Specifically, we require that in the counting task, only DAGs which are consistent with the prior knowledge are counted, and in the sampling task, we force all the generated sample DAGs to be consistent with the prior knowledge. Note that the prior knowledge may not be necessarily realizable, that is, there may not exist any DAGs in the corresponding MEC with the required orientations. In this case, the counting should return zero and the sampling should return an empty set.

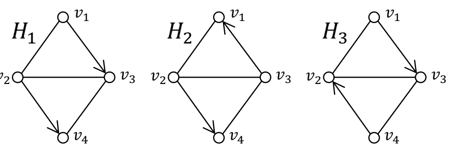

Figure 2: Graphs related to Example 3.

5.1 Counting with Prior Knowledge

We present the available prior knowledge in the form of a hypothesis graph $H = (V(H), E(H))$. Consider an essential graph G^*. For G^*, we call a hypothesis realizable if there exists a member of *MEC*(G^*) with directed edges consistent with the hypothesis. In other words, a hypothesis is realizable if the rest of the edges in G^* can be oriented without creating any v-structures or cycles. More formally:

Definition 6. *For an essential graph G^*, a hypothesis graph $H = (V(H), E(H))$ is called realizable if there exists a DAG D in MEC(G^*), for which $E(D) \subseteq E(H)$.*

For essential graph G^*, let $Size_H(G^*)$ denote the number of the elements of *MEC*(G^*), which are consistent with hypothesis H, i.e., $Size_H(G^*) = |\{D : D \in MEC(G^*), E(D) \subseteq E(H)\}|$. Hypothesis H is realizable if $Size_H(G^*) \neq 0$. As mentioned earlier, each chain component G of a chain graph contains exactly one root variable. We utilize this property to check the realizability and calculate $Size_H(G^*)$ for a hypothesis graph H. Consider essential graph G^* with chain components $\mathcal{G} = \{G_1, \cdots C_c\}$. Following the same line of reasoning as in equation (1), we have $Size_H(G^*) = \prod_{i=1}^{c} Size_H(G_i)$. Also, akin to equation (2), for any $G \in \mathcal{G}$, $Size_H(G) = \sum_{v \in V(G)} Size_H(G^{(v)})$. Therefore, in order to extend the pseudocode to the case of prior knowledge, we modify functions $\mathrm{SIZE}(\cdot)$ and $\mathrm{RS}(\cdot)$ to get H as an extra input. In our proposed pseudocode, the orientation task is performed in lines 2 and 20 of function RS. Let S be the set of directed edges of form (u, v), oriented in either line 2 or 20. In function RS, after each of lines 2 and 20, we check the following:

```
if $S \not\subseteq E(H)$ then Return: 0 end if
```

This guarantees that, any DAG considered in the counting will be consistent with the hypothesis H.

Example 3. *Consider the three hypothesis graphs in Figure 2 for the essential graph in Figure 1(a). For hypothesis H_1, $Size_{H_1}(G^{(v_1)}) = 2$, $Size_{H_1}(G^{(v_2)}) = 1$, and $Size_{H_1}(G^{(v_3)}) = Size_{H_1}(G^{(v_4)}) = 0$. Therefore, we have three DAGs consistent with hypothesis H_1, i.e., $Size_{H_1}(G) = 3$. For hypothesis H_2, $Size_{H_2}(G^{(v_2)}) = Size_{H_2}(G^{(v_3)}) = 2$, and $Size_{H_2}(G^{(v_1)}) = Size_{H_2}(G^{(v_4)}) = 0$, Therefore, four DAGs are consistent with hypothesis H_2, i.e., $Size_{H_2}(G) = 4$. Hypothesis H_3 is not realizable.*

