Counting Background Knowledge Consistent Markov Equivalent Directed Acyclic Graphs (Supplementary Material)

Vidya Sagar Sharma[1]

[1]School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, Maharashtra, India

A GRAPH TERMINOLOGY

Graphs and graph unions. We mostly follow the graph theory terminology of Andersson et al. [1997]. A graph is *undirected* if all its edges are undirected, *directed* if all its edges are *directed*, and *partially directed* if it contains both directed and undirected edges. A directed graph that has no directed cycle is called a *directed acyclic graph* (DAG). A generalization of this idea is that of *chain graphs*: a partially directed graph is called a chain graph if it has no cycle which contains (i) at least one directed edge, and (ii) in which all directed edges are directed in the same direction as one moves along the cycle. We denote the neighbors of a vertex v in a graph G as $N_G(v)$, and an induced subgraph of G on a set $X \subseteq V$ is denoted as $G[X]$. The *graph union* (which we just call "union") $G_1 \cup G_2$ includes vertices and edges present in any one of G_1 or G_2, i.e., $V_{G_1 \cup G_2} = V_{G_1} \cup V_{G_2}$, and $E_{G_1 \cup G_2} = E_{G_1} \cup E_{G_2}$. The *skeleton* of a partially directed graph G is an undirected version of G which we get by ignoring the direction of all the edges: in particular, note that the skeleton of a partially directed graph is also the graph union of *all* the partially directed graphs with the same skeleton as G. A *v-structure* (or *unshielded collider*) in a partially directed graph G is an ordered triple of vertices (a, b, c) of G which induce the subgraph $a \to b \leftarrow c$ in G.

Cliques, separators, chordal graphs, and UCCG. A *clique* is a set of pairwise adjacent vertices for a graph. For a graph G, u and v are said to be pairwise adjacent to each other if $(u, v), (v, u) \in E_G$, i.e, $u - v \in E_G$. For an undirected graph G, a set $S \subset V_G$ is an *x-y separator* for two non-adjacent vertices x and y if x and y are in two different undirected connected components of $G[V_G \setminus S]$. S is said to be a *minimal x-y separator* if no proper subset of S separates x and y. A set S is said to be a *minimal vertex separator* if there exist vertices x and y for which S is a minimal x-y separator.[1] An undirected graph G is *chordal* if, for any cycle of length 4 or more of G, there exist two non-adjacent vertices of the cycle which are adjacent in G. We refer to an undirected connected chordal graph by the abbreviation *UCCG*.

Clique trees. A *rooted clique tree* of a UCCG G is a tuple $\mathcal{T} = (T, R)$, where T is a rooted tree (rooted at the node R) whose nodes are the maximal cliques of G, and which is such that the set $\{C : v \in C\}$ is connected in T, for all $v \in V_G$. Clique trees satisfy the important *clique-intersection property*: if $C_1, C_2, C \in V_T$ and C is on the (unique) path between C_1 and C_2 in the tree T, then $C_1 \cap C_2 \subset C$ (see, e.g., Blair and Peyton [1993]). Further, a set $S \subset V_G$ is a minimal vertex separator in G if, and only if, there are two adjacent nodes $C_1, C_2 \in V_T$ such that $C_1 \cap C_2 = S$ [Blair and Peyton, 1993, Theorem 4.3]. A clique tree for a UCCG G can be constructed in polynomial time. For more details on the above results on chordal graphs and clique trees, we refer to the survey of Blair and Peyton [1993].

B PROOFS OMITTED FROM SECTION 3

Observation B.1. *If G is not $\mathcal{K}$-consistent then $\#AMO(G, \mathcal{K}) = 0$.*

Proof. If G is not $\mathcal{K}$-consistent then there exists an edge $u \to v \in \mathcal{K}$ such that $v \to u \in E_G$, and all the AMOs of G have the edge $v \to u$. This shows that no AMO of G is $\mathcal{K}$-consistent. □

[1]Wienöbst et al. [2021] refer to these objects as *minimal separators*, but we follow here the terminology of [Blair and Peyton, 1993, Section 2.2] for consistency.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

We can verify in polynomial time that G is $\mathcal{K}$-consistent or not, by checking the existence of an edge $u \to v \in \mathcal{K}$ for which $v \to u$ is a directed edge in G. This is why for further discussion we assume that G is $\mathcal{K}$-consistent.

Proposition 3.4 is a direct consequence of the following lemma.

