Causal Inference from an EM-Learned Causal Model

Anna K. Raichev[1], **Jin Tian**[2], **Rina Dechter**[1]
[1]University of California, Irvine
[2] Iowa State University
araichev@uci.com, jtian@iastate.edu, dechter@ics.uci.edu

Abstract

The standard approach to answering a causal query (e.g., P(Y|do(X)) when given a causal diagram and observational data is to first generate an estimand, which is an expression over the observable variables, if the query is identifiable. The estimand is then evaluated from the observational data. In this paper, we propose an alternative paradigm for answering causal queries over discrete observable variables. We suggest learning the full causal Bayesian network containing latent variables from the observational data. Once a full model is available, Probabilistic Graphical Model (PGM) algorithms developed over the past three decades can be applied to answer the causal query. We present this idea and provide analysis, demonstrating that this approach can be far more effective than the estimand-based approach with plug-in estimation, when the diagram has a low treewidth. Our analysis and experiments illustrate the potential of this approach over a collection of synthetically generated causal models.

Introduction

Structural Causal models (SCMs) are a formal framework for reasoning about causal knowledge in the presence of uncertainty [20]. When the full SCM is available, it is possible to use standard probabilistic inference [6, 8] to directly answer causal queries that evaluate how forcing some variable's assignment $X = x$ affects another variable Y, written as $P(Y|do(X = x))$. However, in practice, the full causal model is rarely available and only a causal diagram (a directed graph that captures the causal relationships of the underlying SCM) is assumed. Causal diagrams may include both *observable variables*, which can be measured from data, and *latent variables*, which are unobservable and for which data cannot be collected. Previous work has shown that if, in addition to the diagram, we have access to data sampled from the observational distribution, we can still answer many causal queries or determine whether they cannot be uniquely answered [23, 24, 21, 10].

The main approach developed in the past two decades for answering causal queries under such assumptions is a two-step process which we call *estimand-based causal inference*. The first step is to determine if the causal query is *identifiable* - ie. uniquely answerable from the model's observable distribution, and if so, construct an expression, the estimand, that captures the answer symbolically using quantities defined over the observed distribution only. The second step is to estimate the estimand's value from data sampled from this distribution. Over the past few years a variety of estimand-based strategies were developed using primarily statistical estimation methods [11, 12, 13, 14, 2]. Surprisingly, even though the graph is so central to causal models, probabilistic graphical models (PGM) algorithms play no role in most current causal inference approaches.

In this paper we will explore a new scheme which will open a window for harnessing all the algorithmic power of PGMs developed in the last four decades [6, 8, 20, 15] towards answering causal queries, focusing on discrete Bayesian networks only. We will show that under certain conditions this approach can be far more effective than the estimand-based methodology. The idea is simple: we can first learn a full causal Bayesian network over both observed and latent variables, and then we can apply standard PGM schemes to answer causal queries.

A highly used learning scheme that can learn a full causal Bayesian network containing latent variables is Expectation Maximization (EM) [16] which learns a Bayesian network's parameters given the network's underlying directed graph, when the data has missing values (e.g., when there are latent variables). Although not exact, EM converges to a local optimum of the likelihood of the data and its performance has been shown to be good for many applications [16, 15].

The computational benefit of our approach is tied to the complexity of the given causal graph. If the *treewidth* of the causal graph is low, then this approach promises to be highly effective, significantly more so than the estimand-based schemes. This is because 1) The learned model promises to be more accurate when the treewidth is low since it includes an inference step that can be evaluated exactly, 2) the learning step can be done once, and its time complexity amortized against answering any causal query that is identifiable, each being answered efficiently, and 3) the approach is superior to estimand-based method whenever the estimand includes summations over a large number of variables.

Example. To illustrate when the estimand approach can prove costly on a low treewidth model, consider the model in Figure 1a. To calculate the query $P(V_6 \mid do(V_0))$ using the estimand-based approach, we run the standard *ID* identifiability algorithm [23, 21] and generate the following estimand expression:

$$P(V_6 \mid do(V_0)) = \sum_{V_1,V_2,V_3,V_4,V_5} P(V_5 \mid V_0, V_1, V_2, V_3, V_4)$$
$$\times P(V_3 \mid V_0, V_1, V_2) P(V_1 \mid V_0) \sum_{v_0} P(V_6 \mid v_0, V_1, V_2, V_3, V_4, V_5)$$
$$\times P(V_4 \mid v_0, V_1, V_2, V_3) P(V_2 \mid v_0, V_2) P(v_0). \quad (1)$$

Estimating this expression takes time exponential in the number of observed variables [6]. On the other hand, if we learn the full Causal Bayesian network shown in Figure 1b, then the query $P(V_6 \mid do(V_0))$ becomes a standard probabilistic query $P(V_6 \mid V_0)$ in the truncated Bayesian network shown in Figure 1c. We can use exact PGM algorithms, such as bucket elimination or join-tree inference techniques, to answer any such query efficiently, in time exponential in the treewidth which is 2 in this graph.

One main challenge facing our proposed scheme is that the domains and distributions of the latent variables are unknown - and could be continuous or discrete. Fortunately, recent work [32] shows that any SCM over discrete observable variables is equivalent for answering causal queries to an SCM where all latent variables are discrete with finite domains, and an upper bound on the latent domain sizes is provided. In this paper, we study models over discrete observable variables and therefore can assume all the latent variables are discrete. However, the upper bounds on the latent domain sizes given in [32, Proposition 2.7] are conservative and often very large. We instead empirically test with increasing domain sizes. Once the latent domain sizes are fixed, we use the EM algorithm to learn the parameters of the full causal Bayesian network for answering causal queries.

Contributions. In this work, we will present a new general scheme for answering causal queries over discrete causal Bayesian networks and provide theoretical and empirical analysis demonstrating its potential. Specifically, we provide:

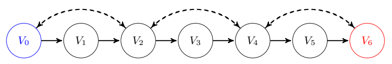
1. A general scheme for computing causal queries that invites the power of well-known algorithms for discrete graphical model's learning and inference, and focus on the special case that utilizes EM for learning.
2. An analysis of the scheme's theoretical properties, comparing to baseline standard approaches, highlighting its challenges and benefits.
3. An empirical evaluation on a set of synthetically generated benchmarks showing that in many cases, especially on graphs with low treewidth, the scheme can be superior to standard and state-of-the-art algorithms.

