Graph-based Complexity for Causal Effect by Empirical Plug-in

Rina Dechter[1] **Anna K. Raichev**[1] **Jin Tian**[2] **Alexander Ihler**[1]
[1]University of California, Irvine [2]Mohamed bin Zayed University of Artificial Intelligence

Abstract

This paper focuses on the computational complexity of computing empirical plug-in estimates for causal effect queries. Given a causal graph and observational data, any identifiable causal query can be estimated from an expression over the observed variables, called the estimand. The estimand can then be evaluated by plugging in probabilities computed empirically from data. In contrast to conventional wisdom which assumes that high dimensional probabilistic functions will lead to exponential evaluation time, we show that estimand evaluation can be done efficiently, potentially in time linear in the data size, depending on the estimand's hypergraph. In particular, we show that both the *treewidth* and *hypertree width* of the estimand's structure bound the evaluation complexity, analogous to their role in bounding the complexity of inference in probabilistic graphical models. In settings with high dimensional functions, the hypertree width often provides a more effective bound, since the empirical distributions are sparse.

1 INTRODUCTION

Given a causal graph and data from the observed distribution of a Structural Causal Model (SCM), a causal effect query can be answered by a two step process: First, determine if the query is *identifiable*, namely if it can be answered uniquely given the graph and observational probabilities, and if so generate an estimand for the query. The estimand is an algebraic expression over probabilistic functions involving observed variables only. Second, the estimand is evaluated using data from the observational distribution. A straightforward approach is what we will call the empirical "plug-in" method, which simply uses the empirical probabilities extracted from the data as estimates for the probabilistic quantities in the estimand.

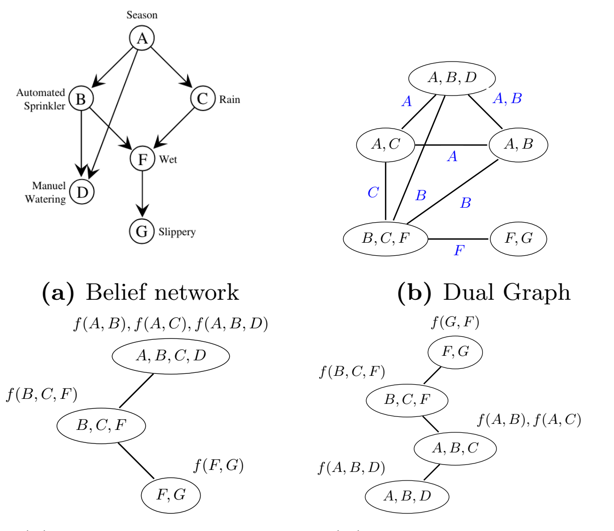
However, the estimand expression often involves high dimensional conditional probability functions. A common assumption is that, for discrete models, the computation required is at least the size of the largest table, and thus exponential in the number of arguments of the largest term in the expression, making an exact evaluation appear too computationally expensive. However, we show that by taking into account the data size, the empirical probability table sizes are bounded, even when the functions' dimension is very high. Consequently, estimand evaluation can be bounded far more effectively as a function of the data size, and in many cases even linearly bounded, which is what our paper explores.

It is well known that probabilistic inference is exponential in the tree-width (Dechter, 2003, 2013). However, when a graphical model's functions are sparse (e.g., have many zeros), the *hypertree width* can provide a tighter bound than the tree-width (Gottlob et al., 2000; Kask et al., 2005; Otten and Dechter, 2008). Building on these results, we show that the tree-width and hypertree width play a similar role in the complexity of estimand evaluation. In particular, since the empirical probabilities are inherently sparse, the hypertree width is often far more effective than the tree-width in guiding estimand evaluation algorithms and bounding their complexity.

Specifically, we associate an estimand expression with a subexpression hierarchy and show how the hypertree widths of these subexpressions additively determines complexity bounds on estimand evaluation. A key novelty is extending these bounds to sum-product queries involving ratios. We illustrate the effectiveness of the hypertree width in capturing the actual time and memory costs of plug-in estimation.

The paper is organized as follows. Section 2 provides necessary background on Structural Causal models (SCMs) and Bayesian networks(BNs), as well as on

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).



(a) Belief network **(b)** Dual Graph

(c) Tree decomposition 1 **(d)** Tree decomposition 2

Figure 1: A Bayesian Network & Tree Decomposition

the role of tree-width and hyperwidth in characterizing complexity for reasoning. Section 3 provides the underlying theory, estimation algorithm and analysis, Section 4 provides empirical illustration, and Section 5 concludes.

2 BACKGROUND

2.1 Structural Causal Models

Definition 1 (Structural Causal Model) *A structural causal model (SCM) (Pearl, 2009) is a 4-tuple $\mathcal{M} = \langle \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{F}, P(\boldsymbol{U})\rangle$ where:*

- *$\boldsymbol{U} = \{U_1, U_2, ..., U_k\}$ is a set of exogenous (latent) variables determined by factors outside the model;*
- *$\boldsymbol{V} = \{V_1, V_2, ..., V_n\}$ is a set of endogenous (observed) variables whose values are determined by other variables in the model;*
- *$\boldsymbol{F} = \{f_i : V_i \in \boldsymbol{V}\}$ is a set of functions f_i, also called mechanisms, where each f_i is a process by which V_i is assigned a value as a function of V_i's causal parents $PA_i \subseteq \boldsymbol{U} \cup (\boldsymbol{V} \setminus V_i)$ so that $V_i \leftarrow f_i(V_i, PA_i)$;*
- *$P(\boldsymbol{U})$ is a probability distribution over the latent variables, assumed to to be mutually independent, i.e., $P(\boldsymbol{U}) = \prod_{U \in \boldsymbol{U}} P(U)$.*

The SCM $\mathcal{M}$ defines a probability distribution $P(U, V)$

$$P(\boldsymbol{V}, \boldsymbol{U}) = \prod_{V_i \in \boldsymbol{V}} f(V_i, PA_i) \cdot \prod_{U_j \in \boldsymbol{U}} P(U_j).$$

The observational distribution, $P(\boldsymbol{V})$, is given by

$$P(\boldsymbol{V}) = \sum_{\boldsymbol{U}} P(\boldsymbol{V}, \boldsymbol{U}).$$

Causal diagrams. The *causal diagram* of an SCM $\mathcal{M}$ is a directed graph $\mathcal{G} = \langle \boldsymbol{V} \cup \boldsymbol{U}, E\rangle$, where each node represents a variable, and an arc from node X to Y is in E iff X is a parent of Y. We assume semi-Markovian SCMs in which latent variables connect to at most two observable variables (Tian, 2002). As is common, we omit latent variables with a single child, and replace any other latent variable with a bidirectional dashed arc between its children (see Figures 2a, 2b, 2c).