One noteworthy application of checking the realizability of a hypothesis is in the context of estimating the causal effect of interventions from observational data (Maathuis, Kalisch, and Bühlmann 2009; Nandy, Maathuis, and Richardson 2017). This could be used for instance, to predict the effect of gene knockouts on other genes or some phenotype of interest, based on observational gene expression profiles. The authors of (Maathuis, Kalisch, and Bühlmann 2009; Nandy, Maathuis, and Richardson 2017) proposed a method called (joint-)IDA for estimating the average causal effect, which as a main step requires extracting possible valid parent sets of the intervention nodes from the essential graph, with the multiplicity information of the sets. To this end, a semi-local method was proposed in (Nandy, Maathuis, and Richardson 2017), which is exponential in the size of the chain component of the essential graph. This renders the approach infeasible for large components. Using our proposed method to address this problem, we can fix a configuration for the parents of the intervention target and count the number of consistent DAGs.

5.2 Sampling with Prior Knowledge

Suppose an experimenter is interested in generating sample DAGs from a MEC. However, due to her prior knowledge, she requires the generated samples to be consistent with a given set of orientations for a subset of the edges. In this subsection, we modify our uniform sampler to apply to this scenario. We define the problem statement formally as follows. Given an essential graph G^* and a hypothesis graph H for G^*, we are interested in generating samples from $\mathit{MEC}(G^*)$ such that each sample is consistent with hypothesis H. Additionally, we require the distribution of the samples to be uniform conditioned on being consistent. That is, for each sample DAG $D \in \mathit{MEC}(G^*)$, $P(D) = 1/\mathit{Size}_H(G^*)$, if $E(D) \subseteq E(H)$, and $P(D) = 0$, otherwise. The mentioned equations for calculating $\mathit{Size}_H(\cdot)$ imply that we can use a method similar to the case of the uniform sampler. That is, we choose a vertex as the root according to the ratio of the DAGs $D \in \mathit{MEC}(G^*)$ which are consistent with H and have the chosen vertex as the root, to the total number of consistent DAGs. More precisely, in UCCG G, vertex v should be picked as the root with probability $\mathit{Size}_H(G^{(v)})/\mathit{Size}_H(G)$. In fact, the uniform sampler could be viewed as a special case of sampler with prior knowledge with $H = G^*$. Hence, the results related to the uniform sampler extend naturally.

6 Application to Intervention Design

In this section, we demonstrate that the proposed method for calculating the size of MEC with prior knowledge can be utilized to design an optimal intervention target in experimental causal structure learning. We will use the setup in (Ghassami et al. 2018a): Let G^* be the given essential graph, and let k be our intervention budget, i.e., the number of interventions we are allowed to perform. Each intervention is on only a single variable and the interventions are designed passively, i.e., the result of one intervention is not used for the design of the subsequent interventions. Let $\mathcal{I}$ denote the *intervention target set*, which is the set of vertices that we intend to intervene on. Intervening on a vertex v resolves the orientation of all edges intersecting with v (Eberhardt, Glymour, and Scheines), and then we can run Meek rules to learn the maximal PDAG (Perković, Kalisch, and Maathuis 2017). Let $R(\mathcal{I}, D)$ be the number of edges that their orientation is resolved had the ground truth underlying DAG been D, and

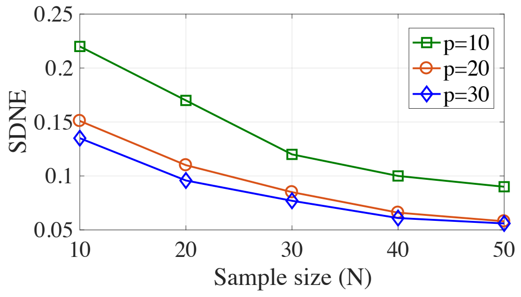

Figure 3: SD of the normalized error versus the sample size.

let $\mathcal{R}(\mathcal{I})$ be the average of $R(\mathcal{I}, D)$ over the elements of the MEC, that is, $\mathcal{R}(\mathcal{I}) = \frac{1}{\mathit{Size}(G^*)} \sum_{D \in \mathit{MEC}(G^*)} R(\mathcal{I}, D)$. The problem of interest is finding the set $\mathcal{I} \subseteq V(G^*)$ with $|\mathcal{I}| = k$ that maximizes $\mathcal{R}(\cdot)$.