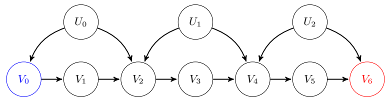
Related work

Earlier work that addresses the computational aspect of causal inference falls into two main categories. The first is highly related to our proposed approach of learning the full causal model by EM. In recent work by Darwiche et. al [7, 5], they first convert the causal diagram into a circuit, exploiting the functional form of SCM mechanisms, to yield compact circuits whenever possible. Subsequently, they learn the circuit parameters by EM and then answer the causal queries. Their work is restricted to only bi-valued variables and no empirical evaluation was provided, as far as we know. Another line of work explores neural network approaches for counterfactual inferences [27, 28]. Other related work that uses the EM scheme is [22, 31]. In the later paper (on Causal EM), they assume knowledge of the functional mechanisms in the model, and only unknown are distributions on the latent variables (whereas we assume knowledge of the graph only).

The second category is work with estimand-based schemes, focusing on various estimation techniques of the estimands. Recent work appears in Jung, Tian, and Bareinboim [11, 12, 13, 14]. Another line of work is a PAC learning analysis of causal inference [3].

The rest of the paper is organized as follows. Section 2 provides background information and definitions, Section 3 outlines the main idea of our approach, Section 4 provides empirical evaluation, and Section 5 concludes.


(a) Causal diagram of chain model with 7 observable and 3 latent variables. Dashed bidirected edges represent latent variables.


(b) The causal diagram in (a) with explicit latent variables.

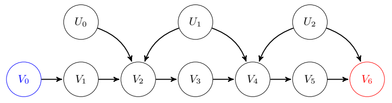

(c) The truncated causal diagram after intervention $do(V_0)$.

Figure 1

Background

Notation. Capital letters (X) represent variables and small letters (x) represent their values. Boldfaced capital letters ($\mathbf{X}$) denote a collection of variables, $|\mathbf{X}|$ its cardinality, $\mathcal{D}(\mathbf{X})$ their joint domains, and $\mathbf{x}$ a particular realization in that joint domain. $P(\mathbf{X})$ stands for the joint distribution over $\mathbf{X}$ and $P(\mathbf{x})$ represent probabilities $P(\mathbf{X} = \mathbf{x})$ of a configuration $\mathbf{x}$; similarly, notation $P(\mathbf{Y} \mid \mathbf{X})$ represents a set of conditional distributions $P(\mathbf{Y} \mid \mathbf{X} = \mathbf{x}), \forall \mathbf{x}$.

Definition. [Structural Causal Model (SCM)] [20] A structural causal model (SCM) is a 4-tuple $\mathcal{M} = \langle \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{F}, P(\boldsymbol{U}) \rangle$ where: (1) $\boldsymbol{U} = \{U_1, U_2, ..., U_k\}$ is a set of exogenous unmeasurable latent variables; (2) $\boldsymbol{V} = \{V_1, V_2, ..., V_n\}$ is a set of observable variables; (3) $\boldsymbol{F} = \{f_i : V_i \in \boldsymbol{V}\}$ is a set of functional mechanisms f_i that each determine the value v_i of their corresponding V_i as a function of V_i's causal parents $PA_i \subseteq \boldsymbol{U} \cup \boldsymbol{V} \setminus V_i$ so that $f_i : \mathcal{D}(PA_i) \rightarrow \mathcal{D}(V_i)$ and $v_i \leftarrow f_i(pa_i)$ and (4) $P(\boldsymbol{U})$ is a probability distribution over the exogenous variables. The exogenous variables U are assumed to be mutually independent, i.e.,$P(\boldsymbol{U}) = \prod_{U \in \boldsymbol{U}} P(U)$.

Causal diagrams. An SCM $\mathcal{M}$ can be associated with a directed graph $\mathcal{G} = \langle \boldsymbol{V} \cup \boldsymbol{U}, E \rangle$ called a causal diagram (also a causal graph). Each node in the graph uniquely maps to a variable in the SCM. Simplifying notation, we use the same capital letters to refer both to the variables in the SCM and its corresponding graph node in the causal diagram. There is an arc from node $X \in (U \cup V)$ to node $V_i \in V$ iff $X \in PA_i$ in the graph. In other words, X is a causal parent of V_i and thus involved in V_i's mechanism of assignment, f_i.

An SCM whose latent variables directly influence at most a single observable variable is referred to as *Markovian*, and one whose latent variables directly influence at most two observable variables is called *Semi-Markovian*. It was shown that any SCM can be transformed into an equivalent Semi-Markovian one such that answers to causal queries are preserved [23]. In Markovian and semi-Markovian models it is common to use an *Acyclic Directed Mixed Graph (ADMG)* for the causal diagram where all nodes corresponding to latent variables having a single child are omitted, and nodes representing latent variables having two children are replaced with a bidirectional dashed arc between the two children (latent variables not explicitly shown, see Figure 1a).

Causal Bayesian Network (CBN) An SCM $\mathcal{M}$ induces a CBN $\mathcal{B} = \langle \mathcal{G}, \mathcal{P} \rangle$ specified by the causal diagram $\mathcal{G} = \langle \boldsymbol{V} \cup \boldsymbol{U}, E \rangle$ along with its associated conditional probability distributions $\mathcal{P} = \{P(V_i|PA_i), P(U)\}$. The distribution $P(\boldsymbol{V}, \boldsymbol{U})$ factorizes according to the causal diagram by

$$P(\boldsymbol{V}, \boldsymbol{U}) = \prod_{V_j} P(V_j|PA_j) \cdot \prod_i P(U_i). \tag{2}$$

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

$$P(\boldsymbol{V}) = \sum_{\boldsymbol{u}} \prod_{V_j} P(V_j|PA_j) \cdot P(\boldsymbol{u}) \tag{3}$$