Lemma B.2. *Let G be an MEC consistent with a given background knowledge $\mathcal{K}$, and let G_d be the directed subgraph of G. Then, $\alpha \in AMO(G, \mathcal{K})$ if, and only if, (i) for each undirected chordal component H of G, $\alpha[V_H]$ is a $\mathcal{K}$-consistent AMO of H and (ii) α is a union of G_d and $\bigcup_H \alpha[V_H]$.*

Proof. If G is $\mathcal{K}$-consistent then each directed edge of G is $\mathcal{K}$-consistent, i.e., G_d is $\mathcal{K}$-consistent. Andersson et al. [1997] show that for an MEC G, an AMO of G can be constructed by choosing an AMO from each one of the chordal components of G and taking the union of the directed subgraph of G and the chosen AMOs of the chordal components. For every undirected connected chordal component H of G, let us pick a $\mathcal{K}$-consistent AMO of H. Then, the union of all the picked AMOs and G_d is a $\mathcal{K}$-consistent AMO of G. Also if α is an AMO of G then for all undirected connected chordal components H of G, $\alpha[V_H]$ is $\mathcal{K}$-consistent. And α must be a union of G_d and the union of all the $\alpha[V_H]$. This proves Lemma B.2. □

C DEFINITION OF C_α

Here, we define C_α. We start with defining a few preliminary terms flower, $<_\mathcal{T}$, and $<_\alpha$ that we use to define C_α.

Definition C.1 (Flowers and bouquets, Wienöbst et al. [2021], Definition 4)**.** *Let G be a UCCG. An S-*flower *for a minimal vertex separator S of G is a maximal subset F of the set of maximal cliques of G containing S such that $\bigcup_{C \in F} C$ is connected in the induced subgraph $G[V \setminus S]$. The* bouquet *$B(S)$ of a minimal separator S is the set of all S-flowers.*

Definition C.2 (The $<_\mathcal{T}$ order for a rooted clique tree $\mathcal{T}$, Section 5 of Wienöbst et al. [2021])**.** *Let G be a UCCG, S be a minimal vertex separator of G, $F_1, F_2 \in B(S)$, and $\mathcal{T} = (T, R)$ be a rooted clique tree of G. $F_1 <_\mathcal{T} F_2$ if F_1 contains a node on the unique path from R to F_2.*

Definition C.3 (The $<_\alpha$ order for an AMO α, Section 5 of Wienöbst et al. [2021])**.** *Let G be a UCCG, $\mathcal{T} = (T, R)$ be a rooted clique tree of G, and α be an AMO of G. We use $<_\mathcal{T}$ to define a partial order $<_\alpha$ on the set of maximal cliques that represent α, as follows: $C_1 <_\alpha C_2$ if, and only if, (i) $C_1 \cap C_2 = S$ is a minimal vertex separator, (ii) C_1 and C_2 are elements of distinct S-flowers $F_1, F_2 \in B(S)$, respectively, and (iii) $F_1 <_T F_2$.*

The following result of Wienöbst et al. [2021] establishes the requisite property of the ordering $<_\alpha$.

Lemma C.4 (Wienöbst et al. [2021], Claim 1)**.** *Let G be a UCCG, α an AMO of G, and $\mathcal{T} = (T, R)$ a rooted clique tree of G. Consider the order $<_\alpha$ defined on the maximal cliques $\mathcal{T}$. Then, there always exists a unique least maximal clique with respect to $<_\alpha$.*

D PROOFS OMITTED FROM SECTION 4

Proof of Lemma 4.1. The claim follows from the fact that for each AMO α of G, C_α was canonically chosen from the set of maximal cliques representing α. □

Proof of Observation 4.4. Proof of item 1: For any edge $u - v \in E_G$, if $u \in C$ and $v \notin C$ then for any AMO that is represented by an LBFS ordering that starts with C, $u \to v$ is a directed edge in the AMO (see "Representation of an AMO" of Section 2). This further implies for any edge $u - v \in E_G$, if $u \in C$ and $v \notin C$ then $u \to v$ is a directed edge in G^C, as G^C is the union of all the AMOs of G that can be represented by an LBFS ordering that starts with C. This proves item 1.

Proof of item 2: We prove item 2 of Observation 4.4 using induction on the size of $\mathcal{L}$.

Base Case: $|\mathcal{L}| = 0$**.** In this case, item 2 of Observation 4.4 is vacuously true.

Let item 2 is true when $|\mathcal{L}| = l \geq 0$. We show that item 2 is true even for $|\mathcal{L}| = l + 1$.

Let at some iteration of Algorithm 1, $\mathcal{L} = \{X_1, X_2, \ldots, X_l, X_{l+1}\}$. From the induction hypothesis, for an edge $u - v \in E_G$, if $u \in X_i$ and $v \notin C \cup X_1 \cup X_2 \cup \ldots \cup X_i$ then $u \to v \in G^C$, when $i \leq l$. Let there exists an edge $u - v \in E_G$ such that $u \in X_{l+1}$ and $v \notin C \cup X_1 \cup X_2 \cup \ldots \cup X_{l+1}$. This means there must exist a vertex $x \in C \cup X_1 \cup X_2 \cup \ldots \cup X_l$ for which

$x - u \in E_G$ and $v - u \notin E_G$, due to which u and v moves to two different sets in $\mathcal{S}$, because initially u and v are in the same set $V \setminus C$. From the induction hypothesis, $x \to u \in G^C$. This implies all the AMOs that are represented by C have edge $x \to u$. This further implies all the AMOs that are represented by C have edge $u \to v$, as $v \to u$ creates an immorality $x \to u \leftarrow v$ (from the definition of AMO, there cannot be a v-structure in an AMO of G). This proves item 2.

