A Proofs of Lemmas and Theorems

A.1 Proof of Proposition 1

Proposition 1. *If the learning target is non-identifiable (i.e., every edge in the target is non-identifiable) a priori, then SCL is not better than random guessing.*

Re-statement: We take learning target as the orientation of an edge as an example, so we are analyzing the performance of a binary classifier against random guessing. The conclusion can be easily extended to general case.

Denote random guessing as a degenerated estimator $r(X) \equiv 0.5$, which indicates the probability of label = 1 is always 0.5, regardless of any input.

Denote the joint probability distribution of X and Y as $\mathbb{P}$ and the space of all joint probability distribution is $\mathcal{P}$, then we aim to prove the following statement which is in an adversarial setting:

$$r = \arg\min_{f \in \mathcal{C}} \sup_{\mathbb{P} \in \mathcal{P}} E_{(X,Y)\sim\mathbb{P}} \left[-Y \log f(X) - (1-Y)\log(1-f(X))\right]$$

The expectation is the standard binary cross entropy loss; we are allowed to enumerate every possible joint probability distribution in $\mathcal{P}$ because the learning target is non-identifiable. $\mathcal{C}$ is the space of all possible binary classifiers.

Proof. Given any binary classifier f, we partition the space of X by A, B and C where $A = \{x|f(x) > 0.5\}$, $B = \{x|f(x) < 0.5\}$, $C = \{x|f(x) = 0.5\}$. Then we construct the following joint probability distribution P^*:

$$P^*(X,Y) = \begin{Bmatrix} P^*(Y=0|X=x) = 1 \text{ if } x \in A \\ P^*(Y=1|X=x) = 1 \text{ if } x \in B \\ \text{arbitrary if } x \in C \end{Bmatrix}$$

Then it is easy to see that $E_{(X,Y)\sim P^*}\left[-Y \log f(X) - (1-Y)\log(1-f(X))\right] \geq 1$. Note that $E\left[-Y \log r(X) - (1-Y)\log(1-r(X))\right] \equiv 1$, thus r achieves minimum worse-case loss. □

A.2 Proof of Proposition 2

Proposition 2. *If ML4C-Learner is a perfect classifier, then ML4C outputs correct CPDAG of a canonical dataset (i.e., ML4C is perfect).*

Proof. Classical constraint-based methods consist of three steps: skeleton identification, v-structure identification, and further edge orientation by applying Meek rules [37]. It has been proved in PC [34] that when learning from a canonical dataset, if both the identified skeleton and v-structures are correct, then the learned CPDAG is correct. ML4C follows the three steps, with the correct skeleton is given as input, and ML4C-Learner is responsible for v-structure identification. Thus, assuming ML4C-Learner is a perfect classifier (i.e., correctly identifies all v-structures) implies that ML4C outputs correct CPDAG. □

A.3 Proof of Lemma 1

Lemma 1. *Sepsets $\mathcal{S}$ of any UT of a canonical dataset is non-empty.*

Proof. According to Lemma 3.3.9 of [35], in a directed acyclic graph G, if X is not a descendant of Y, and X and Y are not adjacent, then X and Y are d-separated by **Parents**(Y). Given an UT $\langle X, T, Y\rangle$, X and Y are not adjacent. Either X is not a descendant of Y, or Y is not a descendant of X, otherwise a loop will be introduced. Thus there either exists **Parents**$(X) \equiv PC_X \cup T$, or **Parents**$(Y) \equiv PC_Y \cup T$, which belongs to $\mathcal{S}$. Thus $\mathcal{S}$ is non-empty. □

A.4 Proof of Lemma 2

Lemma 2 (Existence of weak discriminative predicate)**.** *For a canonical dataset with infinite samples, the following are three weak discriminative predicates: i) $\{X \sim Y|T\} > 0$, ii) $\{X \sim Y|\mathcal{PC}_T\} = 0$, iii) $\{PC_X \sim PC_Y|\mathcal{S} \cup T\} > 0$.*

