Estimating Causal Effects from Learned Causal Networks

Anna K. Raichev[1] **Jin Tian**[2] **Alexander Ihler**[1] **Rina Dechter**[1]

[1]Computer Science Dept., University of California, Irvine, Irvine, CA, USA
[2]Computer Science Dept., University of Iowa, Iowa City, IA, USA

1 INTRODUCTION

The standard approach to answering an identifiable causal-effect query (e.g., $P(Y|do(X))$ when given a causal diagram and observational data is to first generate an estimand, or probabilistic expression over the observable variables, which is then evaluated using the observational data. In this paper, we propose an alternative paradigm for answering causal-effect queries over discrete observable variables. We instead learn the causal Bayesian network and its confounding latent variables directly from the data. Then, efficient probabilistic graphical model (PGM) algorithms can be applied to the learned model to answer queries. Surprisingly, we show that this *model completion* learning approach can be more effective than estimand approaches, particularly for larger models in which the estimand expressions become computationally difficult. We illustrate our method's potential using a benchmark collection of Bayesian networks and synthetically generated causal models.

2 BACKGROUND

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: (1) $\boldsymbol{U} = \{U_1, U_2, ..., U_k\}$ is a set of latent variables; (2) $\boldsymbol{V} = \{V_1, V_2, ..., V_n\}$ is a set of endogenous, observable variables; (3) $\boldsymbol{F} = \{f_i : V_i \in \boldsymbol{V}\}$ is a set of functions f_i such that each f_i determines the value v_i of V_i as a function of V_i's parents $PA_i \subseteq \boldsymbol{U} \cup (\boldsymbol{V} \setminus V_i)$; and (4) $P(\boldsymbol{U})$ is a probability distribution over the latent variables.*

Causal effect and the truncation formula. An external intervention, forcing variables $\boldsymbol{X}$ to take on value $\boldsymbol{x}$, is modeled by replacing the mechanism for each $X \in \boldsymbol{X}$ with the function $X = x$. This is formally represented by the do-operator $do(\boldsymbol{X} = \boldsymbol{x})$. Thus the interventional distribution of an SCM $\mathcal{M}$ by applying $do(\boldsymbol{X})$ is,

$$P(\boldsymbol{V} \setminus \boldsymbol{X}, \boldsymbol{U} \mid do(\boldsymbol{X} = \boldsymbol{x})) = \prod_{V_j \notin \boldsymbol{X}} P(V_j|PA_j) \cdot P(\boldsymbol{U}) \Big|_{\boldsymbol{X}=\boldsymbol{x}} \tag{1}$$

Example 1. *Consider the model in Figure 1. To evaluate the query $P(V_6 \mid do(V_0))$, the ID algorithm Tian [2002], Shpitser and Pearl [2006] gives the expression:*

$$P(V_6 \mid do(V_0)) = \sum_{V_1,V_2,V_3,V_4,V_5} P(V_5|V_0,V_1,V_2,V_3,V_4) \times P(V_3|V_0,V_1,V_2)\, P(V_1|V_0) \sum_{v_0} P(V_6|v_0,V_1,V_2,V_3,V_4,V_5) \times P(V_4|v_0,V_1,V_2,V_3)\, P(V_2|v_0,V_1)\, P(v_0). \tag{2}$$

Unfortunately, in large models the expression elements become unwieldy. In terms of scalability, the required distributions have exponentially many configurations, suggesting they may require a significant amount of data to estimate accurately. Moreover, the required marginalizations are also over high-dimensional spaces, potentially also making them computationally intractable. These issues make it difficult to apply statistically sophisticated or machine learning-based estimators Jung et al. [2020a,b, 2021a,b] in such settings.

Alternatively, we can often maintain tractability by using the simple *plug-in* estimator, in which each term is estimated only on the configurations seen in the observed data. This reduces computation, since each term is non-zero on only a subset of configurations. However, this approach also limits the quality of our estimates. These issues motivate us to explore the effectiveness of a *model completion* approach.