Causal effect and the truncation formula. We use $P(Y|do(X))$ to denote the distributions resulting from an intervention which fixes the value of X, and is called the causal effect of $do(X)$ on Y. This is in contrast to $P(Y|X)$ which means conditioning on an observation. It is obtained from the original model by removing the mechanism of X and setting the value $X = x$. It asserts

$$P(\boldsymbol{V} \backslash \boldsymbol{X}, \boldsymbol{U} \mid do(\boldsymbol{X} = \mathbf{x})) = \prod_{V_j \notin \mathbf{X}} P(V_j|PA_j) \cdot P(\boldsymbol{U}) \Big|_{\boldsymbol{X}=\mathbf{x}} \tag{4}$$

Causal inference task. We study the causal inference task of answering a causal effect query $P(\mathbf{Y} \mid do(\mathbf{X}))$ given a causal diagram $\mathcal{G} = \langle \boldsymbol{V} \cup \boldsymbol{U}, E \rangle$ and data samples from the observational distribution $P(\boldsymbol{V})$. Given a causal diagram $\mathcal{G}$, the causal effect query $P(\mathbf{Y} \mid do(\mathbf{X}))$ is **identifiable** if any two SCMs that agree on the observational distribution $P(\boldsymbol{V})$ also agree on $P(\mathbf{Y} \mid do(\mathbf{X}))$. Otherwise, the answer to $P(\mathbf{Y} \mid do(\boldsymbol{X}))$ is not unique with respect to the observational distribution $P(\boldsymbol{V})$ and thus cannot be determined.

The estimand-based approach for causal inference. As noted, the standard methodology for answering causal queries developed and explored in the last two decades has two primary steps. The first is the *identifiability step*. Namely given a causal diagram and a query $P(\mathbf{Y} \mid do(\mathbf{X}))$, determine if the query is identifiable. If it is, generate an estimand which is an algebraic expression of probabilistic quantities over observed variables only that expresses the answer to the query. A complete polynomial algorithm called ID has been developed for this task [23, 21]. The second step is *estimation*. Namely using samples from the observational distribution, evaluate the value of the estimand.

Expectation Maximization (EM) The EM algorithm is a well-studied learning scheme that learns Bayesian networks when the structure of the model is known but the data has missing values [9, 6, 15]. The EM Algorithm is a maximum likelihood approach that works iteratively through an Expectation Step (E-Step) and Maximization Step (M-Step). For more details see [16]. The E-step is the most computationally demanding requiring inference. When the graph has bounded treewidth, the E-step can be accomplished by exact inference such as the join tree scheme[16, 8]. Otherwise approximate algorithms like belief propagation can be used [20, 17, 30].

The treewidth also known as the induced-width, is a graph parameter that determines how close the graph is to a tree. While it is hard to compute good bounds can be obtained[8]. Exact inference in discrete PGMs is known to be time and space exponential in the treewidth.

Learning-Based Causal Inference

We propose to answer a query $P(\mathbf{Y} \mid do(\mathbf{X}))$ by first learning (the probability parameters $\mathcal{P}$ of) the full CBN $\mathcal{B} = \langle \mathcal{G}, \mathcal{P} \rangle$ given a causal diagram $\mathcal{G} = \langle \boldsymbol{V} \cup \boldsymbol{U}, E \rangle$ and data samples from the observational distribution $P(\boldsymbol{V})$, termed *learning-based* approach. One immediate challenge to this approach is unknown domains and distributions of latent $\boldsymbol{U}$ variables. As noted earlier, we study models over discrete $\boldsymbol{V}$ variables and can safely assume that all $\boldsymbol{U}$ variables take discrete finite domains based on results in [32]. We investigate the impact of latent domain sizes with the hope to empirically determine suitable values.

Another challenge to the learning-based approach is that there could be many parameters $\mathcal{P}$ producing the same observational distribution $P(\boldsymbol{V})$. Would that be a problem for this approach? Let $\mathcal{B} = \langle \mathcal{G}, \mathcal{P} \rangle$ be a CBN inducing an observational distribution $P(\boldsymbol{V})$ by Eq. (3). We say $\mathcal{B}$ is *consistent* with $P(\boldsymbol{V})$. Based on the definition, if the query $P(\mathbf{Y} \mid do(\mathbf{X}))$ is identifiable from $\mathcal{G}$ and $P(\boldsymbol{V})$, then all full CBN $\mathcal{B}$ consistent with $P(\boldsymbol{V})$ will give the same unique answer to the query.

Once a full CBN $\mathcal{B} = \langle \mathcal{G}, \mathcal{P} \rangle$ is learned, any causal query $P(\mathbf{Y} \mid do(\mathbf{X} = \mathbf{x}))$ can be answered based on Eq. (4). Let $\mathcal{G}_{\overline{\boldsymbol{X}}}$ denote the graph obtained by deleting all arrows incoming to $\boldsymbol{X}$. Let $\mathcal{P}_{\mathbf{X}=\mathbf{x}} = \{P(V_i|PA_i), P(U)\}_{V_i \notin \boldsymbol{X}} \cup \{P(\mathbf{X} = \mathbf{x}) = 1\}$. We call $\mathcal{B}_{\mathbf{X}=\mathbf{x}} = \langle \mathcal{G}_{\overline{\boldsymbol{X}}}, \mathcal{P}_{\mathbf{X}=\mathbf{x}} \rangle$ a truncated CBN. Formally, $P(\mathbf{Y} \mid do(\mathbf{X} = \mathbf{x}))$ in $\mathcal{B}$ is given by $P(\mathbf{Y} \mid \mathbf{X} = \mathbf{x})$ in $\mathcal{B}_{\mathbf{X}=\mathbf{x}}$ based on Eq. (4). Next we present our proposed algorithm.