Proof of item 3: Suppose $u, v \in X_i$, and $u - v \in E_G$. Let there exists an AMO α that is represented by C and has the edge $u \to v$. Let τ_1 be an LBFS ordering of G that starts with C, and represents α. We can construct another LBFS ordering τ_2 that also starts with C such that while picking the vertices of X_i, we pick v before u. The AMO corresponding to this LBFS ordering has the edge $v \to u$. Since G^C is the union of all the AMOs of G that is represented by C. This implies G^C has the undirected edge $u - v$. This proves item 3.

Proof of item 4: If $u, v \in C$ then $u - v \in E_G$, because C is a clique of G. Suppose an AMO α is represented by C and has the edge $u \to v$. Then, α must be represented by an LBFS ordering τ_1 that starts with a permutation $\pi_1(C)$ of C such that u comes before v in $\pi_1(C)$. Let $\pi_2(C)$ be a permutation of C such that v comes before u in $\pi_2(C)$. Let us construct an LBFS ordering τ_2 by replacing $\pi_1(C)$ with $\pi_2(C)$ in τ_1. Let β be the AMO represented by the LBFS ordering τ_2. β also represented by C and has the edge $v \to u$. Since G^C is the union of all the AMOs of G that is represented by C, this implies $u - v$ is an undirected edge in G^C, because $u \to v \in \alpha$, and $v \to u \in \beta$, and both are represented by C. This proves item 4.

Proof of item 5: From items 1 and 2, all the edges with only one endpoint in X_l are directed in G^C. And, from item 3, all the edges with both of the endpoints in X_l are undirected in G^C. This further implies all the undirected connected components of $G[X_i]$ are undirected connected components of G^C. □

Proof of Lemma 4.5. At first, we want to recall that Algorithm 1 is background aware version of the modified LBFS algorithm of Wienöbst et al. [2021]. As discussed in the main paper, we do not change any line from the LBFS algorithm of Wienöbst et al. [2021]; the only modifications we do to their LBFS algorithm are (a) introduction of "$flag$", at line 2, which is used to check the $\mathcal{K}$-consistency of G^C, (b) lines 11-13, which is used to update the value of "$flag$", and (c) We also output the value of "$flag$" with $C_G(C)$. The output of Algorithm 1 has 2 components. The first component is the value of "$flag$", and the second value is the value returned by the LBFS algorithm of Wienöbst et al. [2021], which is $C_G(C)$. Thus, the only thing we need to verify is that the first component of our output, i.e., the value of "$flag$", is 1 if G^C is $\mathcal{K}$-consistent, and 0 if G^C is not $\mathcal{K}$-consistent, which is equivalent to show that the value of "$flag$" returned by the algorithm is 0 if, and only if, G^C is not $\mathcal{K}$-consistent (since the value of "$flag$" is either 0 or 1).

Suppose G^C is not $\mathcal{K}$-consistent. Then, from the definition of $\mathcal{K}$-consistency of G^C (Definition 4.3), there must exist an edge $v \to u$ in G^C such that $u \to v \in \mathcal{K}$. From Observation 4.4, if $v \to u \in G^C$ then either (a) $v \in C$ and $u \notin C$, or (b) at some iteration, when $\mathcal{L} = \{X_1, X_2, \ldots, X_l\}$, $v \in X_l$ and $u \notin C \cup X_1 \cup X_2 \cup \ldots \cup X_l$. In both of the cases, v must be picked before u at line-5, as v is present in a set that comes before the set in which u is present. At the iteration when v is picked, the algorithm finds the edge $u \to v$ that obeys the condition stated in line-11. This further sets the value of "$flag$" to 0. From the construction of the algorithm, once the value of "$flag$" sets to 0, it remains at 0. This shows that if G^C is not $\mathcal{K}$-consistent then the value of "$flag$" is 0.

Now suppose the "$flag$" value returned by the algorithm is 0. At line-2, the value of "$flag$" is initialized with 1. If "$flag$" value returned by the algorithm is 0 then there must exist an edge $u \to v \in \mathcal{K}$ (found at line-11), which causes to change the value of "$flag$" to 0 at line-12. Since $u \to v$ obeys the condition state at line-11, we can certainly say that the iteration when v is picked at the line-5, u must be neither in C nor in any set of $\mathcal{L}$. And, v must be either in C, if $\mathcal{L} = \varnothing$, or in X_l, if $\mathcal{L} = \{X_1, X_2, \ldots, X_l\}$ such that $l \geq 1$. From Observation 4.4, $v \to u$ must be a directed edge in G^C. This makes G^C inconsistent with $\mathcal{K}$, as $u \to v \in \mathcal{K}$ (Definition 4.3). This shows if the value of "$flag$" returned by the algorithm is 0 then G^C is not $\mathcal{K}$-consistent. This completes the proof. □