3 LEARNING-BASED CAUSAL INFERENCE

The approach we propose for the causal-effect task is to first learn a full CBN $\mathcal{B} = \langle \mathcal{G}, \mathcal{P}\rangle$ given the causal diagram $\mathcal{G} = \langle \boldsymbol{V} \cup \boldsymbol{U}, E\rangle$ and samples from the observational distribution $P(\boldsymbol{V})$. Then, we can answer causal-effect queries based on the truncated formula Eq. (1) using probabilistic inference over the learned CBN. However, there could be many parameterizations $\mathcal{P}$ that are consistent with the same observational distribution $P(\boldsymbol{V})$. Luckily, the identifiability property ensures that the problem remains well-posed: as long as the query is identifiable, any of these alternative parameterizations $\mathcal{P}$ will generate the same answer:

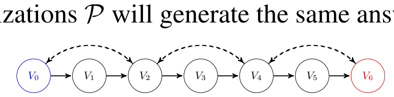


Figure 1: Causal diagram of a chain model. Dashed bidirected edges represent latent variables.

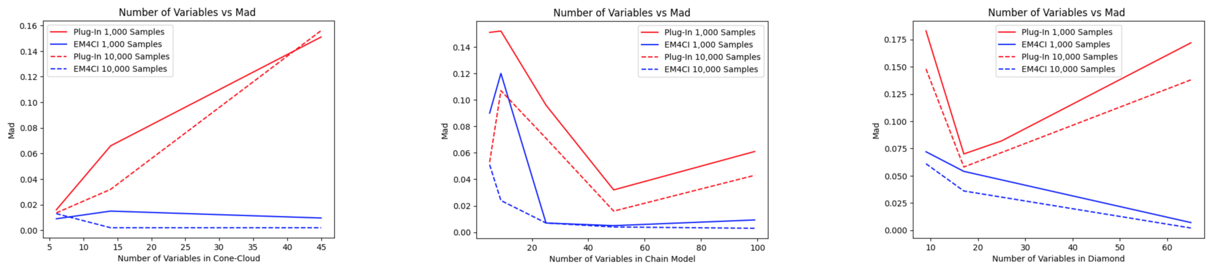


Figure 2: Comparing the accuracy of EM4CI and Plug-In. While both methods improve with more samples (solid to dashed lines), the error (mad) of EM4CI is smaller, even when compared to Plug-In with more samples.

Algorithm 1: EM4CI

input : A causal diagram $\mathcal{G} = \langle \boldsymbol{U} \cup \boldsymbol{V}, E \rangle$, $\boldsymbol{U}$ latent and $\boldsymbol{V}$ observables; $\mathcal{D}$ samples from $P(\boldsymbol{V})$;
output :Estimated $P(\boldsymbol{Y} \mid do(\boldsymbol{X} = \boldsymbol{x}))$

// k= latent domain size, $BIC_{\mathcal{B}}$ = BIC score of $\mathcal{B}$, $\mathcal{D}$,
// $LL_{\mathcal{B}}$ is the log-likelihood of $\mathcal{B}$, $\mathcal{D}$
Initialize: $BIC_{\mathcal{B}} \leftarrow \inf$,
If $\neg\,$identifiable($\mathcal{G}, Q$), terminate.
for $k = 2, ...,$ to upper bound, **do**
  $(\mathcal{B}', LL_{\mathcal{B}'}) \leftarrow \max_{LL} \{\text{EM}(\mathcal{G}, \mathcal{D}, k) | \textbf{for } i = 1 \text{ to } 10\}$
  Calculate $BIC_{\mathcal{B}'}$ from $LL_{\mathcal{B}'}$
  if $BIC_{\mathcal{B}'} \leq BIC_{\mathcal{B}}$,
    $\mathcal{B} \leftarrow \mathcal{B}'$, $BIC_{\mathcal{B}} \leftarrow BIC_{\mathcal{B}'}$
  else, break.
endfor
$\mathcal{B}_{\boldsymbol{X}=\boldsymbol{x}} \leftarrow$ generate truncated CBN from $\mathcal{B}$.
return $\leftarrow$ evaluate $P_{\mathcal{B}_{\boldsymbol{X}=\boldsymbol{x}}}(\boldsymbol{Y})$

Complexity of EM4CI The complexity of our approach is dominated by the time to learn the model. For T iterations of EM we find that the complexity is $O(T \cdot |D| \cdot n \cdot l^w)$, where $n = |V \cup U|$ is the number of variables, sample size $|D|$, l bounds the variable domain sizes, and w is the treewidth. Note that the cost of the EM learning process can be amortized over multiple queries.