Causal effect and the truncation formula. An external intervention forcing variables $\boldsymbol{X}$ to take on value $\boldsymbol{x}$, called $do(\boldsymbol{X} = \boldsymbol{x})$, is modeled by replacing the mechanism for each $X \in \boldsymbol{X}$ with the function $X = x$. Formally,

$$P(\boldsymbol{V} \setminus \boldsymbol{X}, \boldsymbol{U} \mid do(\boldsymbol{X} = \boldsymbol{x})) = \prod_{V_j \notin \boldsymbol{X}} f_j(V_j, PA_j) \cdot P(\boldsymbol{U}) \Big|_{\boldsymbol{X} = \boldsymbol{x}}$$

The effect of $do(\boldsymbol{X})$ on a variable Y, denoted $P(Y | do(\boldsymbol{X}))$, is defined by marginalizing over the truncated distribution:

$$P(\boldsymbol{Y} | do(\boldsymbol{X} = \boldsymbol{x})) = \sum_{(V \cup U) \setminus Y} P(\boldsymbol{V} \setminus \boldsymbol{X}, \boldsymbol{U} \mid do(\boldsymbol{X} = \boldsymbol{x}))$$

The standard formulation of causal inference assumes that we only have access to the causal graph G and the observational distribution $P(\boldsymbol{V})$ (or a sample from it). The identifiability task is to determine if the query can be *uniquely* answered from G and $P(\boldsymbol{V})$ (Pearl, 2009). If it is, then an estimand expression in terms of $P(\boldsymbol{V})$ can be generated and evaluated.

Definition 2 (Causal-effect query) *Given a causal diagram $\mathcal{G} = \langle \boldsymbol{V} \cup \boldsymbol{U}, E\rangle$, data samples from the observational distribution $P(\boldsymbol{V})$, and an identifiable query $P(\boldsymbol{Y} \mid do(\boldsymbol{X}))$, the task is to compute the value of $P(\boldsymbol{Y} \mid do(\boldsymbol{X}))$.*

Estimand-based approaches. The now-standard methodology for answering causal-effect queries is to break the task into two steps. The first is the *identifiability step*: given a causal diagram and a query $P(\boldsymbol{Y} \mid do(\boldsymbol{X}))$, determine if the query is identifiable and if so, generate an *estimand*, or algebraic expression in terms of the observational distribution $P(\boldsymbol{V})$ that answers the query. A complete polynomial algorithm called ID has been developed for this task (Tian, 2002; Shpitser and Pearl, 2006). The second step is *estimation*: use samples from the observational distribution $P(\boldsymbol{V})$ to estimate the value of the estimand expression.

A number of approaches have been applied to estimation. A simple approach, called the *plug-in estimator*, estimates each term in the estimand with its empirical probability value obtained from the data. More sophisticated approaches have been developed recently (Jung et al., 2020a,b; Raichev et al.).

2.2 Bayesian networks and Tree Decompositions

An SCM $\mathcal{M}$'s probability distribution is also a **Bayesian Network (BN)**:

Definition 3 (Bayesian network (BN)) *A Bayesian network is a 3-tuple $\mathcal{B} = \langle \boldsymbol{X}, \boldsymbol{D}, \mathbf{P_G} \rangle$ where G is a directed acyclic graph over a set of variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ defined over domains $\boldsymbol{D} = \{D_1, \ldots, D_n\}$. The functions $\mathbf{P_G} = \{P_1, \ldots, P_n\}$ are conditional probability tables (CPTs), $P_i = P(X_i \mid PA_{X_i})$, where PA_{X_i} are the parents of X_i in G. The Bayesian network $\mathcal{B}$ represents the probability distribution $P_{\mathcal{B}}(X) = \prod_{i=1}^{n} P(X_i \mid PA_{X_i})$.*

The ***dual graph*** *of a BN is a graph whose nodes are the scopes, or arguments, $\{S_1, ..., S_r\}$ of the CPTs, with edges connecting any two nodes whose scopes share variables. The* ***hypergraph*** *of a Bayesian network has variables as its nodes and the hyperedges are the sets of scopes of the functions. We visualize a hypergraph using its dual graph since they are isomorphic, where each hyperedge maps to a node in the dual graph. See Figures 1a and 1b.*

Tree decomposition schemes have been widely used for constraint processing and probabilistic reasoning (Dechter, 2003; Gottlob et al., 2000; Dechter, 2013). The methods vary somewhat in their graph definitions as well as the way the tree decomposition is processed.

Definition 4 (tree & hypertree decompositions) *(Kask et al., 2005; Gottlob et al., 2000) A* ***tree-decomposition*** *of a graphical model $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F} \rangle$ is a triplet $\tau = \langle T, \chi, \psi \rangle$, where $T = (\boldsymbol{V}, E)$ is a tree, and χ and ψ are labeling functions that associate with each vertex $v \in \boldsymbol{V}$ two sets, $\chi(v) \subseteq \boldsymbol{X}$ and $\psi(v) \subseteq \boldsymbol{F}$, that satisfy the following conditions:*

1. *For each function $f_i \in \boldsymbol{F}$, there is* ***exactly one*** *vertex $v \in \boldsymbol{V}$ such that $f_i \in \psi(v)$.*
2. *If $f_i \in \psi(v)$, then $scope(f_i) \subseteq \chi(v)$.*
3. *For each variable $V_i \in \boldsymbol{V}$, the set $\{v \in \boldsymbol{V} | V_i \in \chi(v)\}$ induces a connected subtree of T. This is also called the running intersection or connectedness property.*

The ***treewidth*** *w of a tree-decomposition $\tau = \langle T, \chi, \psi \rangle$ is $\max_{v \in V} |\chi(v)| - 1$. The treewidth of a Bayesian network is the minimum treewidth over all its tree-decompositions.*

The tree-decomposition τ is a ***hypertree decomposition*** *if it also satisfies that the variables in each node, called a cluster, are covered by the arguments of the functions in the node. Formally:*

4. *For each $v \in \boldsymbol{V}$, $\chi(v) \subseteq \cup_{f_j \in \psi(v)} scope(f_j)$.*

A hypergraph is a hypertree if its dual graph can be made into a tree by removing edges while satisfying the running intersection property. The ***hypertree width*** *of τ is $hw = max_v |\psi(v)|$. The hypertree width of a Bayesian network is the minimum hypertree width over all its hypertree decompositions.*

Finding tree or hypertree decompositions having minimal treewidth or hyperwidth is known to be NP-hard. Heuristic algorithms are employed in practice for both parameters; for details see, e.g., Dechter (2003); Kask et al. (2011); Gottlob et al. (2014); Gogate and Dechter (2012).

Example 1 *(Taken from Kask et al. (2005))* Consider the Bayesian network in Figure 1a. Its dual graph is given in Figure 1b and two of its hypertree decompositions in Figure 1c and 1d. Here, the labeling χ is the set of variables in each node. The functions can be assigned to nodes whose χ variables contain their scopes. For example, any function with scope $\{G\}$ must be placed in vertex (GF). In contrast, a function with scope $\{F\}$ may be placed either in vertex (FG) or (BCF).

Relational vs. tabular representation. The common method for specifying functions as a table is exponential in their scopes, where the base of the exponent is the maximum domain size k. Alternatively, we can assume a relational specification where all tuples that are mapped to value "0" (e.g., zero probabilities) are removed from the table. In this case, a function over a scope of l variables is specified by t tuples of length l. The value t is often referred to as the *tightness* of the function representation. Clearly, relational specification size can be far smaller than the worst-case $O(k^l)$. Henceforth we assume that functions are always specified relationally.