Proof of Observation 4.6. If α is a member of AMO$(G, \pi_1(C))$ and AMO$(G, \pi_2(C))$ both, for two different permutations $\pi_1(C)$ and $\pi_2(C)$, then there must exist two vertices $u, v \in C$ such that u comes before v in $\pi_1(C)$, and v comes before u in $\pi_2(C)$. Since u and v are members of the same clique, there exists an edge $u - v \in E_G$. And, since the AMO is a member of both AMO$(G, \pi_1(C))$, and AMO$(G, \pi_2(C))$ it should have both $u \to v$ and $v \to u$, which is not possible. This implies there exists a unique permutation $\pi(C)$ of C that represents α. □

Proof of Lemma 4.7. Claims 1 and 2 of Wienöbst et al. [2021] translate into the "only if" part of Lemma 4.7, while Claim 3 of Wienöbst et al. [2021] translates into the "if" part of Lemma 4.7. We give details of the translation below. Given an AMO α, Wienöbst et al. [2021] define a partial order $\prec_\alpha$ (as described above) on the set of maximal cliques that represent α. Claim 1 of Wienöbst et al. [2021] shows that there exists a unique maximal clique representing α (which we name as C_α)

such that for any maximal clique $C \neq C_\alpha$ that represents α, $C_\alpha <_\alpha C$. Claim 2 of Wienöbst et al. [2021] then translates immediately into the "only if" part of Lemma 4.7.

For the "if" part, we consider any maximal clique C representing α and satisfying both the conditions of Lemma 4.7. Suppose, if possible, that $C \neq C_\alpha$. Then, from the above discussion, $C_\alpha <_\alpha C$. The proof of Claim 3 (and the definition of $FP(C, \mathcal{T})$) then shows that if $\pi(C)$ is the permutation of C representing α, then there is a prefix S of $\pi(C)$ of the form $C_i \cap C_j$ for some two adjacent cliques on the path from R to C in T. This leads to a contradiction with item 2 of Lemma 4.7, and hense shows that $C \neq C_\alpha$ is not possible. □

For the proof of Lemma 4.11, we need the following observation of Wienöbst et al. [2021].

Observation D.1 (Wienöbst et al. [2021], Proposition 1)**.** *For each permutation $\pi(C)$ of a maximal clique C of G, all edges of $G^{\pi(C)}$ coincide with the edges of G^C, excluding the edges connecting the vertices in C. In particular, $\mathcal{C}_G(\pi(C)) = \mathcal{C}_G(C)$.*

Proof of Lemma 4.11. Let G^C be $\mathcal{K}$-consistent, and $\pi(C)$ is a $(\mathcal{K}, \mathcal{T})$-consistent permutation of C. We show that the number of $\mathcal{K}$-consistent AMOs of G that are canonically represented by $\pi(C)$, i.e., $|\{\alpha : \alpha \in \text{AMO}(G, \pi(C), \mathcal{K}) \text{ and } C = C_\alpha\}|$ equals $\Pi_{H \in \mathcal{C}_G(C)} \#\text{AMO}(H, \mathcal{K}[H])$. To prove this, we first show that if α is a $\mathcal{K}$-consistent AMO of G that is canonically represented by $\pi(C)$ then for any connected component H of $\mathcal{C}_G(C)$, $\alpha[H]$ is $\mathcal{K}[H]$-consistent AMO of H. We also show that if we have a $\mathcal{K}[H]$-consistent AMO for each connected component H of $\mathcal{C}_G(C)$ then we can construct a $\mathcal{K}$-consistent AMO of G by combining them. This proves Lemma 4.11.

We first show the first part. Let α be a $\mathcal{K}$-consistent AMO of G that is canonically represented by $\pi(C)$. Then, for any connected component H of $\mathcal{C}_G(C)$, $\alpha[H]$ is $\mathcal{K}[H]$-consistent AMO of H. Otherwise, if for any connected component H of $\mathcal{C}_G(C)$, $\alpha[H]$ is not $\mathcal{K}[H]$-consistent then there must exist an edge $u \to v \in \mathcal{K}[H]$ such that $v \to u \in \alpha[H]$. But, this implies α is not $\mathcal{K}$-consistent either, as if $u \to v \in \mathcal{K}[H]$ then $u \to v \in \mathcal{K}$, and if $v \to u \in \alpha[H]$ then $v \to u \in \alpha$.