4 EMPIRICAL EVALUATION

Benchmarks We use two sources for our benchmarks: synthetically generated models, and real domains from the academic literature of various fields. We examine three scalable classes of graphs whose treewidths vary but can be controlled: chain networks (treewidth 3), diamond networks (treewidth 5), and cone-cloud networks (treewidth $O(\sqrt{n})$. The parameters of each CPT were generated by sampling from a Dirichlet distribution Darwiche [2009]. We also test on networks created for real-world domains. The "A" network, from the UAI literature, is synthetic but known to be daunting to exact algorithms Kozlov and Singh [1996];

Measures of performance. We report the results of EM4CI along the two phases of the algorithm. For the learning process, we report the selected latent domain sizes and the total time at termination. For the inference phase we report the time and *mean absolute deviation (mad)* between the true answer and the estimated answer. The measure mad for a query $P(Y \mid do(\boldsymbol{X}))$ is computed by averaging the absolute error over all single-value queries over all instantiations of the intervened and queried variables, $P(Y = \mathbf{y} \mid do(\boldsymbol{X} = \boldsymbol{x}))$ for $\boldsymbol{x}, \mathbf{y} \in \mathcal{D}(\boldsymbol{X}) \times \mathcal{D}(\mathbf{Y})$.

Results on large synthetic models. Figure 2 shows the accuracy trends for each class as a function of model sizes, for 1,000 and 10,000 samples. We can see that generally, EM4CI using only 1,000 samples is even more accurate than *Plug-In* with 10,000 samples. However, the increase in accuracy comes at the cost of longer learning times.

We expect this improvement is due to the variance reduction of the estimation process. Model completion exploits more information from the causal graph than is apparent in the estimand expression. We can also include simple complexity control, with the latent domain sizes, to further reduce variance. In contrast, it is difficult to impose meaningful regularity or variance reduction on the plug-in estimates. Thus our results suggest that whenever the causal graph's treewidth is bounded and the estimand expression has large scope functions, we should prefer using model completion.

Results on A model. Evaluation for multiple queries for EM4CI and Plug-In are given in Table 1 for the **A network**. Learning time grows with sample size. Accuracy results are excellent and improve with increased sample sizes as well. We highlight how learning time of EM4CI can be amortized effectively over multiple queries. In contrast, the *Plug-In* method requires estimation of each new query from scratch, even if on the same model. Thus, for multiple queries, EM4CI may take far less time per query, while providing superior quality estimates.

Table 1: Plug-In & EM4CI results on the **A Network**
$|\boldsymbol{V}| = 46;\ |\boldsymbol{U}| = 8;\ d = 2;\ k = 2$ treewidth ≈ 16

	Plug-In				*EM4CI*			
	1,000 Samples		*10,000 Samples*		*1,000 Samples*		*10,000 Samples*	
Query	mad	time(s)	mad	time(s)	mad	time(s)	mad	time(s)
$P(V_{51}\|\text{do}(V_{10}))$	0.0584	8.0	0.0114	55.7	0.0139	0.0012	0.0083	0.0012
$P(V_{51}\|\text{do}(V_{14}))$	0.0319	8.3	0.0056	51.3	0.0143	0.0047	0.0086	0.0046
$P(V_{51}\|\text{do}(V_{41}))$	0.0255	13.9	0.0092	48.3	0.0147	0.0042	0.0079	0.0041
$P(V_{51}\|\text{do}(V_{45}))$	0.0496	9.8	0.0206	49.1	0.0140	0.0031	0.0082	0.0030
EM4CI Learning	time=71(s), $k_{lrn} = 4$ (1,000 Samples) and time=541(s), $k_{lrn} = 4$ (10,000 Samples)							

References

Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

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.

Yonghan Jung, Jin Tian, and Elias Bareinboim. Estimating identifiable causal effects through double machine learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 12113–12122, 2021a.

Yonghan Jung, Jin Tian, and Elias Bareinboim. Estimating identifiable causal effects on markov equivalence class through double machine learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, 2021b.

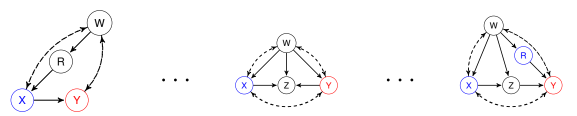
Alexander V. Kozlov and Jaswinder Pal Singh. Parallel implementations of probabilistic inference. *IEEE Computer*, pages 33–40, 1996.

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

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.

Estimating Causal Effects from Learned Causal Networks (Supplementary Material)



(a) Model 1 . . . **(b)** Model 3 . . . **(c)** Model 8

Figure 3: A subset of the small causal diagrams for models used in our experiments. Blue variables are intervened on and red variables are the outcome variables corresponding to the query $P(Y \mid do(X))$.