Definition 5 (tightness) *The tightness of a function f, $t(f)$, is the number of non-zero configuration in its domain. The tightness of a graphical model having the set of functions F is $t = max_{f \in F} t(f)$.*

Once a hypertree decomposition of a BN is generated, any sum-product query (e.g., $P(\boldsymbol{X}|\boldsymbol{Y} = \boldsymbol{y})$ where $\boldsymbol{X}$

Algorithm 1 Cluster Tree Evaluation (CTE)

Input: $\mathcal{M}=\langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}\rangle$, decomposition $\langle T, \chi, \psi\rangle$. Query variables $\boldsymbol{Y}$
Output: Function $O(\boldsymbol{Y})$

```
0. Designate root cluster $r$ such that $\boldsymbol{Y} \subseteq \chi(r)$ and
   generate rooted tree $T_r$.
1. **For** each cluster $u \in T_r$ from leaves to root **do**
2.   Combine original functions $h(u) \leftarrow \prod_{f\in\psi(u)} f$
3.   Let $v$ be the parent of $u$ in $T_r$
4.   Initialize $h_{(u,v)} \leftarrow h(u)$
5.     **For** each child $c$ of u in $T_r$ **do**
6.       $h_{(u,v)} \leftarrow h_{(u,v)} \cdot \sum_{\chi(c)\setminus\chi(u)} h_{(c,u)}$
7.     **End For**
8. **End For**
9. **Return** output function computed at cluster $r$
     $O(\boldsymbol{Y}) = \sum_{\chi(r)\setminus\boldsymbol{Y}} \prod_{h\in cluster(r)} h$
```

and $\boldsymbol{Y}$ are subsets of variables) can be answered by a message passing algorithm where each node of the tree sends a function to each of its neighbors. We sometimes call nodes of a tree-decomposition "clusters".

We use Cluster-tree Elimination (CTE), described in the sequel for completeness, as a generic version of one such message passing algorithm.

Algorithm CTE. (Kask et al., 2005) CTE is described in Algorithm 1, relative to a given query. Given a tree decomposition, T, we designate one cluster as *the output cluster* from which the query answer can be obtained. The algorithm treats T as a rooted tree T_r whose root is the output cluster. The rooted tree inspires child-parent relationships between the clusters.

Starting at the leaves, each node u computes a message to be sent to its parent v, denoted $h_{(u,v)}$. In step 2 it multiplies all input functions $\psi(u)$, yielding $h(u)$. This can be accomplished in time and space $O(t^{|\psi(u)|})$ for a relational specification, and its tightness is $O(t^{|\psi(u)|})$.

Step 4 initializes the message to v, which is the parent of u (step 3). Steps 5-8 form the core of the algorithm. The algorithm iterates over the messages sent from each child cluster (which are already computed), absorbing each into its own outgoing message to the parent v. Once the root receives all its messages, it computes the answer to the query (step 9).

Theorem 1 (Graph-based complexity of CTE) *(Kask et al., 2005) Given a graphical model $\mathcal{M} = \langle \boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}\rangle$ and a hypertree-decomposition $\langle T, \chi, \psi\rangle$, let n be the number of variables in $\boldsymbol{X}$, w its treewidth, hw its hypertree width, t its tightness and k the maximum domain size of a variable. A sum-product query can be computed by CTE in time and space satisfying the following two bounds:*

1. *as a function of the treewidth:*
$$O(n \cdot k^{w+1}) \text{ time and } O(nk^{w}) \text{ space}$$
2. *as a function of the hyperwidth:*
$$O(n \cdot hw \cdot \log t \cdot t^{hw}) \text{ time and } O(t^{hw}) \text{ space}$$

The alternative option of AND/OR search. It was shown (Dechter and Mateescu, 2007) that AND/OR search spaces can also capture problem decomposition in the search space, and lead to search algorithms that exploit tree and hypertree decompositions. Such algorithms can lead to similar complexity bounds (Dechter et al., 2008). While in the sequel we refer only to performing CTE over tree-decompositions, search schemes are equally applicable.

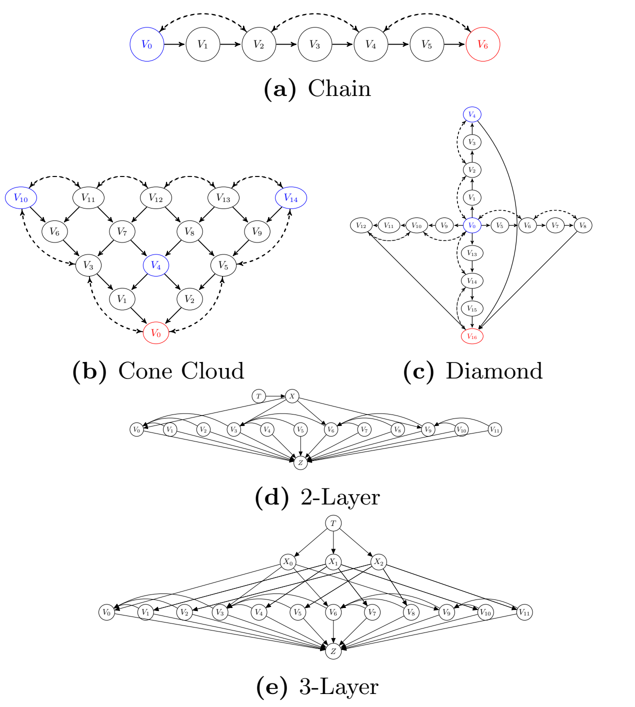
3 CAUSAL EFFECT ESTIMAND EVALUATION

Building on tree decompositions and the CTE algorithm, we introduce the notion of empirical Bayesian networks which is central to the plug-in evaluation of causal queries' estimands.

Empirical Bayesian Network. Given a directed graph G whose nodes are discrete variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ and given a data set $\mathcal{D} = \{d_1, \ldots d_t\}$ over the variables, the *empirical Bayesian network*, $empBN(G, \mathcal{D})$, is the Bayesian network whose graph is G and whose functions are the *empirical CPTs* extracted from the dataset $\mathcal{D}$. That is, any entry (x, pa_X) in the empirical CPT $P_{\mathcal{D}}(X_i = x|PA_{X_i} = pa_X)$ for a variable X_i and its parents PA_{X_i} in G is obtained by counting the number of appearances of (x, pa_X) in $\mathcal{D}$, divided by the number of appearances of pa_X in $\mathcal{D}$. Formally, $P_{\mathcal{D}}(x|pa_X) = \frac{\#\mathcal{D}(x,pa_X)}{\#\mathcal{D}(pa_X)}$ where $\#\mathcal{D}(s)$ is the number of elements in $\mathcal{D}$ that are consistent with s.

It is easy to see that the number of non-zero configurations of any CPT table of $empBN(G, \mathcal{D})$ cannot exceed the data size, t. By Theorem 1, we can immediately conclude the following two corollaries.

Corollary 1 (Complexity of empirical BN.) *Given a DAG G over variables $\{X_1, ..., X_n\}$, let hw and w be the hypertree width and treewidth of the Bayesian network, respectively. Given also a dataset $\mathcal{D}$, where $t = |\mathcal{D}|$, and domain sizes bounded by k, then computing a sum-product query over $empBN(G, \mathcal{D})$ is (w-based) $O(n \cdot k^{w+1})$ time and $O(k^{w})$ space, and (hw-based) $O(n \cdot hw \cdot \log t \cdot t^{hw})$, time and $O(t^{hw})$ space.*


(a) Chain

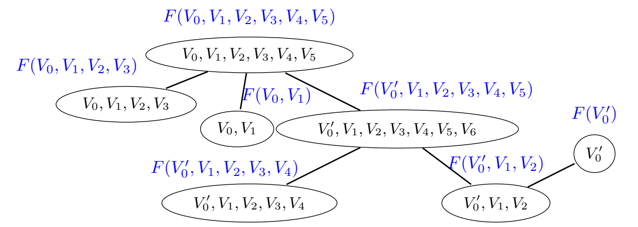
(b) Cone Cloud

(c) Diamond

(d) 2-Layer

(e) 3-Layer

Figure 2: Causal Graphs



$F(V_0,V_1,V_2,V_3,V_4,V_5)$
V_0,V_1,V_2,V_3,V_4,V_5
$F(V_0,V_1,V_2,V_3)$
V_0,V_1,V_2,V_3
$F(V_0,V_1)$
V_0,V_1
$F(V_0',V_1,V_2,V_3,V_4,V_5)$
$V_0',V_1,V_2,V_3,V_4,V_5,V_6$
$F(V_0')$
V_0'
$F(V_0',V_1,V_2,V_3,V_4)$
V_0',V_1,V_2,V_3,V_4
$F(V_0',V_1,V_2)$
V_0',V_1,V_2

Figure 3: Hypertree for Eq. (2) with $hw = 1$.