Algorithm EM for Causal Inference (EM4CI) is presented in Algorithm 1. Given a causal diagram $\mathcal{G}$ and a causal query $Q = P(\mathbf{Y} \mid do(\mathbf{X} = \mathbf{x}))$, *Step 1* checks if the query is

Algorithm 1: EM4CI(k_{hyp})

```
input  : A causal graph $\mathcal{G} = \langle \boldsymbol{U} \cup \boldsymbol{V}, E \rangle$ over latent
         $\boldsymbol{U}$ variables and discrete observables $\boldsymbol{V}$;
         $Samples$ from $P(\boldsymbol{V})$;
         Query $Q = P(\mathbf{Y} \mid do(\mathbf{X} = \mathbf{x}))$;
         Hypothesized latent domain size $k_{hyp}$.
output : Estimated $P(\mathbf{Y} \mid do(\mathbf{X} = \mathbf{x}))$

Step 1: If $\neg$ identifiable($\mathcal{G}, Q$), terminate.
Step 2: CBN $\mathcal{B} = \langle \mathcal{G}, \mathcal{P} \rangle \leftarrow$ EM($\mathcal{G}, |\mathcal{D}_U| = k_{hyp}, Samples\rangle$
Step 3: Construct truncated CBN $\mathcal{B}_{\mathbf{X}=\mathbf{x}}$.
Step 4: return $\leftarrow$ PGM inference($\mathcal{B}_{\mathbf{X}=\mathbf{x}}, P(\mathbf{Y} \mid \mathbf{X} = \mathbf{x})$)
```

identifiable. This can be done via the well-known ID algorithm [23, 21]. If the query is not identifiable, then it is not possible to answer from observational data and we terminate. Otherwise in *Step 2*, the given set of *Samples* from the observational distribution $P(V)$ are used to learn a full CBN model $\mathcal{B}$. This can be accomplished by any learning scheme that assumes a known graph and data with latent variables, such as the EM algorithm [6, 15]. In this basic version of the algorithm, the latent domain sizes k_{hyp} are assumed to be given. *Step 3* generates the truncated CBN $\mathcal{B}_{\mathbf{X}=\mathbf{x}}$, and *Step 4* employs a standard *PGM* inference algorithm, e.g. jointree [6, 8], over the CBN $\mathcal{B}_{\mathbf{X}=\mathbf{x}}$ to compute the conditional marginal $P(\mathbf{Y} \mid \mathbf{X} = \mathbf{x})$, yielding the answer.

Latent domain sizes will affect the EM algorithm's performance and the model learned. Theoretical upper bounds on the domain sizes are a function of the causal diagram's structure and the domain sizes of the observable variables and are often very high [32]. If we assume large domain sizes, learning would be harder, thus begging the question: "How large is large enough?". We will illustrate empirically the impact of assumed domain sizes on model accuracy and learning time. Yet a simple work-around we will explore is to run our scheme iteratively for increasing values of hypothesized latent domain sizes, and terminate once the log-likelihood of the learned model stops increasing.

Complexity. The complexity of our approach can be analyzed along its main 3 steps. *Step 1* of determining identifiability is polynomial in the graph size [23]. *Step 2* of EM depends on the sample size and number of iterations needed for convergence which is hard to predict. The most extensive computation in each iteration is the Expectation step that requires probabilistic inference which can be accomplished in time and memory exponential in the treewidth of the graph. Otherwise approximation algorithms such as belief propagation are commonly used. Thus, each iteration takes $O(m \cdot nk^w)$, where m is the number of samples, k bounds the domain size, and w is the treewidth. If we have T iterations we get that the complexity of EM is $O(T \cdot m \cdot n \cdot k^w)$. *Step 4* of probabilistic inference is $O(n \cdot k^w)$ if performing exact inference. In summary, the complexity of algorithm *EM4CI* is $O(T \cdot m \cdot n \cdot k^w)$.

Table 1: Estimand Expressions for Models 1-8 in Table 2.

Model	Estimate of $P(Y \mid do(X))$
1	$\frac{\sum_W P(X,Y \mid R,W)P(W)}{\sum_W P(X \mid R,W)P(W)}$
2	$\sum_R P(R \mid X_1) \sum_{x_1,Z} P(Y \mid R, x_1, X_2, Z)P(Z \mid R, x_1)P(x_1)$
3	$P(Y)$
4	$\sum_{R,S} P(S \mid X_1, X_2, X_3, Z)P(R \mid X_1) \sum_{x_1,x_3,Z} P(Y \mid R, x_1, X_2, x_3, Z)P(x_3 \mid x_1, X_2, Z)P(x_1, Z)$
5	$\sum_{Z_1,Z_2,Z_3} P(Z_3 \mid Z_2)P(Z_1 \mid X, Z_2) \frac{\sum_x P(Y,Z_3 \mid x,Z_1,Z_2)P(x,Z_2)}{\sum_x P(Z_3 \mid x,Z_1,Z_2)P(x,Z_2)} P(Z_2)$
6	$\sum_{Z_2} P(Y \mid X_1, X_2, Z_2)P(Z_2)$
7	$\sum_{Z_2,Z_3} P(Y \mid X_1, X_2, Z_2, Z_3)P(Z_2, Z_3)$
8	$\sum_{R,W,Z} P(Z \mid R, W, X)P(R \mid W) \sum_x P(Y \mid R, W, x, Z)P(x \mid R, W)P(W)$

Estimand-based vs. learning-based. In the estimand-based scheme, once the estimand is generated[1] we can determine its "complexity to estimate" by inspection using two parameters. The first is the largest scope size of any function appearing in the algebraic expression of the estimand, and the second is the number of summation variables. The first indicates the number of samples needed to estimate the function and the second yields computation which is exponential in the number of summed variables. Consequently, in our experiments we will compare to the estimand-based methods by providing an estimate when possible, or by assessing that the estimand cannot be estimated in a reasonable time based on the two complexity indicators that can be determined by inspection of the estimand expression. In the example in Figure 1, the estimand for $P(V_6 \mid do(V_0))$ obtained by ID shown in Equation 1 has 5 summation variables and has a function on all variables in the model. This pattern will persist as n increases. Evaluating this expression in a brute-force way (e.g., by the plugin method) is not realistic, requiring many samples and too much time (exponential in n). In contrast, since these graphs have treewidth 2, they would be easy to learn and evaluate as we will illustrate in our empirical evaluation. Clearly, the opposite situation can occur when the full CBN has a high treewidth while the estimand expression is small and simple. Even in the chain example, if the query is $P(V_6 \mid do(V_5))$ its estimand is $P(V_6|do(V_5)) = \sum_{v_4} P(V_6|v_4, V_5)P(v_4)$ (by the backdoor criterion [20]) and it can be easily estimated from empirical distributions, without the overhead of learning the full model. Thus, the idea is to apply the respective schemes to instances that play to their strength.