A ADDITIONAL RESULTS

Results on small synthetic models. Results on small models (Figure 3) are presented in Table 2. In all tables d and k represent the cardinality of the domains for observed and latent variables, respectively. Since these models are quite small we report the total time (learning plus inference) for EM4CI. We compare against the *Plug-In* method, which is guaranteed to converge to the exact answer. Therefore, we expect *Plug-In* to produce fairly accurate results on these small models if given enough samples. We observe that the accuracy of both methods are similar at both 100 and 1,000 samples, with EM4CI being more accurate on some cases, and *Plug-In* on others. EM4CI had better time performance for 100 samples, but for 1,000 samples the *Plug-In* was faster since learning time of EM4CI was longer.

WERM comparison. The results for Models 1, 8, and 3' [1] comparing WERM Jung et al. [2020b] to EM4CI are given in Table 3. We use domain size $d = 2$, and 1,000 and 10,000 samples. Again, EM4CI produced more accurate results in several instances, though the disparities are smaller than with the Plug-In method. EM4CI was faster than WERM with 1,000 samples but slower with 10,000 samples.

Table 2: Results of EM4CI & Plug-In on $P(Y|do(\mathbf{X}))$ $(d, k) = (2, 10)$

	100 Samples					1,000 Samples				
	EM4CI			Plug-In		EM4CI			Plug-In	
Model	k_{lrn}	mad	time(s)	mad	time(s)	k_{lrn}	mad	time(s)	mad	times(s)
1	2	0.0037	0.4759	0.0104	1.9	2	0.0032	3.1	0.0025	2.3
2	2	0.1832	1.8643	0.1436	2.3	2	0.0490	8.4	0.0867	2.0
3	2	0.1288	0.9288	0.0569	1.1	2	0.0040	3.6	0.0039	0.7
4	2	0.1819	1.8169	0.1469	2.3	2	0.1438	12.0	0.0704	2.1
5	2	0.4910	1.6539	0.5000	2.0	2	0.0044	17.3	0.0058	2.2
6	2	0.2663	0.3004	0.3930	2.0	2	0.1263	0.5	0.1319	2.1
7	2	0.2520	0.7757	0.2509	1.9	2	0.0891	7.1	0.0238	2.0
8	2	0.1372	0.6348	0.1579	2.0	2	0.2340	4.7	0.1303	1.9

[1] model 3' is the same as Model 3 in Figure 3, but with edge $Y \to Z$ reversed to match the hard-coded model in WERM.

Table 3: Results of absolute error on Query $P(Y = 1|do(\mathbf{X} = 1))$ on Models 1, 8, & 3' by WERM and EM4CI. k_{lrn} is the learned domain sizes of latent variables.

	1,000 Samples					10,000 Samples				
	WERM		EM4CI			WERM		EM4CI		
Model	error	time(s)	error	time(s)	k_{lrn}	error	time(s)	error	time(s)	k_{lrn}
1	0.0071	18.7	0.0059	8.8	2	0.0031	32.6	0.0046	63.5	2
8	0.1082	25.8	0.1566	7.6	2	0.11	47.7	0.0001	81.4	2
3'	0.027	27.2	0.0004	3.5	2	0.001	44.1	0.0009	53.1	2

Table 4: Results for EM4CI & Plug-In on $P(Y|do(\mathbf{X}))$ $(d, k) = (4, 10)$ CH-Chain, CC-Cone-Cloud, D-Diamond network

(a) 1,000 samples

		EM4CI				Plug-In	
Model	Query	k_{lrn}	mad	Learn-time(s)	inf-time(s)	mad	time(s)
5-CH	$P(V_4\|do(V_0))$	4	0.0902	3.5	0.0001	0.1509	2.3
9-CH	$P(V_8\|do(V_0))$	4	0.1204	11.5	0.0002	0.1516	2.4
25-CH	$P(V_24\|do(V_0))$	2	0.0070	77.7	0.0003	0.0959	6.1
49-CH	$P(V_48\|do(V_0))$	4	0.0005	161.2	0.0007	0.0319	17.8
99-CH	$P(V_98\|do(V_0))$	6	0.0093	413.4	0.0023	0.0611	88.1
9-D	$P(V_8\|do(V_0))$	2	0.0719	24.6	0.0002	0.1832	3.4
17-D	$P(V_{16}\|do(V_0))$	6	0.0542	202.3	0.0006	0.0700	4.5
65-D	$P(V_{64}\|do(V_0))$	4	0.0074	432.4	0.0012	0.1716	232.5
6-CC	$P(V_0\|do(V_5))$	4	0.0088	23.5	0.0001	0.0156	2.3
15-CC	$P(V_0\|do(V_{14}))$	4	0.0147	60.8	0.0001	0.0659	4.5
45-CC	$P(V_0\|do(V_{14}, V_{36}, V_{44}))$	6	0.0097	199.2	2.7429	0.1509	18.6

