Reinforcement Causal Structure Learning on Order Graph

Dezhi Yang[1,2], **Guoxian Yu**[1,2], **Jun Wang**[2,*], **Zhengtian Wu**[3], **Maozu Guo**[4]

[1]School of Software, Shandong University, Jinan, China
[2]SDU-NTU Joint Centre for AI Research, Shandong University, Jinan, China
[3]School of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou, China
[4]College of Elec. & Inf. Eng., Beijing University of Civil Engineering and Architecture, Beijing, China
dzyang@mail.sdu.edu.cn, {gxyu, kingjun}@sdu.edu.cn, wzht8@mail.usts.edu.cn, guomaozu@bucea.edu.cn

Abstract

Learning directed acyclic graph (DAG) that describes the causality of observed data is a very challenging but important task. Due to the limited quantity and quality of observed data, and non-identifiability of causal graph, it is almost impossible to infer a single precise DAG. Some methods approximate the posterior distribution of DAGs to explore the DAG space via Markov chain Monte Carlo (MCMC), but the DAG space is over the nature of super-exponential growth, accurately characterizing the whole distribution over DAGs is very intractable. In this paper, we propose Reinforcement Causal Structure Learning on Order Graph (RCL-OG) that uses order graph instead of MCMC to model different DAG topological orderings and to reduce the problem size. RCL-OG first defines reinforcement learning with a new reward mechanism to approximate the posterior distribution of orderings in an efficacy way, and uses deep Q-learning to update and transfer rewards between nodes. Next, it obtains the probability transition model of nodes on order graph, and computes the posterior probability of different orderings. In this way, we can sample on this model to obtain the ordering with high probability. Experiments on synthetic and benchmark datasets show that RCL-OG provides accurate posterior probability approximation and achieves better results than competitive causal discovery algorithms.

Introduction

Causal discovery can reveal the causal mechanisms underlying natural phenomena, it has been attracting the attention of many disciplines (Pearl 2009). The golden standard for causal discovery is to conduct randomized intervention experiments, which is however restricted by various factors (i.e., cost and ethics) and even infeasible. Therefore, inferring causal structure from passively observable data is more attractive and the only road for causal discovery (Pearl 2009; Peters, Janzing, and Schölkopf 2017).

The causal graph captures the underlying causal mechanism of the data, where nodes and edges correspond to variables and causality between variables, respectively. The causal graph typically is a directed acyclic graph (DAG). To find causal graphs, score-based methods assign a score $S(\mathcal{G})$ to each DAG $\mathcal{G}$ and search the DAG with the best score in the DAG space:

$$\min_{\mathcal{G}} S(\mathcal{G}),\ \text{s.t. } \mathcal{G} \in DAGs \tag{1}$$

However, due to the combinatorial nature of acyclic constraints, the search complexity is generally considered to be NP-hard (Chickering, Heckerman, and Meek 2004). Most methods employ heuristic strategies. To name a few, GES (Chickering 2002) greedily adds edges that increase the score of DAG the most; and max-min hill climbing (Tsamardinos, Brown, and Aliferis 2006) searches DAGs in the space reduced by constraint-based methods. Some other methods align learning an optimal DAG with learning an optimal DAG topological ordering, and search causal structures in a smaller ordering space (Teyssier and Koller 2005; Raskutti and Uhler 2018; Bernstein et al. 2020).

Most score-based methods can only provide a single DAG or its Markov equivalent class with the best score, which may produce poor results, especially in the case of limited data, poor data quality, and unidentifiable causal graphs. Different from score-based methods (Tsamardinos, Brown, and Aliferis 2006; Huang et al. 2018), MCMC-based methods (Madigan, York, and Allard 1995; Niinimäki, Parviainen, and Koivisto 2016; Deleu et al. 2022) infer the posterior probability of the DAG structure $P(\mathcal{G}|\mathcal{D})$ from the observation dataset $\mathcal{D}$ to account for the epistemic uncertainty over graph structures. However, the too huge DAG combination space makes it very difficult to characterize the DAG posterior distribution via MCMC and even produces slow mixing.