We now show the other part. Let $H_1, H_2, \ldots, H_l$ are the undirected connected components of $\mathcal{C}_G(C)$, in the same order as we get as the output of Algorithm 1 for input G, C, and $\mathcal{K}$. For each $H_i \in \mathcal{C}_G(C)$, let we have a $\mathcal{K}[H_i]$-consistent AMO D_i, and τ_i be an LBFS ordering of D_i. Then, $\tau = \{\pi(C), \tau_1, \tau_2, \ldots, \tau_l\}$ is a $\mathcal{K}$-consistent LBFS ordering of G starting with $\pi(C)$ that we can get from Algorithm 1. The DAG α represented by τ is an AMO of G represented by $\pi(C)$. And, since $\pi(C)$ is $(\mathcal{K}, \mathcal{T})$-consistent, from Lemma 4.7, $C = C_\alpha$. This implies α is a $\mathcal{K}$-consistent AMO of G and is canonically represented by $\pi(C)$. This gives us a one-to-one mapping between the set of $\mathcal{K}$-consistent AMOs of G that is canonically represented by $\pi(C)$, and the $\mathcal{K}[H]$-consistent AMOs of the connected components H of $\mathcal{C}_G(C)$, which further implies the equality between the size of $\mathcal{K}$-consistent AMOs of G that is canonically represented by $\pi(C)$ and $\Pi_{H \in \mathcal{C}_G(C)} \#\text{AMO}(H, \mathcal{K}[H])$. This completes our proof. □

Proof of Lemma 4.13. From Lemma 4.1, $\#\text{AMO}(G, \mathcal{K}) = \sum_{C \in \Pi(G)} |\{\alpha : \alpha \in \text{AMO}(G, \mathcal{K}) \text{ and } C = C_\alpha\}|$. In Section 4, after defining $\mathcal{K}$-consistency of G^C (Definition 4.3), we show that for any maximal clique C of G, if G^C is not $\mathcal{K}$-consistent then $|\{\alpha : \alpha \in \text{AMO}(G, \mathcal{K}) \text{ and } C = C_\alpha\}| = 0$. From Lemma 4.11, for any maximal clique C of G, if G^C is $\mathcal{K}$-consistent then for any $(\mathcal{K}, \mathcal{T})$-consistent permutation $\pi(C)$ of C, $|\{\alpha : \alpha \in \text{AMO}(G, \pi(C), \mathcal{K}) \text{ and } C = C_\alpha\}| = \prod_{H \in \mathcal{C}_G(C)} \#AMO(H, \mathcal{K}[H])$. And, from Observation 4.10, for any permutation $\pi(C)$ of a maximal clique C of G, if $\pi(C)$ is not a $(\mathcal{K}, \mathcal{T})$-consistent permutation of C then $|\{\alpha : \alpha \in \text{AMO}(G, \pi(C), \mathcal{K}) \text{ and } C = C_\alpha\}| = 0$. This further implies for any maximal clique C of G, if G^C is $\mathcal{K}$-consistent then $|\{\alpha : \alpha \in \text{AMO}(G, \mathcal{K}) \text{ and } C = C_\alpha\}| = \Phi(C, FP(C, \mathcal{T}), \mathcal{K}[C]) \times \prod_{H \in \mathcal{C}_G(C)} \#AMO(H, \mathcal{K}[H])$, where $\Phi(C, FP(C, \mathcal{T}), \mathcal{K}[C])$ is the number of $(\mathcal{K}, \mathcal{T})$-consistent permutations of a maximal clique C of G (from Definitions 4.9 and 4.12). All these things further imply $\#\text{AMO}(G, \mathcal{K}) = \sum_{C : G^C \text{ is } \mathcal{K}\text{-consistent}} \Phi(C, FP(C, \mathcal{T}), \mathcal{K}[C]) \times \prod_{H \in \mathcal{C}_G(C)} \#AMO(H, \mathcal{K}[H])$. This proves the correctness of Lemma 4.13. □

Proof of Lemma 4.15. Proof of item 1: If $R = \varnothing$, then a permutation $\pi(S)$ of S is $\mathcal{K}$-consistent if ordering of $V_\mathcal{K}$ in $\pi(S)$ is $\mathcal{K}$-consistent. $\Psi(V_\mathcal{K}, \mathcal{K})$ gives the number of $\mathcal{K}$-consistent permutations of $V_\mathcal{K}$. Number of permutations of S that has the same ordering of $V_\mathcal{K}$ in it is $\frac{|S|!}{|V_\mathcal{K}|!}$. This completes the proof of item 1.

Proof of item 2: If there exists an edge $u \to v \in \mathcal{K}$ such that $u \in S \setminus R_l$ and $v \in R_l$ then no $\mathcal{K}$-consistent permutation of S exists that starts with R_l.

Proof of item 3: If there does not exist an edge $u \to v \in \mathcal{K}$ such that $u \in S \setminus R_l$ and $v \in R_l$, then one way to compute $\Phi(S, R, \mathcal{K})$ is to first compute number of $\mathcal{K}$-consistent permutations of S that do not start with $R_1, R_2, \ldots, R_{l-1}$, i.e.,