(b) 10,000 samples

		EM4CI				Plug-In	
Model	Query	k_{lrn}	mad	Learn-time(s)	inf-time(s)	mad	time(s)
5-CH	$P(V_4\|do(V_0))$	4	0.0508	17.3	0.0001	0.0537	2.5
9-CH	$P(V_8\|do(V_0))$	4	0.0236	150.0	0.0002	0.1074	3.1
25-CH	$P(V_24\|do(V_0))$	6	0.0068	697.1	0.0005	0.0714	26.4
49-CH	$P(V_48\|do(V_0))$	10	0.0017	2412.6	0.0036	0.0160	133.7
99-CH	$P(V_98\|do(V_0))$	6	0.0028	3887.9	0.0022	0.0433	850.6
9-D	$P(V_8\|do(V_0))$	4	0.0611	390.7	0.0002	0.1481	3.0
17-D	$P(V_{16}\|do(V_0))$	6	0.0360	1849.6	0.0007	0.0582	8.4
65-D	$P(V_{64}\|do(V_0))$	4	0.0022	4787.2	0.0013	0.1376	2258.5
6-CC	$P(V_0\|do(V_5))$	6	0.0138	116.9	0.0003	0.0136	2.7
15-CC	$P(V_0\|do(V_{14}))$	4	0.0022	489.5	0.0043	0.0321	10.9
45-CC	$P(V_0\|do(V_{14}, V_{36}, V_{44}))$	6	0.0026	1833.7	2.757	0.1561	105.8

Unfortunately, the code for WERM is specific to these models, so we were unable to compare against it more generally. This also highlights a general lack of available estimand-based implementations applicable to general settings.

Results on large synthetic models. In Tables 4 we show results on larger models of chains, diamonds, and cone-cloud graphs. The first two tables compare EM4CI to the plug-in method for a single query over all the models. Specifically, Table 4a presents time and accuracy results for 1,000 samples. We see that EM4CI was consistently more accurate, and in many cases significantly better (e.g., in 45-cone-cloud). We see the same trend for 10,000 samples in Table 4b. Focusing on time, we see that the time of EM4CI is significantly more costly than *Plug-In*, with EM learning be-

Table 5: EM4CI on large synthetic models

(a) 99 chain:
$|\boldsymbol{V}| = 99$; $|\boldsymbol{U}| = 49; d = 4; k = 10$. treewidth=3

	1,000 Samples		*10,000 Samples*	
Learning	time(s)	k_{lrn}	time(s)	k_{lrn}
	413.4	6	3887.9	6
Inference				
Query	mad	time(s)	mad	time(s)
$P(V_{98}\|do(V_0))$	0.0093	0.0023	0.0028	0.0022
$P(V_{49}\|do(V_0))$	0.0113	0.0011	0.0041	0.0011
$P(V_{98}\|do(V_{49}))$	0.0093	0.0011	0.0028	0.0011
$P(V_{98}\|do(V_{90}))$	0.0152	0.0004	0.0063	0.0004

(b) 65 diamond:
$|\boldsymbol{V}| = 65$; $|\boldsymbol{U}| = 32; d = 4; k = 10$. treewidth=5

	1,000 Samples		*10,000 Samples*	
Learning	time(s)	k_{lrn}	time(s)	k_{lrn}
	432.3	4	4787.2	4
Inference				
Query	mad	time(s)	mad	time(s)
$P(V_{64}\|do(V_0))$	0.0074	0.0012	0.0022	0.0013
$P(V_{32}\|do(V_{16}))$	0.0193	0.0004	0.0046	0.0005
$P(V_{16}\|do(V_0))$	0.0283	0.0004	0.0150	0.0005
$P(V_{48}\|do(V_{32}))$	0.0086	0.0004	0.0093	0.0004

ing the most time consuming part. Interestingly, while time grows with model size for both schemes, the inference time component remains efficient, likely due to the low treewidth of some of the models (e.g., the chains and the diamonds). In the 45-cone case, inference time is impacted more by model size, since its treewidth increases with the square root of the number of variables. Generally, for a single query, we find *Plug-In* has better time performance, and its time increases at a slower pace. In both methods, time also increases with sample size, e.g., when moving from 1,000 samples table to 10,000 samples.