Another benefit for our learning-based approach is that the learning process, done once, can be amortized over multiple queries. Alternatively, the estimand-based approach requires generating and estimating a separate estimand for each query.

Empirical Evaluation

Experimental setup

The first goal of our empirical evaluation is a proof of concept to show that our learning approach is viable for causal inference from accuracy and computational aspects. Within this we explore different choices of hypothesized domain sizes for the latent variables. To the extent possible we compare against 2 estimand-based approaches: the brute-force plug-in approach and a current state of the art, called (WERM) [12].

[1]We assume the standard ID algorithm with estimand simplification add-ons [25, 26] is used to generate the estimand. We note that, in principle, one can generate different estimands using say do-calculus. However, no systematic algorithm is available to generate and look for estimands that are 'better' for estimation purpose.

All experiments were run on a 2.66 GHz processor with 8 GB of memory.

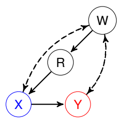
Note that the use of EM for learning graphical models from data with latent variables is quite abundant [15, 4, 18]. Yet our novel contribution is in proposing this as a methodology for causal inference (which was only sporadically explored) and in illustrating that it is a viable approach in many cases, as we show in this section.

Benchmarks.

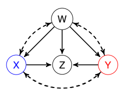
The input to a causal inference algorithm is 1) a causal diagram (of a full underlying SCM), 2) data from the model's observational distribution, and 3) a query $Q = P(\mathbf{Y} \mid do(\mathbf{X}))$. We generated each such triplet benchmark instance by first choosing a causal diagram and a query. Then, subject to assumptions on variables' observed and latent domain sizes, we generated the conditional probability tables (CPTs), one per variable and its parents, yielding a full CBN. Given such full models we generated samples from the observational distribution by forward sampling [6] and then, from each generated sample, we remove the latent variable's values. We also computed the exact answer to the given query over the full CBN by an exact algorithm (e.g., the join-tree scheme [8]), so we can compare against it in our evaluation.

Selected causal graphs. The causal diagrams for our experiments were selected in two ways. First, we used a set of 9 small diagrams from the literature [20, 29, 12], 3 of which are shown in Figure 2 (See the Supplemental for the rest). Second, we propose three classes of graphs that can have any number of variables n, but whose treewidth is bounded. This is done in order to highlight the strength of our learning-based scheme when the treewidth is bounded and small. In particular, we have chain networks (treewidth is 2 as in Figure 1), diamond networks (treewidth 5 as in Figure 3), and the cone-cloud networks (treewidth $O(\sqrt{n})$ as in Figure 4).

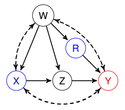
Domain sizes and CPTs. Since, as mentioned before, domain sizes of the latent variables can play an important role in our approach, we generated a number of different models for each graph by varying the domain sizes of all the variables. Each CPTs' parameters were generated by sampling the values for each row from a Dirichlet distribution [6], ensuring they sum to 1. The Dirichlet distribution is parameterized by a vector of real values $\alpha = [\alpha_1, \ldots, \alpha_n]$. We chose $\alpha_i \in [1, 16]$ uniformly at random. For the small diagrams in Figure 2 we chose domains of size $d = 2$ (observed variables) and $k = 10$ (latent variables). For the chain, diamond, and cone-cloud diagrams we created a set of problem instances



(a) Model 1



(b) Model 3



(c) Model 8

Figure 2: A subset of causal diagrams for models used in our experiments. Blue variables are intervened on and red variables are the outcome variables corresponding to the query $P(Y \mid do(X))$.

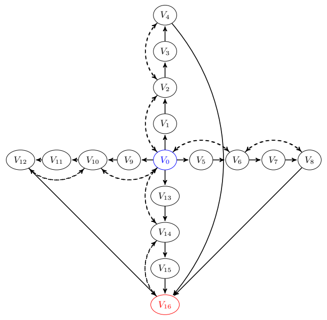


Figure 3: Diamond Model: $n = 17$

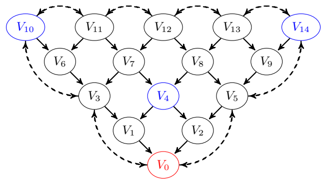


Figure 4: Cone Cloud Model: $n = 15$

having $(d, k) = (2, 8), (2, 16), (4, 4), (4, 8)$. ($(d, k) = (2, 2)$ and $(8, 16)$ are shown in the Supplemental.)

Algorithms and performance measures.

Algorithm EM4CI. We evaluated our EM4CI algorithm (Algorithm 1) and compared against the estimand-based approaches. For learning we used the EM algorithm from the SMILE tool: Structural Modeling, Inference, and Learning Engine package [1], written in C++. The EM algorithm learns the network's parameters given the graph and data on the observed variables only. In SMILE the inference in the E step [16], which is the costly operation, is done by the join-tree algorithm [19, 8, 16] which is also applied to answer $P(\mathbf{Y} \mid \mathbf{X})$ over the learned model.

We evaluated the algorithm in 2 modes. The *basic mode* where we run it for different assumed latent domain sizes (k_{hyp}), and the *iterated mode* where we iterated EM4CI for an increasing list of latent domain sizes, terminating when the Log-Likelihood (LL) of the model stops increasing compared with the previous iteration. This mode emerged as we were experimenting, realizing that it would be hard to predict the best latent domain sizes to choose and estimating many values of k_{hyp} will lead to longer runs.

Esimand-based algorithms: plug-in and WERM. The estimand-based method first generates an estimand. In the brute-force plug-in method, estimates of the estimand are generated from the observational data directly by computing the empirical conditional probabilities. More advanced estimand-based algorithms were explored in recent works focusing on statistical estimation techniques [12, 11, 13, 14]. We will compare against the state-of-the-art scheme called WERM [12] on some models.

Performance measures. We ran our algorithm multiple times on each problem instance and computed the *mean absolute deviation (mad)* between the true answer (computed from the full model) and the answer produced by the estimating algorithm. The measure mad for a query $P(\mathbf{Y} \mid do(\mathbf{X} = \mathbf{x})$ is computed by averaging the mad over all single-value queries $P(\mathbf{Y} = \mathbf{y} \mid do(\mathbf{X} = \mathbf{x}))$ for $y \in \{0, \ldots, d-1\}$ where d is the observed domain size. (See *mean relative deviation (mrd) in the Supplemental.*) We also show the *average log likelihood* of the learned model as computed by EM and report average time per domain size (basic mode) or of the total time of the iterated version.