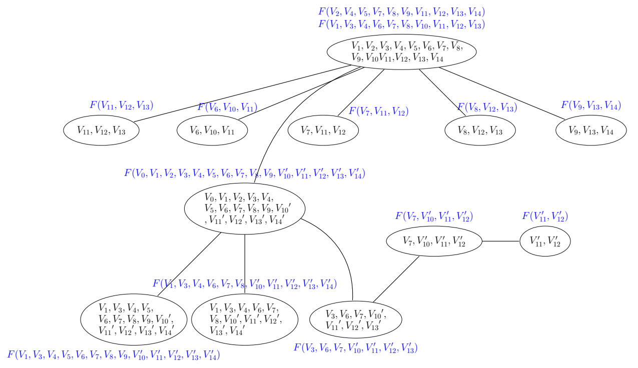


Figure 4: Cone-cloud Estimand's Hypertree with $hw = 2$.

Corollary 2 *If empBN has hypertree width 1, a sum-product query can be answered in almost linear time: $O(n \cdot t \log t)$.*

Implication to estimands evaluation. We next show that a causal effect estimand can be associated with a hierarchy of empirical Bayesian networks, so that evaluating the sum-product on each will accomplish the estimand evaluation. Applying a hypertree decomposition algorithm like CTE to each sub-expression yields an estimand evaluation scheme whose performance is bounded by the graph parameters of treewidth and hyperwidth. We first illustrate via several examples.

Example 2 *Consider the model in Figure 2a. To evaluate the query $P(V_6 \mid do(V_0))$, the ID algorithm (Tian, 2002; Shpitser and Pearl, 2006) generates the expression:*

$$P(V_6 \mid do(V_0)) = \sum_{V_1,V_2,V_3,V_4,V_5} P(V_5|V_0,V_1,V_2,V_3,V_4)\times$$
$$P(V_3|V_0,V_1,V_2)\,P(V_1|V_0)\sum_{V_0} P(V_6|V_0,V_1,V_2,V_3,V_4,V_5)$$
$$\times\, P(V_4|V_0,V_1,V_2,V_3)P(V_2|V_0,V_1)\,P(V_0). \quad (1)$$

Renaming the summation variables in the second sum and moving all summations to the head of the expression gives an equivalent expression (e.g., we rename V_0 to V_0' in the inner sum), yielding:

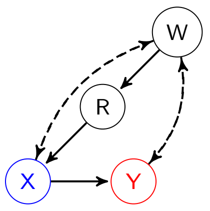
$$P(V_6 \mid do(V_0)) = \sum_{V_1,V_2,V_3,V_4,V_5,V_0'} P(V_5|V_0,V_1,V_2,V_3,V_4)$$
$$\times\, P(V_3|V_0,V_1,V_2)\,P(V_1|V_0)P(V_6|V_0',V_1,V_2,V_3,V_4,V_5)$$
$$\times\, P(V_4|V_0',V_1,V_2,V_3)\,P(V_2|V_0',V_1)\,P(V_0'). \quad (2)$$

The latter expression corresponds to a sum-product reasoning task over a Bayesian network defined by the 7 CPTs in Eq. (2). A hypertree decomposition with $hw = 1$ is given in Figure 3. By Theorem 1, the expression can be computed by CTE in time $O(n \cdot \log t \cdot t)$ where t is the data size (so t bounds the tightness of the Empirical BN). In contrast the decomposition has treewidth $w = 6$, a potentially loose bound. Generally, for chains of length n, the treewidth is $w = n - 1$, giving an exponential bound of $O(n \cdot k^n)$ while hyperwidth stays $hw = 1$. Clearly, hyperwidth provides a far tighter and more informative bound for large chains.

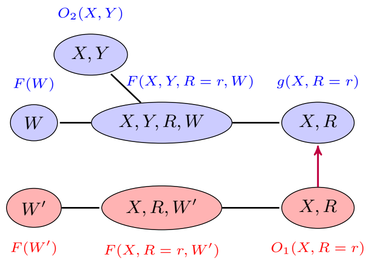
Example 3 *Consider the estimand expression in Eq. (3), obtained from a 15 variable cone-cloud graph (Figure 2b). After renaming variables and moving summations to the head of the first sum, we obtain the expression Eq. (4). Again, this can be viewed as a sum-product inference over an empirical BN. Although the hypergraph is not a hypertree, but can be embedded in a hypertree of $hw = 2$, depicted in Figure 4 (the top cluster includes 2 functions). Therefore the complexity of evaluating this expression by CTE is $O(n \cdot \log t \cdot t^2)$. The treewidth is $w = 14$ (there is a cluster with 15 variables) leading to a bound of $O(nk^{15})$ time.*

$$P(V_0|do(V_{14},V_{10},V_4)) = \sum_{V_1,V_2,V_3,V_5,V_6,V_7,V_8,V_9,V_{11},V_{12},V_{13}} P(V_2|V_4,V_5,V_7,V_8,V_9,V_{11},V_{12},V_{13},V_{14})P(V_9|V_{13},V_{14})\times$$
$$P(V_8|V_{12},V_{13})P(V_1|V_3,V_4,V_6,V_7,V_8,V_{10},V_{11},V_{12},V_{13})P(V_7|V_{11},V_{12})P(V_6|V_{10},V_{11})P(V_{11},V_{12},V_{13})\times$$
$$\sum_{V'_{10},V_{11},V_{12},V_{13},V'_{14}} P(V_0|V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,V'_{10},V_{11},V_{12},V_{13},V'_{14})P(V_3,V_{13}|V_6,V_7,V'_{10},V_{11},V_{12})P(V_{11},V_{12})\times$$
$$P(V_5|V_1,V_3,V_4,V_6,V_7,V_8,V_9,V'_{10},V_{11},V_{12},V_{13},V'_{14})P(V'_{14}|V_1,V_3,V_4,V_6,V_7,V_8,V'_{10},V_{11},V_{12},V_{13})P(V'_{10}|V_7,V_{11},V_{12}) \quad (3)$$

$$P(V_0|do(V_{14},V_{10},V_4)) = \sum_{V_1,V_2,V_3,V_5,V_6,V_7,V_8,V_9,V_{11},V_{12},V_{13},V'_{10},V'_{11},V'_{12},V'_{13},V'_{14}} P(V_2|V_4,V_5,V_7,V_8,V_9,V_{11},V_{12},V_{13},V_{14})$$
$$\times P(V_9|V_{13},V_{14})P(V_8|V_{12},V_{13})P(V_1|V_3,V_4,V_6,V_7,V_8,V_{10},V_{11},V_{12},V_{13})P(V_7|V_{11},V_{12})P(V_6|V_{10},V_{11})P(V_{11},V_{12},V_{13})$$
$$\times P(V_0|V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,V'_{10},V'_{11},V'_{12},V'_{13},V'_{14})P(V_5|V_1,V_3,V_4,V_6,V_7,V_8,V_9,V'_{10},V'_{11},V'_{12},V'_{13},V'_{14})$$
$$\times P(V_3,V'_{13}|V_6,V_7,V'_{10},V'_{11},V'_{12})P(V'_{14}|V_1,V_3,V_4,V_6,V_7,V_8,V'_{10},V'_{11},V'_{12},V'_{13})P(V'_{10}|V_7,V'_{11},V'_{12})P(V'_{11},V'_{12}) \quad (4)$$