Proof. For a canonical dataset with infinite samples,

1. $\{X \sim Y|T\} > 0$: 1) $\langle X, T, Y\rangle$ is a v-structure $\Rightarrow$ T is a collider $\Rightarrow$ T unblocks X and Y through path $X - T - Y$ $\Rightarrow$ $\{X \sim Y|T\} > 0$ holds TRUE. 2) if $\langle X, T, Y\rangle$ is not a v-structure, then $\{X \sim Y|T\} > 0$ can be TRUE or FALSE. e.g., it is FALSE for $X \to T \to Y$ (no more paths connect X and Y), but if there exists another node $X \to T' \to Y$, it is evaluated TRUE. Therefore, it satisfies criterion ii) of definition 12, but not i) hence it is a weak discriminative predicate.

2. $\{X \sim Y|\mathcal{PC}_T\} = 0$: 1) $\langle X, T, Y\rangle$ is not a v-structure $\Rightarrow T$ is a non-collider $\Rightarrow \forall pc_t \in \mathcal{PC}_T$, there exists a path $X - T - Y$ from X to Y, where T is the only node on path, T is a non-collider, and $T \notin \{pc_t\} \Rightarrow pc_t$ does not block the path $\Rightarrow \{X \sim Y|\mathcal{PC}_T\} = 0$ always holds FALSE. 2) if $\langle X, T, Y\rangle$ is a v-structure, then $\{X \sim Y|\mathcal{PC}_T\} = 0$ can be TRUE or FALSE. Therefore, it satisfies criterion i) but not ii) hence it's a weak discriminative predicate.

3. $\{PC_X \sim PC_Y|\mathcal{S} \cup T\} > 0$: 1) $\langle X, T, Y\rangle$ is a v-structure $\Rightarrow T$ is a collider $\Rightarrow \forall pc_x \in PC_X, pc_y \in PC_Y, S \in \mathcal{S}$, $S \cup T$ unblock pc_x and pc_y through path $pc_x - X - T - Y - pc_y \Rightarrow \{pc_x \sim pc_y|S \cup T\} > 0$ always hold TRUE. 2) if $\langle X, T, Y\rangle$ is not a v-structure then it can be TRUE or FALSE. Therefore, it satisfies criterion ii) but not i) hence it's a weak discriminative predicate.

□

A.5 Proof of Lemma 3

Lemma 3 (Existence of strong discriminative predicate)**.** *For a canonical dataset with infinite samples, the following are three strong discriminative predicates: i)* $\text{OLP}(T, \mathcal{S}) = 0$*, ii)* $\text{OLP}(T, \mathcal{S}) < 0.5$*, iii)* $\text{OLP}(T, \mathcal{S}) < 1 \wedge \min\{X \sim Y|T \cup S\} > 0$*.*

Proof. First, it is known that the following three algorithms are sound and complete for a canonical dataset with infinite samples: CPC [28], MPC [8] and GLL-MB [3]. Below we translate each predicate and then show that these predicates are equivalent to the criterion to identify v-structures in CPC [28], MPC [8] and GLL-MB [3] respectively.

1. Predicate $\text{OLP}(T, \mathcal{S}) = 0 \iff \forall S \in \mathcal{S}, T \notin S$, which states that predicate is TRUE if and only if T is not in any d-separation set of X and Y. This is exactly the criterion of CPC for identifying v-structures [28].

2. Predicate $\text{OLP}(T, \mathcal{S}) < 0.5$ indicates that only if more than half of the d-separation sets do not contain T, then the UT is oriented as a v-structure. This is called majority-rule PC algorithm MPC [8] for v-structure identification.

3. Predicate $\text{OLP}(T, \mathcal{S}) < 1 \wedge \min\{X \sim Y|T \cup \mathcal{S}\} > 0 \Rightarrow \exists S \in \mathcal{S}, T \notin S$ and X and Y are dependent when conditioning on $T \cup S$, which is the criterion used for GLL-MB [3] to identify v-structures.