Results

Results on small models. Results on Models 1-8 (Figure 2) are presented in Table 2 We ran EM4CI on each instance query $P(Y = y \mid X = x)$ 10 times to account for its stochastic behavior and report the mean deviation, *mad*, and the average time in seconds. Each problem instance was given 100 and 10,000 samples. The plug-in method was run once since it does not change when given the same data set. See Table 1 for the estimands expressions. We see that on 10,000 samples the mean deviation on all models tested was very small and comparable to the estimand-based plug-in, which is guaranteed to converge to the exact answer given enough samples. However, on a reduced sample size of 100 samples, EM4CI was even superior to the plug-in, maybe because EM4CI has an advantage against the plug-in in the sizes of the CPTs to be estimated, which only manifests itself in small sample size situations. EM4CI was also significantly faster than the plug-in in most cases, yet note that the plug-in method is implemented in Python, while EM4CI is implemented in C++, so a real time comparison is hard to make. Additional domain choices yielded similar behavior on these small models (see the Supplemental).

Results on large models. In Tables 3 and 4 we present the results on the large models of chains, diamonds, and cone-clouds graphs. For each graph we created several full models by varying the domain sizes of the observed (d) and latent variables (k). For each model and query we ran the EM4CI algorithm using different hypothesized sizes of the latent variable k_{hyp}. The estimand expressions for these models and queries were large both in the functions size involved and in the number of summation variables, so the estimand-based methods are infeasible and are not reported. We performed extensive experimentation. We report here only on a representative subset of our experiments due to space constraints (see the Supplemental).

Basic mode results Table 3 reports the algorithm's performance on selected sets of models from each graph type,

Table 2: Mean deviation results for EM4CI and Plug-in estimates on $P(Y = 0|do(X = 0))$ with $d = 2, k = 10, k_{hyp} = 16$

	100 Samples		10,000 Samples	
(Model, True Value)	EM4CI (mad, time(s))	Plugin (error, time(s))	EM4CI (mad, time(s))	Plugin (error, time(s))
(1, 0.530)	(0.0374, 0.006)	(0.054, 0.108)	(0.0008, 0.62)	(0.0009, 1.40)
(2, 0.481)	(0.0155, 0.025)	(0.0708, 0.029)	(0.013, 2.47)	(0.0144, 0.75)
(3, 0.519)	(0.022, 0.024)	(0.0393, 0.016)	(0.0087, 2.36)	(0.01, 0.55)
(4, 0.479)	(0.0186, 0.045)	(0.0459, 0.041)	(0.0061, 0.04)	(0.0012, 1.56)
(5, 0.489)	(0.0085, 0.097)	(0.0123, 0.024)	(0.0137, 9.49)	(0.0123, 0.65)
(6, 0.422)	(0.1097, 0.007)	(0.2915, 0.022)	(0.0193, 0.72)	(0.1821,0.74)
(7, 0.559)	(0.0192, 0.007)	(0.0804, 0.021)	(0.0007, 0.71)	(0.0364, 0.73)
(8, 0.512)	(0.105, 0.007)	(0.0956, 0.027)	(0.0049, 2.66)	(0.0019, 0.61)

(a) Chain model with $n = 49$
$(d, k) = (2, 16)$
Query: $P(V_{48}|do(V_0 = 1))$

	1,000 Samples			10,000 Samples		
k_{hyp}	mad	avg LL	time(s)	mad	avg LL	time(s)
2	0.01196	-31047.6	0.45	0.004986	-310726	4.41
8	0.01198	-31048.6	0.55	0.004987	-310723	5.45
16	0.01199	-31048.7	0.85	0.004989	-310723	8.4
32	0.01199	-31048.7	2.06	0.004989	-310723	20.5

(b) Diamond model with $n = 65$
$(d, k) = (4, 8)$
Query: $P(V_{64}|do(V_0 = 0))$

	1,000 Samples			10,000 Samples		
k_{hyp}	mad	avg LL	time(s)	mad	avg LL	time(s)
2	0.02677	-82867.3	3.92	0.00587	-837197	8.2
8	0.02669	-83114	1.71	0.00586606	-837209	16.7
16	0.02669	-83114.8	15.9	0.0058661	-837210	137.4
32	0.02669	-83115	402.9	0.00586609	-837210	3794.7

(c) Cone-cloud model with $n = 45$
$(d, k) = (2, 8)$
Query: $P(V_0|do(V_{36} = 1, V_{44} = 1, V_4 = 1))$

	1,000 Samples			10,000 Samples		
k_{hyp}	mad	avg LL	time(s)	mad	avg LL	time(s)
2	0.002548	-27356.7	0.47	0.003688	-274029	4.5
8	0.002398	-27356.8	0.81	0.003709	-274027	8.15
16	0.002396	-27356.8	2.94	0.003710	-274027	29.7
32	0.002396	-27356.8	27.8	0.003711	-274027	273.6

Table 3: Each table shows results from all potential domain sizes for one pair of observed and latent domain sizes (d, k). Here $n = |\boldsymbol{V}|$, $d = |D(V)|$, $k = |D(U)|$ in the true model, and k_{hyp} the hypothesized domain of the learned model. We display the mad, calculated and averaged over the target variable's domain. Minimum errors and maximum log likelihoods are highlighted.

where for each instance we vary the hypothesized domain sizes and show the accuracy and the log-likelihood results per k_{hyp}. We show the results for two sample sizes of 1,000 and 10,000. The queries for each model type appear in the Figure heading. Notice that each line corresponds to the average accuracy over each of the values of the targeted variable, as explained earlier. We clearly see that the accuracy of the algorithm is very high for both sample sizes of 1,000 and 10,000 as reflected by the accuracy measure mad. As expected we observe both higher accuracy and more time as the sample size increases. However, we observe very little impact due to the hypothesized domain sizes selected. In particular there is no clear pattern on how the latent variable's domain sizes influence the algorithm's query accuracy and model's log-likelihood. On the other hand, the choice of latent domain sizes definitely impacted the algorithm's execution time (see increase in rows in the time column.)