(a) Causal Graph



(b) Heirarchy Hypergraph

Figure 5: Napkin Model

Ratio estimands. It is sometimes the case that estimands include ratios of expressions. How should we treat those? One option is to treat a denominator expression as a distinct sum-product expression which connects to its numerator by its "output function". This suggests a *sub-expressions hierarchy* which dictates a sum-product processing order. The components of the hierarchy are flattened sum-product subexpressions as seen in the previous examples, where each corresponds to a sum-product expression over a Bayesian network. Each layer in the hierarchy is a single expression and its sum-product output function is processed by *CTE* over its own hypertree decomposition. These output functions connect the layers in a chain hierarchy, where the output function from each layer is included in the sub-expression of its parent layer.

Example 4 *Consider the ratio estimand expression*[1] *associated with the Napkin model of Figure 5a.*

$$P(Y|do(X)) = \frac{\sum_W P(X,Y|R,W)P(W)}{\sum_{W'} P(X|R,W')P(W')}$$

The denominator expression is composed of two functions whose scopes are the two sets $\{W'\}$ and $\{X,R,W'\}$. The corresponding dual graph is depicted under the red arrow in Figure 5b. The output function for this sum-product is a new function $O_1(X,R)$ whose inverse $g(X,R) = \frac{1}{O_1(X,R)}$ will be noted as a factor of its numerator. The red arrow emanating from node O_1 with arguments $\{X,R\}$ indicates that the function will be generated in that child layer and will be used in the parent layer. The sum-product at the top layer of the hierarchy has four nodes corresponding to four functions: the two in the numerator, the function $g(X,R)$ generated by the child expression, and the output function O_2 produced as a result of the summation over W. Since the denominator expression has $hw = 1$, the tightness of the output function $g(X,R)$ remains bounded by $t^1 = t$ (the data size). Since the numerator's expression, including output function $g(X,R)$, also has $hw = 1$, the overall hyperwidth of the hierarchy of this estimand is $hw = 1$, and since the $g(\cdot)$ has the same tightness as the input functions, the computation remains (near) linear in the data size.

Due to the flattening process, each sum-product expression will have a single sum-product child from a denominator (see the hierarchy graph in Figure 5b). We next formalize the needed concepts.

Definition 6 (sum-product chain hierarchy) *Given an estimand whose expressions are all flattened, so the nodes of the sum-product hierarchy are flat sum-product expressions, and the top layer is the numerator sum-product expression. Each denominator sum-product expression is a child of its direct numerator sum-product expression. The function generated by a sum-product computation is called its* output function.

[1]Notice that here $P(Y|do(X)))$ appears to be invariant to the value of R, so we can assign any value R = r during the CTE step of the algorithm as shown in Figure 5b.

Algorithm 2 describes our plug-in hypertree evaluation (PI-HTE) algorithm. Its input is an estimand and a dataset. The first step is flattening, where for each sum-product expression the summation variables are renamed in all the functions within the summation's scope. Step 2 creates the sum-product chain hierarchy as defined above; again, note that the output function of level $i-1$ is included as a factor in the dual graph of level i and thus is in the hypertree decomposition of layer i. In steps 3-10 the algorithm processes the sum-product hierarchy in order from leaf to root. When layer i is processed, its empirical BN is well defined. Then a hypertree decomposition is created by a hypertree decomposition scheme (Gottlob et al., 2014), taking into account the output function obtained from its child level. CTE can now be applied, producing the output function O_i. The output function is then inverted (step 9) and incorporated into the BN and the dual graph of the parent, and computation continues with the parent layer. The answer is obtained in the output function of the root layer. It is easy to see that,

Algorithm 2 Plug-In Hypertree Evaluation (PI-HTE)

Input: data $\mathcal{D}$, and estimand expression for $P(\boldsymbol{Y} \mid do(\boldsymbol{X}))$ over a given DAG G.
Output: an estimate of $P(\boldsymbol{Y} \mid do(\boldsymbol{X}))$

```
1.  Flatten sum-products by renaming and moving
      all summations to the front.
2.  Generate a hierarchy of sum product expressions.
3.  Let $P$ be the lowest sum-product expression.
4.  While $P$ has a parent expression do
5.    Let $G$ be a DAG of BN defined by functions in $P$
6.    $empBN \leftarrow$ Generate $empBN(G, D)$
7.    $H_P \leftarrow$ Generate a hypertree for $empBN$
        aiming for $\min(hw_P)$.
8.    $O_P \leftarrow$ apply CTE to tree decomposition $H_P$
9.    $g_P \leftarrow \frac{1}{O_P}$; $parent(P) \leftarrow parent(P) \cup \{g_P\}$
10.   $P \leftarrow parent(P)$
11. EndWhile
12: Return   $O(root)$ and compute $P(\boldsymbol{Y} \mid do(\boldsymbol{X}))$
```

Theorem 2 (soundness) *Algorithm PI-HTE is sound for estimand evaluation.*

Complexity. We associate the dual graph of the BN at level i with a hypertree decomposition having hyperwidth hw_i and treewidth w_i. The complexity of computing the output function in each level is dominated by t^{hw_i} and k^{w_i}. Thus, the overall complexity is controlled by the treewidths and hypertree widths over all the levels. This seems to imply that complexity is exponential in the largest hw and the largest w; however, for hypertree width the complexity can be more involved. In particular, the tightness of the output functions may no longer be bounded by the data size t; the worst-case tightness of O_i is instead t^{hw_i}. Since the dominant factors in the complexity are t^{hw}, and k^w, for simplicity we will drop the $\log t$ factor appearing in the complexity bound. The following theorem summarizes.