□

A.6 Proof of Theorem 1

Theorem 1. *ML4C-Learner tends to a perfect classifier on classifying a canonical dataset with sufficient samples.*

Proof. According to Lemma 3, there exists strong discriminative predicate P which achieves zero loss given a canonical dataset and sufficient samples. Thus, when adequate ML model is chosen, ML4C-Learner can achieve no worse performance than P (e.g., we can set the parameters of ML4C-Learner so that it approximates predicate P initially, and then apply standard gradient descent procedure). By considering proposition 2, we complete the proof. □

B Implementation Details

B.1 Calculating conditional dependencies

There are several ways to measure the conditional dependence, such as p-value by testing of conditional independence, or conditional mutual information [9]. For categorical variables, a good choice is G^2 test [1]. In our implementation, we adopt an approximate version of G^2 statistic, and use p-value to measure the conditional dependence.

Moreover, considering p-value can easily vanish due to numerical precision in 64-bit computers. Therefore, we use a transformation of p-value to avoid the issue, as additional quantity to measure conditional dependency. We first define complementary error function as

$$g(z) = 1 - \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \mathrm{d}t,$$

and we use quantity z by inverse of g:

$$z = g^{-1}(x).$$

Given a p-value x, we use g^{-1} as a non-linear transformation to obtain a better re-scaled quantity to measure conditional dependency. Intuitively, z can be viewed as z-sigma for a standard normal distribution, e.g., if p-value is 0.05, then $z = 2$, since 2-sigma indicates probability of values that lie within 2-sigma interval in a normal distribution is 0.95.

B.2 ML4C Training and inference details

B.2.1 Data synthesis details

Graph structure: We adopt the Erdős-Rényi (ER) model [13] and the Scale-Free (SF) model [2], which are two commonly used model for graph synthesis. We categorize the scale of the graph (number of nodes d) into four classes: small, medium, large, and very large, corresponding to d being uniformly sampled from intervals $[10, 20]$, $[21, 50]$, $[51, 100]$, and $[101, 1000]$, respectively. Given the number of nodes d, the sparsity of the graph (defined as the ratio of the average number of edges to the number of nodes, i.e., the average in-degree of all nodes) is randomly sampled from a uniform distribution $[1.2, 1.7]$. Given the number of nodes and the expected number of edges, the graph skeleton is generated accordingly by the two random graph models. Then the skeleton is randomly oriented to a DAG by upper triangular permutation.

Conditional probability table: Now we illustrate how we come up with Conditional Probability Table (CPT) for each node. In accordance with the topological ordering of the graph, each node is first assigned its cardinality, which is randomly sampled from a truncated normal distribution $\mathcal{N}(\mu = 2, \sigma = \frac{1.5}{m}, \min = 2)$, where m denotes the maximum number of peers of the node (i.e. $\max\{$in-degree of the effect nodes of this node$\}$). This regularization is designed to make the forward sampling process faster and prevent some certain nodes with many cause nodes from getting stuck. Since the number of different conditions to be enumerated is exponential ($\Pi_{c\in \text{ causes}}\text{cardinality}_c$), node with a larger maximum peers number tends to have smaller cardinality. Next, we enumerate each of its unique conditions (given by combinations of its cause nodes' cardinalities) and randomly generate its probability distribution at each condition. The probability distribution is sampled from a Dirichlet distribution with parameter $\alpha \sim U[0.1, 1.0]$ and grid number as this code's cardinality.

Training data: Having CPT specification of each node, a sample of 10k rows of observations is obtained for each graph according to the standard Bayesian network forward sampling. This generates a total dataset of 4 scales $\times$ 2 graph models $\times$ 50 graphs for each class = 400 unique graphs and the corresponding sampled data. Different SCL algorithms are then further used to extract the required features corresponding to the respective learning targets, e.g., all edges of all graphs for pairwise SCL algorithms. For our ML4C learning targets, all UTs are extracted from graphs, consisting of a total of 97,010 V-structures (label=1) and 195,691 non-V-structures (label=0).