In table 5a and 5b, we see the results of EM4CI on the **99-chain** and **65-diamond** on multiple queries. We see the same trend discussed in the paper, where learning time is longer than inference time, and grows with sample size. Again, the learned domain sizes are the same for both sample sizes, and close to the true domain size of the latent variables. We also see that inference is very fast and therefore we can amortize the learning time over multiple queries if desired. Finally, we see that EM4CI is very accurate.

Results on real networks. In 6 we see results on the Plug-In method and EM4CI on the **Alarm network**. Again, we see that EM4CI is more accurate than the Plug-In, but the learning time is longer. However, for multiple queries, EM4CI may take less time per query, while providing superior quality estimates.

Lastly in 7 the results for EM4CI on the **Barley network** and the **Win95** are displayed in the context of multiple queries, illustrating a similar pattern. For the **Barley network** (Table 7a), the learned domain sizes are large ($k_{lrn} = 14$) and 10

Table 6: Plug-In & EM4CI results on the **Alarm Network**
$|\boldsymbol{V}| = 32$; $|\boldsymbol{U}| = 5$; $2 \leq d \leq 4$; $2 \leq k \leq 3$. treewidth≈ 3

(a) Plug-In

	1,000 Samples		*10,000 Samples*	
Query	mad	time(s)	mad	time(s)
$P(HRBP\|$do(Shunt))	0.0190	2.9	0.0075	5.7
$P(HRBP\|$do(VentAlv))	0.0433	3.5	0.0212	5.7
$P(HR\|$do(VentAlv))	0.04706	4	0.0184	5.53
$P(HR\|$do(Shunt))	0.0229	3.0	0.00296	6.4

(b) EM4CI

	1,000 Samples		*10,000 Samples*	
Learning	time(s)	k_{lrn}	time(s)	k_{lrn}
	16	2	181	4
Inference				
Query	mad	time(s)	mad	time(s)
$P(HRBP\|$do(Shunt))	0.0076	0.0002	0.0043	0.0002
$P(HRBP\|$do(VentAlv))	0.0146	0.0003	0.0033	0.0003
$P(HR\|$do(VentAlv))	0.0106	0.0002	0.0020	0.0003
$P(HR\|$do(Shunt))	0.0101	0.0002	0.0027	0.0002

for both sample sizes, and accordingly learning time is also large for both settings. However, inference time remains very low.

Summary. Our experiments illustrate the strength of the direct-learning approach as a viable alternative for answering causal effect queries. We saw that model-completion by learning implemented in our EM4CI yields highly accurate estimates of causal effect queries on a variety of benchmarks, both synthetic networks and real life networks, on both small and large models. Moreover EM4CI shows clear superiority to the Plug-In estimands approach. In terms of time efficiency however, EM4CI was consistently slower due to learning time overhead. Yet, when answering multiple queries is desired the time overhead can be amortized over multiple queries.

Table 7: EM4CI results on the Real Networks

(a) "Barley":
$|\boldsymbol{V}| = 42$; $|\boldsymbol{U}| = 6$; $2 \leq d \leq 67; 2 \leq k \leq 9$. treewidth$\approx 7$

	1,000 Samples		*10,000 Samples*	
Learning	time(s)	k_{lrn}	time(s)	k_{lrn}
	199	14	820	10
Inference				
Query	mad	time(s)	mad	time(s)
P(Protein|do(expYield))	0.0066	0.9038	0.0031	0.8882
P(FieldCap|do(protein))	0.0108	0.0004	0.0032	0.0004
P(expYield|do(precipitation))	0.0284	0.0002	0.0064	0.0002
P(weight|do(precipitation))	0.0107	0.0002	0.0023	0.0002

(b) "Win95":
$|\boldsymbol{V}| = 59$; $|\boldsymbol{U}| = 17$; $d = 2; k = 2$ treewidth≈ 8

	1,000 Samples		*10,000 Samples*	
Learning	time(s)	k_{lrn}	time(s)	k_{lrn}
	109	2	1081	4
Inference				
Query	mad	time(s)	mad	time(s)
P(Ouput|do(NnPsGrphic))	0.0113	0.0002	0.0008	0.0002
P(PrintData|do(localOK))	0.0768	0.0002	0.0049	0.0002
P(PCtoPRT|do(netOK))	0.0167	0.0001	0.0141	0.0002
P(PrintData|do(lnetOK))	0.0116	0.0002	0.0016	0.0002