Iterated mode results. Since choosing the appropriate latent domain sizes is elusive, in Table 4 we now show the results when running the algorithm in the iterated mode. The table exposes the algorithm to more models that vary by the variable domain sizes ($(d, k) \subseteq \{(2, 8), (2, 16), (4, 4), (4, 8)\}$). We report accuracy (avg mad) and avg LL, (which would occur at different k_{hyp} val-

(a) Chain Model with $n = 49$
Query for $d = 2$: $P(V_{48}|do(V_0 = 1))$
Query for $d = 4$:$P(V_{48}|do(V_0 = 0))$

		1,000 Samples				10,000 Samples			
d	k	mad	avg LL	time(s)	mode(k_{hyp})	mad	avg LL	time(s)	mode(k_{hyp})
2	8	0.004622	-30876.9	1.5	2	0.003179	-309852	24.9	8,16
2	16	0.01196	-31047.6	1.5	2	0.004985	-310722	15.7	8
4	4	0.008875	-62454.9	1	2	0.003802	-626488	13.7	2,8
4	8	0.006826	-63055.9	1	2	0.001832	-633980	14.9	2

(b) Diamond Model with $n = 65$
Query for $d = 2, 4$: $P(V_{64}|do(V_0 = 0))$

		1,000 Samples				10,000 Samples			
d	k	mad	avg LL	time(s)	mode(k_{hyp})	mad	avg LL	time(s)	mode(k_{hyp})
2	8	0.00778	-40364.6	5	2	0.00021	-112442	901.7	8
2	16	0.01578	-40935.2	5.6	2	0.00418	-409056	497.5	8
4	4	0.005213	-82377.8	2.7	2	0.00504	-834986	38.6	2
4	8	0.026772	-82867.3	2.7	2	0.00587	-837197	240.5	2

(c) Cone Cloud Model with $n = 45$
Query for $d = 2, 4$: $P(V_0|do(V_{36} = 1, V_{44} = 1, V_4 = 1))$

		1,000 Samples				10,000 Samples			
d	k	mad	avg LL	time(s)	mode(k_{hyp})	mad	avg LL	time(s)	mode(k_{hyp})
2	8	0.00255	-27355.8	2.8	2,8	0.003657	-274024	24.5	2,8
2	16	0.01558	-26806.4	1.9	2	0.006846	-266517	24.4	2,8
4	4	0.01201	-55192.5	6	2	0.003809	-561970	42.4	8
4	8	0.02230	-55362	6.3	2	0.003376	-563135	37.3	8

Table 4: Results on Chain, Diamond and Cone-cloud Models from the iterated mode of EM4CI, increasing domain sizes $k_{hyp} = \{2, 8, 16, 32\}$. Displayed is the the highest average LL of the output learned model, its *mad*, calculated and averaged over the target variable's domain, avg time until termination. mode(k_{hyp}) is the hypothesized domain size(s) that most frequently produced the highest log-likelihood model.

ues across the 10 runs and across different values of the target variable). We also report the average total time at termination and provide information on the k_{hyp} at termination.

Parts (a,b) of Table 4 show **results on chains** having 49 observed variables. For sample size of 1,000 the algorithm was fast and the best performance was obtained in the first iteration. For 10,000 samples the results are more accurate and naturally required more time. These results are reassuring but not surprising, since we expected good learning and inference on models with small treewidth. What is notable is that *the plug-in would not produce reasonable results* due to the estimand's complex structure as shown in Equation (1). The **results on diamonds** (Table 4 c,d) had a similar pattern to chains. We ran on models of size 65 so performance times were higher, but accuracy was very high and often obtained for $k_{hyp} = 2$ (for 1,000 samples), and sometimes at higher latent domains sizes for 10,000 samples. We observe higher accuracy and more time when we used 10,000 samples. Note that, here too the estimand-based approaches (the plug-in) are infeasible due to their highly complex expressions generated by the ID algorithm. Finally, the **results on cone-cloud** (Table 4 e,f) were also highly accurate exhibiting the same pattern of performance.

WERM comparison The results for Models 1, 8, and 3', comparing the WERM [12] scheme to EM4CI are presented in Table 5. Model 3' is the same as Model 3 in Figure 2, with edge $Y \rightarrow Z$ reversed to $Z \rightarrow Y$ to match the hard-coded model in WERM. We used domain size $d = 2$, $k_{hyp} = \{2, 16\}$, and 1,000 samples. We see that the k_{hyp} that produced the smallest error varied between models and number of samples used. The best results with EM4CI (with the best k_{hyp}) provided smaller error than WERM in all cases except for Model 1 with 10,000 samples. Yet, both schemes are highly accurate for these small models as expected given the number of samples for the size of the models. Notice that EM4CI is far faster, yet WERM is implemented in R, while EM4CI is implemented in C++. Note that we were unable to compare with WERM on the large models for lack of general code - the code provided is hard-coded for the given models.

Summary. Our experiments here (and more in the Supplemental) demonstrated that algorithm EM4CI produced highly accurate results on all the benchmarks in a timely manner. The algorithm performed as well or better than estimand-based schemes plug-in and WERM. On the current benchmarks, hypothesized domain size choices did not have a significant impact on accuracy of the results but impacted the running time since larger domain sizes require more time for learning the model. Granted, these models have small treewidth favorable for the learning-based scheme. Still, they showcase a whole class of benchmarks where standard estimand-based schemes may fail while this path will work well.

Summary and Conclusion

The paper introduces a computational method for causal inference, challenging the conventional practice of using the estimand-based approach. Instead of estimating the estimand of a given query from observational data, our approach learns the entire CBN from this data. Utilizing the EM algorithm, the EM4CI algorithm learns the model and addresses identifiable queries through standard PGM techniques. A key advantage lies in its one-time model learning, enabling streamlined computation of various identifiable queries. Our empirical analysis underscores its viability; the algorithm yields highly accurate answers surpassing current methods. In particular, its efficiency parallels PGM's performance when the model's structure has bounded treewidth. Future work includes further exploring model selection for latent domain sizes and exploring additional benchmarks to advance our grasp of this approach's scope and limitations. Our approach complements, rather than replaces, estimand-based methods, offering an alternative solution tailored to specific effectiveness scenarios. In particular when the graph's treewidth is too high, we can still employ PGM's approximation algorithms within the learning-based approach. This introduces a nuanced trade-off between model learning and estimand-based estimation, offering intriguing possibilities for exploration.