B.2.2 XGBoost hyper-parameter settings

We use `xgb.XGBClassifier()`, the Python API provided by XGBoost [6], to implement the binary classifier ML4C-Learner. All hyper-parameters are set as default. We set the threshold value T = 0.1.

B.3 Post processing

Although ML4C-Learner achieves high accuracy on classifying UTs into v-structures or non-v-structures (UT-F1 = 0.9, as shown in Table 1), it is still possible to have conflicts among the detected v-structures. We adopt a straightforward heuristic to resolve conflicts: suppose we have two conflict v-structures $A \to B \leftarrow C$ and $B \to C \leftarrow D$, we discard the one with lower probability score (by ML4C-Learner). We continue such pairwise conflict resolving until no more conflicts exist. We use the left v-structures to construct the partial DAG (bottom-right of Figure 1(b)). Pseudo-code is shown in Algorithm 1.

C Details of Evaluation

C.1 Evaluation metrics

We calculate SHD at CPDAG level. Specifically, SHD is computed between the learned CPDAG($\hat{G}$) and ground truth CPDAG(G), i.e., the smallest number of edge additions, deletions, direction reversals and type changes (directed vs. undirected) to convert the output CPDAG to ground truth CPDAG. As is shown in Table 5, SHD is equal to the sum of the number of ✗s in the table.

F1-score is then calculated based on the identifiable edges of CPDAG($\hat{G}$) and CPDAG(G), where the accuracy (precision) is equal to True Positive Rate (TPR) and the recall (recall) is equal to 1 - False Discovery Rate (FDR).

input : v-structure candidates $VC = \{v_1, \cdots, v_p\}$,
score querier $s : v_i \rightarrow s_i$, returning v_i's probability score
output : Final v-structure candidates FV, which is self-consistent.
Initialize: removing v-structure set RV.
for $v_i \in VC$ **do**
    $s_i \leftarrow s(v_i)$
    flag$\leftarrow$ FALSE
    for $v_j \in VC$ **do**
        $s_j \leftarrow s(v_j)$
        if v_i conflicts with v_j **and** $s_i < s_j$ **then**
            flag$\leftarrow$ TRUE
            break
    if flag **then**
        SV $\leftarrow$ SV$\cup\{v_i\}$
FV$\leftarrow$VC$\setminus$RV.

Algorithm 1: Conflict resolving

Table 5: SHD calculation details.

in result CPDAG→ in truth CPDAG↓	ien (directed) right	iden (directed) wrong	uniden (undirected)	missing in skeleton
iden	✓ ①	✗ ②	✗ ③	✗ ④
uniden	✗ ⑤		✓ ⑥	✗ ⑦
nonexist	✗ ⑧		✗ ⑨	✓ ⑩

Details about the specific calculation can also refer to Table 5:

$$\text{precision=TPR} = \frac{\textcircled{1}}{\textcircled{1}+\textcircled{2}+\textcircled{3}+\textcircled{4}},$$

$$\text{recall=1-FDR} = \frac{\textcircled{1}}{\textcircled{1}+\textcircled{2}+\textcircled{5}+\textcircled{8}},$$

C.2 Full result of Table 1: End-to-end comparison

Here we report full results including other 5 smallest and trivial datasets. Note that 1) All F1-score degrade into 0. on sachs dataset, because that sachs has no identifiable edges. 2) The rank(SHD) row is also re-calculated over full datasets.