In this paper, we propose a deep reinforcement learning based solution (RCL-OG) that approximates the posterior probability of DAG topological orderings using order graph. The order graph can clearly capture the generation process of each kind of ordering, and sampling the order can be regarded as a Markov decision process, starting from the empty set, by selecting one variable to join the set based on the ordered variables. By designing new transition rewards for the nodes of the order graph, we give the formula for calculating the probability of transition between nodes and prove the detailed-balance condition. We show that computing the transfer rewards is a dynamic programming problem, so we use the Nature DQN (Mnih et al. 2015) strategy with a new target function to approximate the transfer rewards, as outlined in Figure 1.

*Corresponding author.

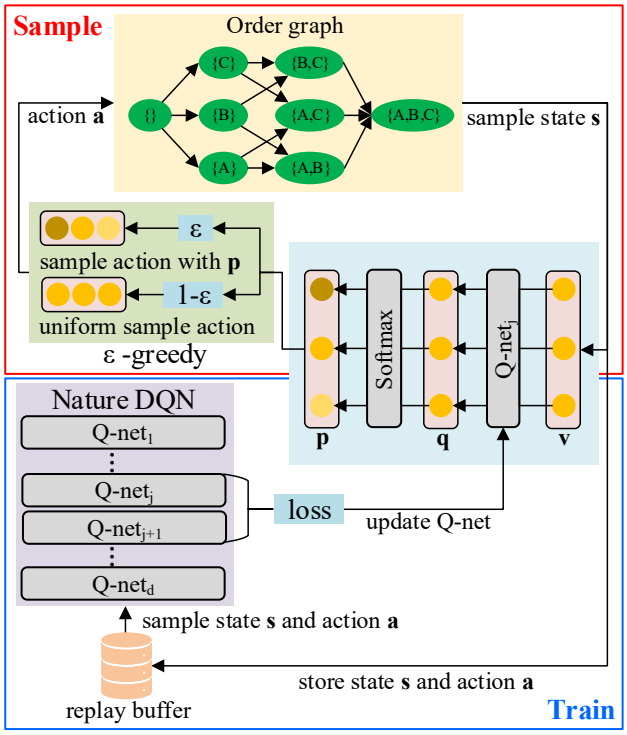


Figure 1: Schematic framework of RCL-OG. In the sampling phase, RCL-OG computes the transition probabilities $\mathbf{p}$ by Q-nets, uses the ϵ-greedy policy to sample states and actions, and stores them in the replay buffer. During the training phase, it draws a batch of samples from the replay buffer to update Q-nets via Nature DQN strategy. The trained Q-nets will be used as the probability transition model to encode possible ordering space.

The main contributions of our work are listed as follows:
(i) We introduce an order graph with a much smaller search space to improve the efficacy of causal structure search, and define states, actions and rewards for reinforcement learning based search on this graph.
(ii) We introduce neural networks to approximate the transition probability on the order graph, and reformulates ordering search and probability model training into the reinforcement learning framework to leverage their advantages for learning the posterior probability distribution of DAG topological ordering.
(iii) Experiments on simulated datasets with known transition probability confirm the reliability of RCL-OG on inferring the transition probability. By comparing with state-of-the-art methods on simulated and real-word datasets, we prove that RCL-OG can sample on learned probability model to accurately find causal structures.

Related Work

Score-based methods search for DAGs using various well-defined score functions (i.e. Bayesian Information Criterion (BIC) (Schwarz 1978) and Minimum Description Length (MDL) (Chickering 2002)). But the combinatorial nature of search makes it difficult for these methods to explore large graph structures. Some methods attempt the continuous optimization to learn causal structure from data. Zheng et al. (2018) formulated the combinatorial optimization problem as a continuous optimization problem by introducing a smooth characterization for the acyclicity. Gran-DAG (Lachapelle et al. 2019) and NOTEARS-MLP (Zheng et al. 2020) extend the smooth acyclic constraints to neural networks for nonlinear data. Some other methods (i.e. CGNN (Goudet et al. 2018) and SAM (Kalainathan et al. 2018)), also use neural networks for causal discovery, but they do not guarantee acyclicity.