$\Phi(S, R - \{R_l\}, \mathcal{K})$. But, $\Phi(S, R - \{R_l\}, \mathcal{K})$ also counts the $\mathcal{K}$-consistent permutations of S that starts with R_l but not with any R_i, for $1 \leq i < l$. We subtract such permutations from $\Phi(S, R - R_l, \mathcal{K})$. To construct a $\mathcal{K}$-consistent permutation of S that starts with R_l but does not start with any R_i, $1 \leq i < l$, we have to first construct a permutation of R_l that does not start with any R_i, $1 \leq i < l$, and then we have to construct a $\mathcal{K}$-consistent permutation of the remaining vertices of S. This implies the number of $\mathcal{K}$-consistent permutations of S that start with R_l and not with any R_i, $1 \leq i < l$, is $\Phi(R_l, R - \{R_l\}, \mathcal{K}) \times \Phi(S \setminus R_l, \varnothing, \mathcal{K})$. □

Proof of Observation 4.16. If $R = \varnothing$ then from item 1 of Lemma 4.15, $\Phi(S, R, \mathcal{K}) = \frac{|S|!}{|V_{\mathcal{K}}|!} \times \Psi(V_{\mathcal{K}}, \mathcal{K})$. Line 2 of Algorithm 2 returns the same.

If $R = \{R_1, R_2, \ldots, R_l\} \neq \varnothing$, and there exist an edge $u \to v \in \mathcal{K}$ such that $u \in S \setminus R_l$ and $v \in R_l$, then from item 2 of Lemma 4.15, $\Phi(S, R, \mathcal{K}) = \Phi(S, R - R_l, \mathcal{K})$. Line 6 of Algorithm 2 returns the same.

If $R = \{R_1, R_2, \ldots, R_l\} \neq \varnothing$, and there does not exist an edge $u \to v \in \mathcal{K}$ such that $u \in S \setminus R_l$ and $v \in R_l$, then from item 3 of Lemma 4.15, $\Phi(S, R, \mathcal{K}) = \Phi(S, R - R_l, \mathcal{K}) - \Phi(R_l, R - \{R_l\}, \mathcal{K}[R_l]) \times \Phi(S \setminus R_l, \varnothing, \mathcal{K}[S \setminus R_l])$. The line 8 of Algorithm 2 returns the same. □

Proof of Theorem 4.17. At line 4, Algorithm 3 constructs a rooted clique tree of G. Lines 5-8 deals with a special case when G is a clique. If $R = V$ (at line 5) then G is a clique. In this case, the number of $\mathcal{K}$-consistent AMOs of G equals the number of $\mathcal{K}$-consistent permutation of V. Lines 5-8 do the same. For the general case (when G is not a clique), we implement Lemma 4.13. We create a queue Q at line 10, that stores maximal cliques of G. For each maximal clique C of G, we run lines 11-22. At line 14, we call Algorithm 1 (our LBFS-algorithm) for input G, C and $\mathcal{K}$. If the first component of the output of Algorithm 1 is 0 then G^C is not $\mathcal{K}$-consistent (from Lemma 4.5). This further implies $|\{\alpha : \alpha \in \text{AMO}(G, \pi(C), \mathcal{K}) \text{ and } C = C_\alpha\}| = 0$. This is why we skip lines 16-21 if the first component of LBFS$(G, C, \mathcal{K})$ (Algorithm 1) is 0. If the first component LBFS$(G, C, \mathcal{K})$ is 1 then G^C is $\mathcal{K}$-consistent. In this case, at lines 17-19, we compute $\prod_{H \in C_G(C)} \#AMO(H, \mathcal{K}[H])$, by recursively calling Algorithm 3. At line 20, we compute $|\{\alpha : \alpha \in \text{AMO}(G, \mathcal{K}) \text{ and } C = C_\alpha\}|$ using the discussion following Lemma 4.11. At the end of line-22, the variable `sum` has the value $\sum_{C: G^C \text{ is } \mathcal{K}\text{-consistent}} \Phi(C, FP(C, \mathcal{T}), \mathcal{K}[C]) \times \prod_{H \in C_G(C)} \#AMO(H, \mathcal{K}[H])$, i.e., `sum` equals $\#AMO(G, \mathcal{K})$ (from Lemma 4.13). `memo`$[G]$ stores the number of $\mathcal{K}$-consistent AMOs of G, once it is computed. This completes the proof.

□

E PROOFS OMITTED FROM SECTION 5

Proof of Observation 5.1. The "While" loop on lines 3-19 runs at most $|V_G|$ times. Using two additional arrays (one for checking whether v is in C or not, and another for checking whether v is in $\mathcal{L}$ or not) we can run lines 6-9 in $O(1)$ time. Checking the existence of edge $u \to v \in \mathcal{K}$ at line 11 takes $O(|\mathcal{K}|)$ time cumulatively, for all $v \in V_G$. Finding neighbors of v at line 15 takes $O(E_G)$ time cumulatively, for all $v \in V_G$. Partitioning each set at line 17, also takes $O(E_G)$ time cumulatively. Since the size of $\mathcal{K}$ can be at most E_G, this implies the overall time complexity of Algorithm 1 is $O(|V_G| + |E_G|)$ (with the same technique that is used to implement the standard LBFS of Rose et al. [1976]). □