C.3 Predicates in Table 2: Reliability

Table 2 shows the performance of 4 weak discriminative predicates and 4 strong discriminative predicates. Specifically, the four strong predicates are respectively 1) $t \sim U[0,1]$, $\text{OLP}(T, \mathcal{S}) \geq t$; 2) $\text{OLP}(T, \mathcal{S}) == 0$; 3) $\text{OLP}(T, \mathcal{S}) == 0$ and $\{X \sim Y | \mathcal{S} \cup T\} > 0$; 4) $\{X \sim Y | \mathcal{S} \vee T\} > 0$. The four weak predicates are respectively 1) $\{PC_X \sim PC_Y | T\} > 0$; 2) $\{PC_X \sim PC_Y | \mathcal{S} \vee T\} > 0$; 3) $\{X \sim Y | PC_T\} == 0$; 4) $\{X \sim Y | \mathcal{S} \vee \mathcal{PC}_T\} > 0$.

C.4 Details of Table 4: Transferability

To evaluate ML4C's transferability across different domains, we train on dataset generated using one configuration, and test on another. By default the configuration is that: #nodes=50, sparsity=#edges/#nodes=1.5, generating model=ER, and sample size=10000. We conduct controlled trials on the four configuration domains listed above (shown as the four big bars of Table 4).

When we test transferability over one domain (e.g., the first bar, #nodes), then #nodes is set from 4 options (10, 50, 100, 1k), and $4 \times 4 = 16$ pairs of train-test experiments are conducted. For each experiment, 50 graphs are synthesized for training and another 5 graphs for test. Except for the target domain (#nodes), all the other

Table 6: Full result of Table 1.