In (Ghassami et al. 2018a), it was proved that $\mathcal{R}(\cdot)$ is a sub-modular function and hence, a greedy algorithm recovers an approximation to the optimum solution. Still, calculating $\mathcal{R}(\mathcal{I})$ for a given $\mathcal{I}$ remains as a challenge. For an intervention target candidate, in order to calculate $\mathcal{R}(\mathcal{I})$, conceptually, we can list all DAGs in the MEC and then calculate the average according to the formula for $\mathcal{R}(\mathcal{I})$. However, for large graphs, listing all DAGs in the MEC is computationally intensive. Note that the initial information provided by an intervention is the orientation of the edges intersecting with the intervention target. Hence, we propose to consider this information as the prior knowledge and apply the method in Section 5. Let $\mathcal{H}$ be the set of hypothesis graphs, in which each element H has a distinct configuration for the edges intersecting with the intervention target. If the maximum degree of the graph is Δ, cardinality of $\mathcal{H}$ is at most $2^{k\Delta}$, and hence, it does not grow with p. For a given hypothesis graph H, let $G^*_H = \{D : D \in \mathit{MEC}(G^*), E(D) \subseteq E(H)\}$ denote the set of members of the MEC, which are consistent with hypothesis H. Using the set $\mathcal{H}$, we can break the expression of $\mathcal{R}(\mathcal{I})$ into two sums as follows.

$$\mathcal{R}(\mathcal{I}) = \frac{1}{\mathit{Size}(G^*)} \sum_{H \in \mathcal{H}} \sum_{D \in G^*_H} R(\mathcal{I}, D) = \sum_{H \in \mathcal{H}} \frac{\mathit{Size}_H(G^*)}{\mathit{Size}(G^*)} R(\mathcal{I}, D). \tag{3}$$

Therefore, we only need to calculate at most $2^{k\Delta}$ values instead of considering all elements of the MEC, which reduces the complexity from super-exponential to constant in p.

6.1 Simulation Results

An alternative approach to calculating $\mathcal{R}(\mathcal{I})$ is to estimate its value by evaluating uniform samples. We generated 100 random UCCGs of order $p = 10, 20, 30$, with $r \times \binom{p}{2}$ edges, where $r = 0.2$. In each graph, we selected two variables randomly to intervene on. We obtained the exact $\mathcal{R}(\mathcal{I})$ using equation (3). Furthermore, for a given sample size N, we estimated $\mathcal{R}(\mathcal{I})$ from the aforementioned Monte-Carlo approach using our proposed uniform sampler and obtained empirical standard deviation of the normalized error (SDNE) over all graphs with the same size, defined as $SD(|\mathcal{R}(\mathcal{I}) - \hat{\mathcal{R}}(\mathcal{I})|/\mathcal{R}(\mathcal{I}))$. Figure 3 depicts SDNE versus the number of samples. As can be seen, SDNE becomes fairly low for sample sizes greater than 40.

7 Conclusion

We proposed a new technique for calculating the size of a MEC, which is based on the clique tree representation of chordal graphs. We demonstrated that this technique can be utilized for uniform sampling from a MEC, which provides a stochastic way to enumerate DAGs in the class, which can be used for estimating the optimum DAG, most suitable for a certain desired property. We also extended our counting and sampling method to the case where prior knowledge about the structure is available, which can be utilized in applications such as causal intervention design.

Acknowledgments

This work was supported in part by MURI grant ARMY W911NF-15-1-0479 and ONR grant W911NF-15-1-0479.

References

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

Bernstein, M., and Tetali, P. 2017. On sampling graphical Markov models. *arXiv preprint arXiv:1705.09717*.

Blair, J. R., and Peyton, B. 1993. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*. Springer. 1–29.

Chickering, D. M. 2002. Optimal structure identification with greedy search. *JMLR* 3(Nov):507–554.

Eberhardt, F.; Glymour, C.; and Scheines, R. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables. In *UAI 2005*.