Proof of Proposition 5.2. Algorithm 3 calls itself at line 18, for each $H \in \mathcal{L}$, when `flag` $= 1$. The value of `flag` is always 1 when $\mathcal{K} = \varnothing$, and $\mathcal{L} = \mathcal{C}_G(C)$ (from Algorithm 1) does not depend on $\mathcal{K}$. This further implies that the `count` function has the maximum number of distinct recursive calls to itself when $\mathcal{K} = \varnothing$. But, for $\mathcal{K} = \varnothing$, Algorithm 3 is the same as the Clique-Picking algorithm of Wienöbst et al. [2021], who showed that the number of these distinct recursive calls is at most $2|\Pi(G)| - 1$ times. This completes the proof. □

Proof of Lemma 5.3. Let us denote $\mathcal{R}_i = \{R_1, R_2, \ldots, R_i\}$. For the computation of $\Phi(S, \mathcal{R}_l, \mathcal{K})$, from items 2 and 3 of Lemma 4.15, we need to compute $\Phi(S, \varnothing, \mathcal{K})$, $\Phi(S \setminus R_1, \varnothing, \mathcal{K}[S \setminus R_1])$, $\Phi(S \setminus R_2, \varnothing, \mathcal{K}[S \setminus R_2]), \ldots, \Phi(S \setminus R_l, \varnothing, \mathcal{K}[S \setminus R_l])$, $\Phi(S, \mathcal{R}_1, \mathcal{K})$, $\Phi(S, \mathcal{R}_2, \mathcal{K}), \ldots, \Phi(S, \mathcal{R}_{l-1}, \mathcal{K})$, and for each $1 \leq i \leq l$, $\Phi(R_i, \varnothing, \mathcal{K}[R_i])$, $\Phi(R_i \setminus R_1, \varnothing, \mathcal{K}[R_i \setminus R_1]), \ldots,$ $\Phi(R_i \setminus R_{i-1}, \varnothing, \mathcal{K}[R_i \setminus R_{i-1}])$, $\Phi(R_i, \mathcal{R}_1, \mathcal{K}[R_i])$, $\Phi(R_i, \mathcal{R}_2, \mathcal{K}[R_i]), \ldots, \Phi(R_i, \mathcal{R}_{i-1}, \mathcal{K}[R_i])$.

Computation of each $\Phi(X, \phi, \mathcal{K}' = \mathcal{K}[X])$ takes $O(k^2 \cdot k!)$ arithmetic operations. To see this, note that item 1 of Lemma 4.15 gives $\Phi(X, \phi, \mathcal{K}') = \frac{|X|!}{|V_{\mathcal{K}'}|} \times \Psi(V_{\mathcal{K}'}, \mathcal{K}')$. From our assumption, the factorials are already pre-computed. On the other hand, to compute $\Psi(V_{\mathcal{K}'}, \mathcal{K}')$ we can check one-by-one the $\mathcal{K}'$-consistency of each permutation of $V_{\mathcal{K}'}$. The number of permutations of $V_{\mathcal{K}'}$ is $O(k!)$, since $|V_{\mathcal{K}'}| \leq k$ (max-clique knowledge), while the verification of whether a permutation of

$V_{\mathcal{K}'}$ is $\mathcal{K}'$-consistent or not takes $O(|\mathcal{K}'|)$ time. Since $V_{\mathcal{K}'} \leq k$, we have $|\mathcal{K}'| \leq k^2$. Thus, the computation of $\Psi(V_{\mathcal{K}'}, \mathcal{K}')$ takes $O(k^2 \cdot k!)$ arithmetic operations.

Since the number of required computations of the type $\Phi(X, \phi, \mathcal{K}' = \mathcal{K}[X])$ (as already listed above) is $O(l^2)$, their total cost is $O(k! \cdot k^2 \cdot l^2)$. After all the values listed above of the type $\Phi(X, \phi, \mathcal{K}' = \mathcal{K}[X])$ have been computed, we can implement a dynamic programming procedure using items 2 and 3 of Lemma 4.15 (or, alternatively, a memoized version of Algorithm 2) to compute all the required values of type $\Phi(X, Y \neq \varnothing, \mathcal{K}[X])$ (as listed above) using $O(1)$ arithmetic operations each. Since the number of required computations of the type $\Phi(X, Y \neq \varnothing, \mathcal{K}[X])$ (again, as already listed above) is also $O(l^2)$, it follows that the total cost of these computations is $O(l^2)$. Adding the computational costs for both types of computations, we see that the total cost is $O(k! \cdot k^2 \cdot l^2)$ arithmetic operations.

As the value of l can be at most $|\Pi(G)|$ (the number of nodes in the clique tree $\mathcal{T}$), the overall number of arithmetic operations we need to compute $\Phi(S, \mathcal{R}_l, \mathcal{K})$ is $O(k! \cdot k^2 \cdot |\Pi(G)|^2)$.