Datasets		supervised					unsupervised								no skeleton input		
#nodes/#edges		ML4C	Jarfo	D2C	RCC	NCC	PC	CPC	MPC	GMB	GES	GS	HC	CDS	DGNN	BLIP	GNIP
cancer	SHD	**0**	4	3	4	**0**	2	**0**	**0**	**0**	3	**0**	**0**	2	4	4	**0**
5/4	F1	**1.0**	.00	.25	.00	**1.0**	.50	**1.0**	**1.0**	**1.0**	.25	**1.0**	**1.0**	.50	.00	.00	**1.0**
earthquake	SHD	**0**	4	2	4	**0**	2	**0**	**0**	**0**	**0**	**0**	4	**0**	4	5	**0**
5/4	F1	**1.0**	.00	.50	.00	**1.0**	.50	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	.00	**1.0**	.00	.00	**1.0**
survey	SHD	1	4	5	4	**0**	**0**	1	1	**0**	4	**0**	6	5	6	6	6
6/6	F1	.83	.50	.25	.33	**1.0**	**1.0**	.83	.83	**1.0**	.44	**1.0**	.00	.25	.00	.00	.00
asia	SHD	**0**	5	7	1	**0**	1	3	**0**	**0**	**0**	**0**	**0**	2	16	6	2
8/8	F1	**1.0**	.33	.15	.80	**1.0**	.80	.67	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	.67	.20	.57	.91
sachs	SHD	**0**	9	11	8	12	13	**0**	**0**	13	14	11	**0**	8	16	1	**0**
11/17	F1	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**	**.00**
child	SHD	**0**	18	16	18	20	22	13	9	20	15	13	13	18	23	**0**	**0**
20/25	F1	**1.0**	.24	.43	.33	.12	.12	.00	.74	.12	.47	.59	.57	.34	.25	**1.0**	**1.0**
insurance	SHD	**5**	41	30	34	28	36	34	21	29	34	28	19	36	53	35	14
27/52	F1	**.89**	.26	.44	.42	.44	.39	.00	.66	.55	.46	.56	.76	.36	.05	.51	.82
water	SHD	5	33	43	31	**0**	4	60	7	8	38	27	38	18	61	65	52
32/66	F1	.94	.52	.34	.56	**1.0**	.97	.00	.91	.87	.49	.62	.46	.76	.00	.20	.50
mildew	SHD	6	-	17	25	34	21	-	-	7	**3**	9	23	18	52	36	-
35/46	F1	.87	-	.68	.50	.33	.56	-	-	.85	**.93**	.80	.64	.65	.19	.41	-
alarm	SHD	**1**	21	26	18	20	20	20	6	17	8	3	21	18	46	17	2
37/46	F1	**.98**	.57	.44	.64	.57	.57	.57	.92	.64	.86	.94	.66	.62	.12	.82	.98
barley	SHD	5	48	55	50	**0**	3	-	-	8	42	-	34	50	87	60	42
48/84	F1	.95	.46	.38	.44	**1.0**	.96	-	-	.91	.59	-	.72	.43	.00	.48	.67
hailfinder	SHD	11	47	41	43	**0**	17	-	-	26	60	-	59	44	76	111	118
56/66	F1	.80	.37	.45	.42	**1.0**	.85	-	-	.70	.21	-	.23	.42	.00	.18	.12
hepar2	SHD	**0**	54	81	59	**0**	35	27	37	14	46	40	35	75	123	79	61
70/123	F1	**1.0**	.59	.34	.54	**1.0**	.72	.81	.70	.89	.75	.70	.81	.39	.00	.54	.68
win95pts	SHD	1	65	51	33	**0**	8	42	7	5	32	21	16	50	112	103	-
76/112	F1	.99	.43	.54	.73	**1.0**	.95	.64	.95	.97	.77	.85	.91	.57	.00	.47	-
pathfinder	SHD	25	157	145	151	**0**	150	-	-	147	158	-	168	148	196	241	-
109/195	F1	.77	.21	.29	.21	**1.0**	.29	-	-	.30	.29	-	.28	.31	.00	.07	-
munin1	SHD	**10**	169	154	153	72	86	117	-	84	109	-	233	151	-	257	-
186/273	F1	**.97**	.42	.47	.46	.77	.71	.58	-	.72	.67	-	.26	.50	-	.42	-
andes	SHD	**0**	226	209	246	**0**	4	83	4	5	47	15	38	149	-	175	-
223/338	F1	**1.0**	.35	.41	.29	**1.0**	.99	.75	.99	.98	.92	.96	.92	.60	-	.76	-
diabetes	SHD	25	220	395	237	48	**0**	-	-	204	146	-	592	368	-	534	-
413/602	F1	.96	.62	.38	.62	.96	**1.0**	-	-	.68	.77	-	.03	.43	-	.43	-
pigs	SHD	**0**	350	332	263	400	400	-	-	268	**0**	-	532	316	-	6	-
441/592	F1	**1.0**	.44	.46	.59	.35	.35	-	-	.56	**1.0**	-	.18	.50	-	1.0	-
link	SHD	**0**	731	630	638	749	737	-	-	204	324	-	1047	400	-	947	-
724/1125	F1	**1.0**	.38	.45	.45	.39	.40	-	-	.81	.80	-	.14	.64	-	.49	-
munin	SHD	72	967	790	816	**0**	156	-	-	458	661	-	1397	795	-	1599	-
1041/1397	F1	.95	.36	.48	.44	**1.0**	.89	-	-	.69	.62	-	.00	.51	-	.29	-
munin2	SHD	**118**	554	611	646	1052	898	-	-	536	632	-	1240	753	-	1321	-
1003/1244	F1	**.92**	.60	.56	.55	.19	.30	-	-	.57	.58	-	.01	.49	-	.46	-
munin3	SHD	**113**	616	629	688	1048	860	-	-	544	566	-	1306	819	-	1539	-
1041/1306	F1	**.92**	.58	.57	.54	.25	.37	-	-	.60	.65	-	.00	.46	-	.26	-
munin4	SHD	**126**	696	658	776	1058	876	-	-	649	618	-	1388	812	-	1627	-
1038/1388	F1	**.93**	.54	.56	.50	.29	.39	-	-	.55	.64	-	.00	.49	-	.28	-
rank(SHD)	mean	**1.6**	9.5	8.9	8.3	4.7	6.7	9.4	7.9	4.2	6.1	8.2	8.0	7.9	13.5	10.9	9.5
	±stdd	**1.0**	3.1	3.6	2.3	4.3	3.8	4.3	4.9	3.1	3.7	4.6	3.7	2.7	1.8	3.6	5.0
UT-F1	mean	**.90**	.22	.19	.27	.66	.50	.53	.87	.59	.54	.77	.47	.30	.09	.36	.70
	±stdd	**.13**	.17	.13	.18	.40	.34	.33	.16	.32	.28	.24	.35	.22	.07	.29	.33