Theorem 3 (complexity) *Given an estimand expression having a sum-product hierarchy of depth l, algorithm PI-HTE has time and space complexity $O(n \cdot t^{\sum_{i=1}^{l} hw_i - l+1})$, where n is the number of variables, t the tightness, and hw_i is the hypertree width of the expression at level i. The algorithm's complexity is also $O(n \cdot k^{max_{i=1}^{l}\{(w_i)\}})$ where w_i is the treewidth of the expression at level i.*

Proof 1 *(proof by induction) Case 1: It is clear that if we have no denominators the estimand has only one level. This defines a sum-product product expression. Since it can be viewed as an empirical Bayesian network having tightness t, it can be processed in $O(n \cdot t^{hw})$ (Theorem 1), or in $O(n \cdot t^{w+1})$. This proves the base case of $l = 1$. When $l > 1$ the inductive hypothesis for level $l-1$ assumes that the effective hyperwidth for that level is $\sum_{j=1}^{l-1}(hw_j - l + 2)$. This implies that its output function, O_{l-1}, computed at level $l-1$ has a tightness also bounded by $t^{\sum_{j=1}^{l-1} hw_j - l+2}$. The output function O_{l-1} is placed at the parent level l, and grouped in one of the clusters with at most $hw_l - 1$ functions. The product of $hw_l - 1$ functions of tightness t with the single O_{l-1} function having tightness $t^{\sum_{j=1}^{l-1} hw_j - l+2}$ requires computation of at most $O(t^{hw_l - 1} \cdot t^{\sum_{j=1}^{l-1} hw_j - l+2}) = O(t^{\sum_{j=1}^{l} hw_j - l+1})$. Thus the effective hyperwidth grows at most additively with each level in both time and memory, yielding the claim. The bound based on treewidth is not dependent on the tightness but only on the domain size, and we therefore see that the relevant treewidth that dominates the computations is simply the maximum treewidth over the whole hierarchy.*

Note, however, that in the special case that O_{i-1} resides in its own cluster in the tree decomposition of its parent level i, the effective hyperwidth of level i is simply the max between the hyperwidth of the child expression and its parent expression. If this occurs at each level, the effective hyperwidth of the full computation is just the maximum hyperwidth over all levels, resulting in bound $O(t^{\max_{j=1}^{l} hw_j})$.

Table 1: Empirical plug-in results for different sample sizes. Here t is the tightness of the data, and k is the variables' domain sizes.

(a) Chain: $P(V_{99}|do(V_0))$
$|\boldsymbol{V}| = 99$; $|\boldsymbol{U}| = 49$ **hw** $= 1$, **w** $= 98$
$P(V_{98}|V0, \dots V_{97})$- largest factor

#Samples	time	Max table size	t	density
100	11.2	100	100	9.9×10^{-34}
200	19.7	200	200	2.0×10^{-33}
400	31.4	400	400	3.9×10^{-33}
800	53.7	800	800	7.9×10^{-33}
1,000	69.5	1,000	1,000	1.0×10^{-32}
1,600	97.5	1,600	1,600	1.6×10^{-32}
3,200	192.4	3,200	3,200	3.2×10^{-32}
6,400	369.2	6,400	6,400	6.4×10^{-32}
10,000	608.8	10,000	10,000	9.9×10^{-32}

(b) Cone Cloud $P(V_0|do(V_{14}, V_{10}, V_4))$
$|\boldsymbol{V}| = 15$; $|\boldsymbol{U}| = 8$; $k = 10$; **hw** $= 2$, **w** $= 14$
largest factor $P(V_0|V_1 \dots V_{14})$

#Samples	time (s)	Max table size	t	density
100	2.2	9,900	100	1e-13
200	3.3	39,203	200	2e-13
400	7.3	152,877	400	4e-13
800	28.8	589,815	800	8e-13
1,000	39.1	882,604	1,000	1e-12
1,600	84.8	2,206,528	1,600	1.6e-12
3,200	280.3	7,553,700	3,200	3.2e-12
6,400	788.74	21,949,125	6,400	6.4e-12
10,000	1,594.2	40,404,630	10,000	1e-11

(c) Diamond Model, $P(V_{16}|do(V_0, V_4))$
$|\boldsymbol{V}| = 17$; $k = 10$; **hw** $= 2$, **w** $= 16$
$P(V_{16}|V_0, \dots V_{15})$- largest factor

#Samples	time (s)	Max table size	t	density
100	0.6	8,672	100	1×10^{-15}
200	1.9	35,605	200	2×10^{-15}
400	6.9	156,160	400	4×10^{-15}
800	26.4	603,790	800	8×10^{-15}
1,000	39	928,510	1,000	1×10^{-14}
1,600	87.4	2,184,880	1,600	1.6×10^{-14}
3,200	299.4	7,600,350	3,200	3.2×10^{-14}
6,400	988.8	22,208,750	6,400	6.4×10^{-14}
10,000	2,037.5	39,943,750	10,000	1×10^{-13}

(d) 3-Layer Model, $P(\mathbf{V}|do(Z, T))$
$|\boldsymbol{V}| = 17$; $2 \leq k \leq 50$; **hw** $= 4$, **w** $= 15$
$P(V_0|V_1, V_2, V_3, X_0)$- largest factor

#Samples	time (s)	Max table size	t	density
25	2.8	3,219	25	0.00125
50	26.5	79,136	50	0.0025
100	640.8	1,623,559	100	0.005
200	30,511.5	34,018,439	200	0.01

4 EMPIRICAL VALIDATION

Our empirical evaluation aims to validate that the PI-HTE algorithm's performance is aligned with our provided complexity analysis. We explore: 1. how informative are the graph parameters hypertree width (hw) and treewidth (w) to actual performance, and which is more effective; and 2. how the quality of the hypertree decomposition guiding algorithm PI-HTW impacts actual performance.

4.1 Benchmarks.

For benchmarks we used three semi-Markovian synthetic benchmarks (Figures 2a, 2b, 2c), and two Markovian (i.e., without latent confounding) (Figures 2d, 2e). The semi-Markovian examples were taken from Raichev et al., and the Markovian example was constructed to investigate higher hyperwidth. The new Markovian benchmarks called *layered networks*, introduced here, are illustrated in Figures 2d and 2e.

We generated the data over all visible variables using different distributions: uniform, deterministic, Dirichlet, and their mixtures. Queries were selected to correspond to complex estimands that could pose computational challenges.

4.2 Performance Measures

We report the time and maximum table size during the computation of our algorithm on different sample sizes. The maximum table size reflects the space needed to evaluate the estimand. We also report the tightness t, and the *density*, which is computed on the largest factor in the estimand, by $density = \frac{\text{\# entries in table}}{\text{\# of configurations in the joint table}}$, and measures the sparsity in our functions.

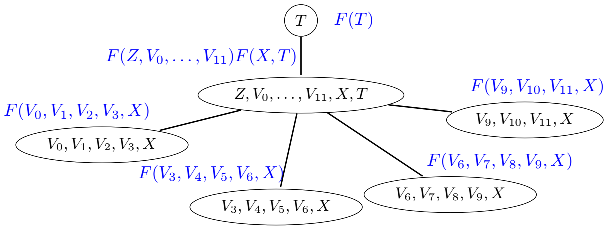
4.3 Results

The PI-HTW algorithm's performance when varying sample sizes are presented in Table 1 on the Chain, Cone-cloud, Diamond, and 3-layer models (Figure 2). We observe that the complexity for estimand evaluation time is reflected in the hyperwidth of the estimand, and provides a much tighter bound than the treewidth.

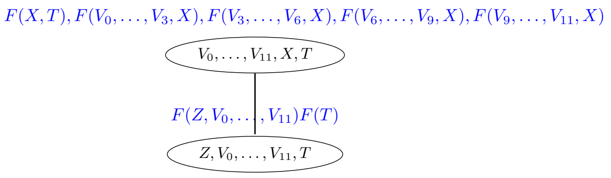
The disparity between bounds is most evident in the Chain model, where $hw = 1$ and so computation is linear in the tightness; but $w = 98$, with max domain size $k = 4$. So bounding space and time complexity via treewidth gives $O(k^{99}) \approx 4 \times 10^{59}$ time and $O(k^{98}) \approx 1{\times}10^{59}$ space, resulting in a loose and irrelevant bound. Even with 10,000 samples (so $t = 10,000$), the hyperwidth bound is $O(t^1) = O(10,000^1)$ and so far more informative.