□

Proof of Theorem 5.4. Algorithm 3 is analogous to Clique-Picking Algorithm of Wienöbst et al. [2021]. We can also say that Algorithm 3 is the background knowledge version of the Clique-Picking Algorithm of Wienöbst et al. [2021]. At line 4, we construct a clique tree of G, which takes $O(|V_G| + |E_G|)$ time. Computing Φ function at line 6, for a clique C, takes $O(|\Pi(G)|^2 \cdot k^2 \cdot k!)$ time (from Lemma 5.3). While loop (at lines 11-22) runs for $O(|\Pi(G)|)$ times, as the number of maximal cliques of G is $|\Pi(G)|$. Running LBFS-algorithm (Algorithm 1) at line 14 takes $O(|V| + |E|)$ time (from Observation 5.1). Computation of function Φ at line 20 takes $O(k! \cdot k^2 \cdot |\Pi(G)|^2)$ time (from Lemma 5.3).

Note that it was assumed in Algorithm 2 that factorials of integers from 1 to $|V_G|$ are pre-computed. The pre-computation of this table can be done using $O(|V_G|)$ arithmetic operations. From Proposition 5.2, the number of distinct calls to `count` function of Algorithm 3 is $O(|\Pi(G)|)$. Together, the above calculations show that the running time of Algorithm 3 is at most $O(k! \cdot k^2 \cdot |\Pi(G)|^4)$ or $O(k! \cdot k^2 \cdot |V_G|^4)$, as for a chordal graph G, $|\Pi(G)|$ can be at most $|V_G|$. □

F DETAILED EXPLANATION OF EXPERIMENTAL RESULTS

Construction of chordal graphs with n vertices: We first construct a connected *Erdős-Rényi graph* G with n vertices such that each of its edges is picked with probability p, where p is a random value in [0.1, 0.3). We give a unique rank to each node of the graph. We then process the vertices in decreasing order of rank. For each vertex x of the graph, if u and v are two neighbors of x such that u and v are not connected, and u and v both have lesser rank than the rank of x, then we put an edge between u and v in G. This makes G an undirected chordal graph, because, by construction, the decreasing order of ranks is a perfect elimination ordering. After this, we use rejection sampling to get an undirected connected chordal graph, i.e., if G is not a connected graph then we reject G, and repeat the above process until we get an undirected connected chordal graph G.

Construction of background knowledge edges: For each chordal graph constructed above, and for each $k \in \{5, 6, \ldots, 13\}$, we construct a set of background knowledge edges such that (i) for any maximal clique C of size greater than or equal to k, the number of vertices of the clique that are part of an edge of the background knowledge with both endpoint in C is k; and (ii) for any maximal clique of size less than k, the number of vertices of the clique that are part of an edge of background knowledge equals to the size of the clique. To do this, we pick one by one each maximal clique of G, and select the edges of the clique such that at most k vertices are involved in the set of selected edges with both of its endpoints in that clique. For this, we construct a rooted clique tree $\mathcal{T} = (T, R)$ of G. We start with R and then do a depth-first search (DFS) on T to cover all the maximal cliques of G. At the iteration when we are at a maximal clique C, we first compute the set of vertices of the picked edges having both of their endpoints in C. If the size of the set is k (or $|C|$, if $|C| < k$) we move to the next maximal clique. Otherwise, we one by one pick edges of C and add to the selected set of edges until the set of vertices of the picked edges having both of their endpoints in C reaches k (or $|C|$, if $|C| < k$). We won't get into a situation where the set of vertices of the picked edges having both of their endpoints in C exceeds k (due to the clique intersection property of the clique-tree).

We write a python program for the construction of chordal graphs, background knowledge edges, and implementation of Algorithm 3. The experiments use the open source `networkx` (Hagberg et al. [2008]) package.

n	k	$\lvert\mathcal{K}_1\rvert$	$\lvert\mathcal{K}_2\rvert$	T_1	T_2
600	6	58	73	95	92
600	6	52	65	192	190
700	7	47	67	144	144
700	7	49	74	132	128
800	8	56	83	199	195
800	8	55	85	200	195
900	9	46	89	258	256
900	9	39	68	257	252
1000	10	65	116	370	353
1000	10	75	137	346	338
1100	11	55	121	467	455
1100	11	51	104	460	453

Table 1: Exploring runtime dependence on number of background knowledge edges: detailed table

Effect of changing the size of the background knowledge while keeping k fixed Here we describe the construction of the background knowledge edge sets $\mathcal{K}_1$ and $\mathcal{K}_2$ used in table 1. We first construct $\mathcal{K}_1$ as above. We then construct $\mathcal{K}_2$ by adding a few edges to $\mathcal{K}_1$ in such a way that the value of k does not change. We do this experiment for different n and k, where n is the number of nodes of the chordal graph, and k is the maximum number of vertices of any clique of the chordal graph that is part of a background knowledge edge that lies completely inside that clique, as defined earlier also. Table 1 gives a more detailed version of table 1 (given in the main section of the paper) with more dataset points.

We can also see from the table that the running time decreases slightly by increasing the size of background knowledge edges. This is because as the number of background knowledge increases the number of background consistent permutations of any clique decreases.

References

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

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

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

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

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