633 domains use the default configuration. The result SHD and F1-score are reported as mean value and standard
634 deviation over the five test graphs.

635 D Code and Data

636 D.1 URLs of all competitors

637 We use open-source codes of other algorithms for evaluation.

638 For Jarfo, RCC, NCC, GES, GS(Grow-Shrink), and CDS, we use the API provided by Causal Discovery
639 Toolbox [18]: `https://github.com/FenTechSolutions/CausalDiscoveryToolbox`.

640 For HC(Hill-Climbing) we use pgmpy `https://github.com/pgmpy/pgmpy` with BDeu score.

641 For PC we use the official R package pcalg `https://cran.r-project.org/web/packages/pcalg`.

For Conservative-PC and Majority-rule PC, we slightly modify the source code of pcalg to enable a faster run on large scale datasets. GLL-MB is also implemented based on pcalg. Reviewers can download our modified implementation of these 3 algorithms from `http://ml4c.xyz`.

D.2 Algorithms starting from data: DAG-GNN/BLIP/GOBNILP

D.2.1 Code URL

1. GOBNILP: `https://bitbucket.org/jamescussens/pygobnilp/`.
2. BLIP: `https://cran.r-project.org/web/packages/r.blip/`.
3. DAG-GNN: We use a repository with a standard and clean version of the DAG-GNN algorithm, which is well maintained and can be found at `https://github.com/ronikobrosly/DAG_from_GNN/`.

D.2.2 Hyper-parameter settings

1. Time limit: The running time of all programs is limited to 24 hours.
2. Max-in-degree: The max-in-degree threshold for BILP is set to 6. The max-in-degree threshold for GOBNILP is set to 3.
3. Configurations of DAG-GNN are as follows. Epochs=300, batch size=100, learning rate=3e-3, graph threshold=0.3. Graph threshold is a threshold for weighted adjacency matrix (i.e., any weights > -0.3 and < 0.3 means the two variables are not adjacent).

D.2.3 Verifying the results of DAG-GNN

To make sure DAG-GNN is correctly executed, we have carefully experiment DAG-GNN from the following two aspects:

Reproducing the results of paper [38] We take the child dataset as an example to test the reproducibility, because the data set has been reported by [38]. As can be seen from Table 7, the BIC scores are similar to the results reported in the original paper (child: -1.38e+4). That is to say, the results in Yu et al.'s paper are reproduced by us.

Table 7: Reproducing results for child

	groundtruth	child
BIC	-1.23e+4	-1.36e+4

Different graph thresholds The following are the BIC scores on the data sets of alarm and water with different graph thresholds. The graph threshold recommended by [38] is 0.3. It can be seen that the performance is stable when the threshold is around 0.3. We have verified that there are similar conclusions on other data sets. Therefore 0.3 should be a reasonable threshold.

Table 8: Results with different graph thresholds

	Groundtruth	0.1	0.2	0.3	0.4	0.5
alarm	-1.08e+5	-1.90e+5	-1.44e+5	-1.59e+5	-1.77e+5	-1.91e+5
water	-1.35e+5	-1.32e+5	-1.37e+5	-1.44e+5	-1.53e+5	-1.62e+5

D.3 ML4C: Code and data

For reviewers to check reproducibility of our results reported in §5, we put our code and data on an anonymous site `http://ml4c.xyz`.