We also show that the performance can be non-linear, for example in the case of Cone-cloud and Diamond models, both with $hw = 2$. Here the relationship between t vs time and t vs max table size are both $O(t^2)$. For example when $t = 1,000$ the max table size for the cone-cloud and diamond, respectively, is 882,604 and 928,510, just under $1,000^2$. Time also grows faster than linear in t, matching $O(t^2)$. Again, the treewidth is much higher than the hyperwidth, and the max do-

Figure 6: Two hypertrees for the 2-Layer model


T
$F(T)$
$F(Z, V_0, \ldots, V_{11})F(X, T)$
$Z, V_0, \ldots, V_{11}, X, T$
$F(V_9, V_{10}, V_{11}, X)$
V_9, V_{10}, V_{11}, X
$F(V_0, V_1, V_2, V_3, X)$
V_0, V_1, V_2, V_3, X
$F(V_3, V_4, V_5, V_6, X)$
V_3, V_4, V_5, V_6, X
$F(V_6, V_7, V_8, V_9, X)$
V_6, V_7, V_8, V_9, X

(a) 2-Layer model hypertree decomposition with hw=2


$F(X, T), F(V_0, \ldots, V_3, X), F(V_3, \ldots, V_6, X), F(V_6, \ldots, V_9, X), F(V_9, \ldots, V_{11}, X)$
$V_0, \ldots, V_{11}, X, T$
$F(Z, V_0, \ldots, V_{11})F(T)$
$Z, V_0, \ldots, V_{11}, T$

(b) 2-Layer model hypertree decomposition with hw=5

Table 2: Impact of different hyper tree decompositions

(a) 2-Layer Model, $P(Z|do(T))$
$|\boldsymbol{V}| = 15$; with hypertree decomposition w/ **hw** = 2, **w** = 14

#Samples	time (s)	Max table size	t
100	1.73	636	100
200	1.78	1,679	200
400	2.00	3,600	400
800	2.36	7,200	800
1,000	2.71	9,000	1000

(b) 2-Layer Model, $P(Z|do(T))$
$|\boldsymbol{V}| = 15$; with hypertree decomposition w/ **hw** = 5, **w** = 13

#Samples	time (s)	Max table size	t
100	2.27	8,489	100
200	4.23	62,447	200
400	36.85	812,926	400
800	348.84	8,949,996	800
1,000	775.93	20,188,458	1000

main size, k, is 10 and 50, so the treewidth bound is large and uninformative.

In the 3-layer model (Figure 2e) we observe a hyperwidth of 4 and a treewidth of 15, with a max domain size of 50. The dependence of time on t and max table size appears roughly cubic. So, while the worst-case is not realized, we do see significant impact from the larger hyperwidth. Again, however, the treewidth bound is very loose: $O(50^{15})$.

In Table 1 we also show the density of the largest function which is very small. This highlights why the hypertree width is far more effective than the treewidth for bounding the computation of these estimands

In Table 2 we present results for two different hypertree decompositions of the 2-layer model (Figure 6), to illustrate how selecting a lower hyperwidth decomposition impacts performance on the same problem instance. Indeed, we see that our algorithm's performance using the low hyperwidth decomposition was small for all sample sizes, taking less than 3 seconds. In contrast, when using the hypertree decomposition of hyperwidth 5, at 1,000 samples the algorithm took 775.93 seconds, a significant increase. The same behavior is also reflected in memory use, where the maximum table created with $hw = 2$ is 9,000, compared to 20,188,458 for $hw = 5$. This illustrates the significant impact of hyperwidth on the computation. In summary, as theory suggests, we see that organizing an estimand's evaluation around a lower hyperwidth decomposition significantly improves performance.

5 CONCLUSION

This paper provides a new algorithm and analysis for evaluating plug-in estimands for causal effect queries. Our approach uses the structural parameters of treewidth and hypertree width to efficiently organize the computation and bound its complexity, in a manner similar to the parameters' role in Bayesian networks and other graphical models. We introduce a new algorithm for Plug-in hypertree evaluation (PI-HTE) which harnesses hypertree decomposition algorithms to efficiently estimate the estimand, exploiting the sparse nature of the empirical probability tables. We then evaluate our algorithm's performance over several classes of benchmarks for causal effect queries. Our results confirm the significance of the hypertree width at capturing the algorithm's performance, and its superiority over treewidth in the context of causal effect evaluation.

In particular, our algorithm and bounds help characterize the computational feasibility of using the empirical plug-in scheme, enabling it to be applied to estimands previously thought computationally infeasible, and establishing it as a simple and practical baseline for causal effect estimation. Moreover, since a causal query may have many candidate estimands, tree-width and hyperwidth-based bounds can also be used as one metric to select among different estimands.

6 ACKNOWLEDGMENTS

This work was supported in part by NSF grant CNS-2321786.

References

Rina Dechter. *Constraint Processing.* Morgan Kaufmann, 2003.

Rina Dechter. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013.

Rina Dechter and Robert Mateescu. AND/OR search spaces for graphical models. *Artif. Intell.*, 171(2-3): 73–106, 2007.

Rina Dechter, Lars Otten, and Radu Marinescu. On the practical significance of hypertree vs. treewidth. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *ECAI 2008, Proceedings*, volume 178, pages 913–914, 2008.

Vibhav Gogate and Rina Dechter. A complete anytime algorithm for treewidth. *CoRR*, abs/1207.4109, 2012. URL `http://arxiv.org/abs/1207.4109`.

Georg Gottlob, Nicola Leone, and Francesco Scarcello. A comparison of structural CSP decomposition methods. *Artif. Intell.*, 124(2):243–282, 2000.

Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Treewidth and hypertree width. In Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli, editors, *Tractability: Practical Approaches to Hard Problems*, pages 3–38. Cambridge University Press, 2014.

Yonghan Jung, Jin Tian, and Elias Bareinboim. Estimating causal effects using weighting-based estimators. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 10186–10193, 2020a.

Yonghan Jung, Jin Tian, and Elias Bareinboim. Learning causal effects via weighted empirical risk minimization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020b.

Kalev Kask, Rina Dechter, Javier Larrosa, and Avi Dechter. Unifying tree decompositions for reasoning in graphical models. *Artif. Intell.*, 166(1-2):165–193, 2005.

Kalev Kask, Andrew Gelfand, Lars Otten, and Rina Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*, pages 54–60. AAAI Press, 2011.

Lars Otten and Rina Dechter. Bounding search space size via (hyper)tree decompositions. In David A. McAllester and Petri Myllymäki, editors, *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, 2008*, pages 452–459, 2008.

Judea Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2nd edition, 2009.

Anna Raichev, Alexander Ihler, Jin Tian, and Rina Dechter. Estimating causal effects from learned causal networks. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI 2024).*

Ilya Shpitser and Judea Pearl. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, page 1219, 2006.