Eigenmann, M. F.; Nandy, P.; and Maathuis, M. H. 2017. Structure learning of linear gaussian structural equation models with weak edges. *arXiv preprint arXiv:1707.07560*.

Ghassami, A.; Salehkaleybar, S.; Kiyavash, N.; and Bareinboim, E. 2018a. Budgeted experiment design for causal structure learning. In *International Conference on Machine Learning*, 1719–1728.

Ghassami, A.; Salehkaleybar, S.; Kiyavash, N.; and Zhang, K. 2018b. Counting and sampling from Markov equivalent DAGs using clique trees. *arXiv preprint arXiv:1802.01239*.

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.

Hauser, A., and Bühlmann, P. 2012. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research* 13(Aug):2409–2464.

He, Y., and Geng, Z. 2008. Active learning of causal networks with intervention experiments and optimal designs. *Journal of Machine Learning Research* 9(Nov):2523–2547.

He, Y., and Yu, B. 2016. Formulas for counting the sizes of Markov equivalence classes of directed acyclic graphs. *arXiv preprint arXiv:1610.07921*.

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

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(1):2589–2609.

Hoyer, P. O., and Hyttinen, A. 2009. Bayesian discovery of linear acyclic causal models. In *UAI 2009*, 240–248.

Hoyer, P. O.; Hyvarinen, A.; Scheines, R.; Spirtes, P. L.; Ramsey, J.; Lacerda, G.; and Shimizu, S. 2012. Causal discovery of linear acyclic models with arbitrary distributions. *arXiv preprint arXiv:1206.3260*.

Koller, D., and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

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

Meek, C. 1995. Causal inference and causal explanation with background knowledge. In *UAI 1995*, 403–410.

Nandy, P.; Maathuis, M. H.; and Richardson, T. S. 2017. Estimating the effect of joint interventions from observational data in sparse high-dimensional settings. *The Annals of Statistics* 45(2):647–674.

Pearl, J. 2009. *Causality*. Cambridge university press.

Perković, E.; Kalisch, M.; and Maathuis, M. H. 2017. Interpreting and using cpdags with background knowledge. *arXiv preprint arXiv:1707.02171*.

Peters, J.; Janzing, D.; and Schölkopf, B. 2017. *Elements of causal inference: foundations and learning algorithms*. MIT press.

Rothenhäusler, D.; Ernest, J.; and Bühlmann, P. 2018. Causal inference in partially linear structural equation models: identifiability and estimation. *Ann. Stat. To appear*.

Sachs, K.; Perez, O.; Pe'er, D.; Lauffenburger, D. A.; and Nolan, G. P. 2005. Causal protein-signaling networks derived from multiparameter single-cell data. *Science* 308(5721):523–529.

Scheines, R.; Spirtes, P.; Glymour, C.; Meek, C.; and Richardson, T. 1998. The TETRAD project: Constraint based aids to causal model specification. *Multivariate Behavioral Research* 33(1):65–117.

Spirtes, P.; Glymour, C. N.; and Scheines, R. 2000. *Causation, prediction, and search*. MIT press.

Steinsky, B. 2013. Enumeration of labelled essential graphs. *Ars Combinatoria* 111:485–494.

Vandenberghe, L., and Andersen, M. S. 2015. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization* 1(4):241–433.

Verma, T., and Pearl, J. 1990. Equivalence and synthesis of causal models. In *UAI 1990*, 220–227.

Wang, Y.; Solus, L.; Yang, K.; and Uhler, C. 2017. Permutation-based causal inference algorithms with interventions. In *NIPS 2017*, 5822–5831.

Zhang, B.; Gaiteri, C.; Bodea, L.-G.; Wang, Z.; McElwee, J.; Podtelezhnikov, A. A.; Zhang, C.; Xie, T.; Tran, L.; Dobrin, R.; et al. 2013. Integrated systems approach identifies genetic nodes and networks in late-onset alzheimer's disease. *Cell* 153(3):707–720.