Comparison between WERM and EM4CI with $d = 2$

Model	$\|S\|$	True Value	WERM error	time(s)	EM4CI $k_{hyp} = 2$ error	time(s)	EM4CI $k_{hyp} = 16$ error	time(s)
1	10^3	0.0911	0.0071	18.7	0.002	0.043	0.0091	0.068
8	10^3	0.6972	0.1082	25.8	0.0769	0.069	0.1747	0.56
3'	10^3	0.496	0.027	27.2	0.0305	0.054	0.0114	0.427
1	10^4	0.0919	0.0031	32.6	0.0046	0.428	0.0034	0.655
8	10^4	0.699	0.11	47.7	0.0132	0.7	0.1315	5.636
3'	10^4	0.491	0.001	44.1	0.0006	0.549	0.0005	4.252

Table 5: Results of absolute error on Query $P(Y = 1|do(\mathbf{X} = 1))$ on Models 1, 8, & 3' by WERM and EM4CI with $k_{hyp} = \{2, 16\}$. Model 3' has one edge reversed from Model 3. Number of samples is $|S|$. The minimum error is highlighted.

References

[1] 2022. *QGeNIe Modeler USER MANUAL.*

[2] Bhattacharya, R.; Nabi, R.; and Shpitser, I. 2022. Semiparametric Inference For Causal Effects In Graphical Models With Hidden Variables. *Journal of Machine Learning Research*, 23: 1–76.

[3] Bhattacharyya, A.; Gayen, S.; Kandasamy, S.; Raval, V.; and Variyam, V. N. 2022. Efficient interventional distribution learning in the PAC framework. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, 7531–7549.

[4] Bishop, C. M. 2007. *Pattern Recognition and Machine Learning*.

[5] Chen, Y.; and Darwiche, A. 2022. On the definition and computation of causal treewidth. In Cussens, J.; and Zhang, K., eds., *Uncertainty in Artificial Intelligence, UAI, 2022*.

[6] Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

[7] Darwiche, A. 2022. Causal Inference Using Tractable Circuits. *CoRR*.

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

[9] Dempster, L. N. R. D., A.P. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B.*, 39: 1–38.

[10] Huang, Y.; and Valtorta, M. 2008. On the completeness of an identifiability algorithm for semi-markovian models. *Annals of Mathematics and Artificial Intelligence*, 54(4): 363–408.

[11] Jung, Y.; Tian, J.; and Bareinboim, E. 2020. Estimating Causal Effects Using Weighting-Based Estimators. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, 10186–10193.

[12] Jung, Y.; Tian, J.; and Bareinboim, E. 2020. Learning Causal Effects via Weighted Empirical Risk Minimization. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33*.

[13] Jung, Y.; Tian, J.; and Bareinboim, E. 2021. Estimating Identifiable Causal Effects on Markov Equivalence Class through Double Machine Learning. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*.

[14] Jung, Y.; Tian, J.; and Bareinboim, E. 2021. Estimating Identifiable Causal Effects through Double Machine Learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, 12113–12122.

[15] Koller, N., Daphne; Friedman. 2009. *Probabilistic Graphical Models*. The MIT Press.

[16] Lauritzen, S. L. 1995. The EM algorithm for graphical association models with missing data . *Computational Statistics Data Analysis*, 191–201.

[17] Mateescu, R.; Kask, K.; Gogate, V.; and Dechter, R. 2010. Join-Graph Propagation Algorithms. *J. Artif. Intell. Res.*, 37: 279–328.

[18] McDonald, R. P.; and Algina, J. 2005. Maximum Likelihood Estimation in Latent Variable Models with an EM Algorithm. *Psychometrika*, 70(2): 1–19.

[19] Pearl, J. 1989. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann.

[20] Pearl, J. 2009. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition.

[21] Shpitser, I.; and Pearl, J. 2006. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, 1219.

[22] Shpitser, I.; Richardson, T. S.; and Robins, J. M. 2012. An Efficient Algorithm for Computing Interventional Distributions in Latent Variable Causal Models. *CoRR*, abs/1202.3763.

[23] Tian, J. 2002. *Studies in Causal reasoning and Learning*. Ph.D. thesis, University of California, Los Angeles.

[24] Tian, J.; and Pearl, J. 2002. A General Identification Condition for Causal Effects. In Dechter, R.; Kearns, M. J.; and Sutton, R. S., eds., *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI)*, 567–573.

[25] Tikka, S.; and Karvanen, J. 2018. Enhancing identification of causal effects by pruning. *Journal of Machine Learning Research*, 18: 1–23.

[26] Tikka, S.; and Karvanen, J. 2018. Simplifying Probabilistic Expressions in Causal Inference. *J. Mach. Learn. Res.*, 18: 36:1–36:30.

[27] Xia, K.; Lee, K.; Bengio, Y.; and Bareinboim, E. 2021. The Causal-Neural Connection: Expressiveness, Learnability, and Inference. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *NeurIPS 34*, 10823–10836.

[28] Xia, K.; Pan, Y.; and Bareinboim, E. 2022. Neural Causal Models for Counterfactual Identification and Estimation. *CoRR*, abs/2210.00035.

[29] Y. Jung, J. T.; and Bareinboim, E. 2021. Double Machine Learning Density Estimation for Local Treatment Effects with Instruments. NeurIPS 2021.

[30] Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2000. Generalized Belief Propagation. In Leen, T. K.; Dietterich, T. G.; and Tresp, V., eds., *NIPS*.

[31] Zaffalon, M.; Antonucci, A.; and Cabañas, R. 2021. EM Based Bounding of Unidentifiable Queries in Structural Causal Models. *Why-21 workshop at NeurIPS*.

[32] Zhang, J.; Tian, J.; and Bareinboim, E. 2022. Partial Counterfactual Identification from Observational and Experimental Data. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, 26548–26558. PMLR.