Jin Tian. *Studies in Causal reasoning and Learning.* PhD thesis, University of California, Los Angeles, 2002.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model.
   yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm.
   Yes in section 2 and section 3
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results.
   yes, we explain the setting for causal effect estimation in section 2
   (b) Complete proofs of all theoretical results.
   yes
   (c) Clear explanations of any assumptions.
   yes
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL).
   yes, included in appendix
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen).
   not applicable
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times).
   not applicable
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider).
   no, we are showing worst case results, and interested in only how the results are in $O(t^{hw})$
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets.
   not applicable
   (b) The license information of the assets, if applicable.
   not applicable
   (c) New assets either in the supplemental material or as a URL, if applicable.
   not applicable
   (d) Information about consent from data providers/curators.
   not applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content.
   not applicable
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots.
   not applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable.
   not applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation.
   not applicable

A Appendix

Here we will give details on some of the hypertree decompositions and queries that were not detailed in the main part of the paper. In table 1 we presented results on the Diamond Model in Figure 2c. The estimand expression is displayed below. Despite, the diamond model containing only 17 observable variables, it has a very complicated estimand also including ratios.

Example 5 *Estimand Expression for the diamond model:*
$P(V_{16}|do(V_0,V_4)) = \sum_{V_5,V_9,V_{13},V_6,V_{10},V_{14},V_7,V_{11},V_{15},V_8,V_{12}} \frac{num}{denom}$

$$num = \sum_{V_0,V_2,V_6,V_{10}} P(V_{16}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4,V_8,V_{12})\times$$
$$P(V_{12}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4,V_8)\times$$
$$P(V_8|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4)\times$$
$$P(V_4|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15})P(V_{14}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10})\times$$
$$P(V_{10}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6)P(V_6|V_0,V_1,V_5,V_9,V_{13},V_2)P(V_2|V_0,V_1,V_5,V_9,V_{13})P(V_0)\times$$
$$\Bigg(\sum_{V_0,V_2,V_6,V_{10},V_4,V_8,V_{12},V_{16}} P(V_{16}|V_0,V_1,V_5,V_9,V_{13},V_2,V_{6,V10},V_14,V_3,V_7,V_{11},V_{15},V_4,V_8,V_{12})\times$$
$$P(V_{12}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4,V_8)\times$$
$$P(V_8|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4)\times$$
$$P(V_4|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15})\times$$
$$P(V_{14}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10})P(V_{10}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6)\times$$
$$P(V_6|V_0,V_1,V_5,V_9,V_{13},V_2)P(V_2|V_0,V_1,V_5,V_9,V_{13})P(V_0)\Bigg)\left(\sum_{V_0} P(V_{12}|V_0,V_9,V_{10},V_{11})P(V_{10}|V_0,V_9)P(V_0)\right)\times$$
$$\left(\sum_{V_0} P(V_8|V_0,V_5,V_6,V_7)P(V_6|V_0,V_5)P(V_0)\right)P(V_{15}|V_0,V_{13},V_{14})P(V_{11}|V_0,V_9,V_{10})P(V_7|V_0,V_5,V_6)\times$$
$$P(V_{13}|V_0)P(V_9|V_0)P(V_5|V_0)$$

$$denom = \sum_{V_0,V_2,V_6,V_{10},V_{16}} P(V_{16}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4,V_8,V_{12})\times$$
$$P(V_{12}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4,V_8)\times$$
$$P(V_8|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15},V_4)\times$$
$$P(V_4|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10},V_{14},V_3,V_7,V_{11},V_{15})\times$$
$$P(V_{14}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6,V_{10})P(V_{10}|V_0,V_1,V_5,V_9,V_{13},V_2,V_6)\times$$
$$P(V_6|V_0,V_1,V_5,V_9,V_{13},V_2)P(V_2|V_0,V_1,V_5,V_9,V_{13})P(V_0)$$

In table 2 we presented results on the 2-layer model shown in Figure 2d. Here two different estimands were generated via CTE on the hyper tree decomposition:

Example 6 *Estimand Expression for the 2-layer model corresponding to hypertree decoposition with* $hw = 2$ *in Figure 6a:*
$P(Z|do(T)) = \sum_{V_0,V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,V_{10},V_{11},X} P(Z|V_0,V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,V_{10},V_{11})$
$P(V_0|V_1,V_2,V_3,X)P(V_3|V_4,V_5,V_6,X)P(V_6|V_7,V_8,V_9,X)P(V_9|V_{10},V_{11},X)P(X|T)P(T)$

Example 7 *Estimand Expression for the 2-layer model corresponding to hypertree decoposition with $hw = 5$ in Figure 6b:*
$P(Z|do(T)) = \sum_{V_0,V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,V_{10},V_{11}} P(Z|V_0,V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,V_{10},V_{11})P(T)$
$(\sum_X P(V_0|V_1,V_2,V_3,X)P(V_3|V_4,V_5,V_6,X)P(V_6|V_7,V_8,V_9,X)P(V_9|V_{10},V_{11},X)P(X|T))$

In table 1 we presented results on the 3-layer model shown in Figure 2e. Here O_1 is the output function over $V_0..V_{11}$, this is included here because the query variables are not part of a single function together. Here we show the hypertree decomposition and the estimand:

Example 8 *Estimand Expression for the 3-layer model:*
$P(V_0..V_{11}|do(Z,T)) = \sum_{X_0,X_1,X_2} P(X_0|T)P(X_1|T)P(X_2|T)P(V_0|V_1,V_2,V_3,X_0)$
$P(V_3|V_4,V_5,V_6,X_0)P(V_6|V_7,V_8,V_9,X_0)P(V_9|V_{10},V_{11},X_0)P(V_1|X_1)P(V_4|X_1)P(V_7|X_1)P(V_{10}|X_1)$
$P(V_2|X2_)P(V_5|X_2)P(V_8|X_2)P(V_{11}|X_2)P(T)$

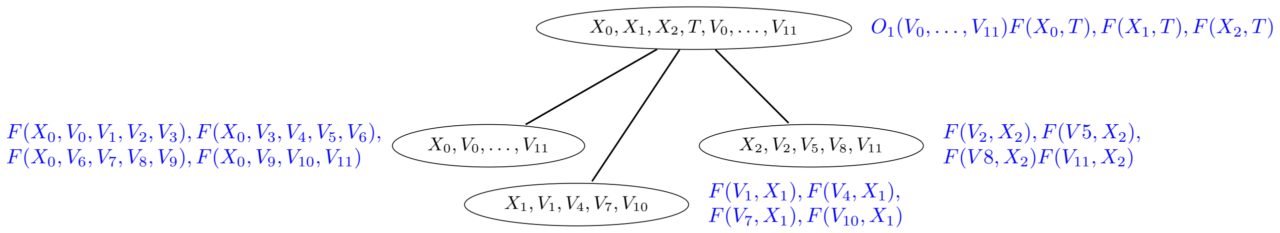

$X_0, X_1, X_2, T, V_0, \ldots, V_{11}$
$O_1(V_0,\ldots,V_{11})F(X_0,T), F(X_1,T), F(X_2,T)$
$F(X_0,V_0,V_1,V_2,V_3), F(X_0,V_3,V_4,V_5,V_6),$
$F(X_0,V_6,V_7,V_8,V_9), F(X_0,V_9,V_{10},V_{11})$
$X_0, V_0, \ldots, V_{11}$
$X_2, V_2, V_5, V_8, V_{11}$
$F(V_2,X_2), F(V5,X_2),$
$F(V8,X_2)F(V_{11},X_2)$
$X_1, V_1, V_4, V_7, V_{10}$
$F(V_1,X_1), F(V_4,X_1),$
$F(V_7,X_1), F(V_{10},X_1)$

Figure 7: 3-Layer model's hypertree with $hw = 4$

A.1 Code

The code can be found at: https://github.com/anniemeow/PlugIn