Most causal structure learning methods return a single DAG, they are limited by observable data and often fail to produce good results. Markov chain Monte Carlo (MCMC)-based methods approximate a whole posterior distribution over DAGs by exploring the DAG space. In essence, MCMC is a sampler that simulates a Markov chain in the DAG space by local transitions (e.g. adding or removing edges) (Madigan, York, and Allard 1995). Friedman and Koller (2003) proposed an MCMC sampler in the space of node orders to reduce the search space and avoid slow mixing, but also introduced a bias. Kuipers and Moffa (2017) further refined these methods by modifying the underlying space or local transition of Markov chains. DAG-GFlowNet (Deleu et al. 2022) used generative flow network instead of MCMC and approximated posterior of DAGs with deep learning. A smaller and well-defined search space can help MCMC-based methods better sample possible structure. We propose to replace MCMC with order graphs to reduce the underlying search space, and model the transition between orderings as a Markov decision process to improve the search performance of RCL-OG.

RL has also made good progress in causal structure learning (Khalil et al. 2017). Bello et al. (2016) used the simple reward mechanisms of RL to solve combinatorial optimization problems. RL-BIC (Zhu, Ng, and Chen 2019) uses deep-RL (Sutton and Barto 2018) to search for a single high score structure, but it neglects the decision process. Exploring the search space is essential for causal structure learning, but we typically can only explore a small part of it, due to the high search complexity and huge space. In this work, we use neural networks to approximate the posterior probability of DAG topological ordering, and reformulate the exploration and learning into the RL framework to effectively leverage the searched structure knowledge and infer the posterior distribution of the search space.

Background

Structure Learning

Given a causal graph $\mathcal{G}$ with d random variables $\mathcal{X} = \{X_1, \cdots, X_d\}$, whose joint distribution can be decomposed according to $\mathcal{G}$:

$$P(X_1, \cdots, X_d) = \prod_{k=1}^{d} P(X_k | \mathrm{Pa}(X_k)) \qquad (2)$$

where $\mathrm{Pa}(X_k)$ is the set of parents of X_k in $\mathcal{G}$. We assume that the observed dataset $\mathcal{D}$ is generated by Struc-

tural Equation Model (SEM) with additive noises: $x_i = f_i(Pa(x_i))+\epsilon_i, i = 1, 2, \cdots, d$. We consider the case where the model is fully identifiable and also assume the causal minimality, namely each function f_i is not a constant (Peters et al. 2014).

The problem of finding a DAG can be cast as finding a variable ordering (Teyssier and Koller 2005). We can easily represent the ordering as a fully connected DAG by setting frontal nodes in the order as parents of subsequent nodes. Each DAG can be consistent with more than one orderings, so searching the real causal graph $\mathcal{G}^*$ can be regarded as searching an ordering, whose fully connected DAG is a super-graph of $\mathcal{G}^*$.

Given a dataset $\mathcal{D}$, we can infer the posterior distribution of DAGs over data. A natural idea is that the probability of a DAG is proportional to its score. If the score function can be decomposed, we have the following:

$$P(\mathcal{G}|\mathcal{D}) \propto R(\mathcal{G}) = \prod_{k=1}^{d} S(X_k, \mathrm{Pa}(X_k)|\mathcal{D}) \tag{3}$$

where $S(\cdot)$ is the decomposed partial score function. We can also apply the ordering to the above formula: $P(\mathcal{L}|\mathcal{D}) \propto \prod_{k=1}^{d} S(X_k, \mathrm{U}(X_k)|\mathcal{D})$, where $\mathcal{L}$ stands for an ordering and $\mathrm{U}(X_k)$ is the set of variables ahead X_k in $\mathcal{L}$.

Order Graph

The order graph is introduced to describe the generation process of variable orderings, and its structure is exampled in the top of Figure 1, where nodes represent the set of sorted variables and edges means adding a variable to the set of its head nodes. We define nodes on the order graph as states $s \in \mathcal{S}$ and edges as actions $a \in \mathcal{A}$. Different paths from the top empty set node s_0 to the bottom universal set node s_d capture different variable orderings. Therefore, determining a path is a Markov decision process.

The Proposed Methodology

In this section, we first introduce how to model the whole variables ordering space with the order graph. We design new definitions of states, rewards and actions for the order graph, provide the formula of transition probability between states and prove the detailed-balance condition. Then we discuss how to use RL to estimate the transition probability and obtain the posterior distribution of DAG topological orderings.

Modeling Variable Orderings

States and rewards: The order graph is a DAG, in which there are only forward transitions (add variables to sorted set) and no backward transitions. To enable the order graph to model transitions between complete orderings, we modify its node s meaning to be the set of orderings corresponding to paths through s (not all paths, as we will explain later). Take the order graph in Figure 1 for example, node $s = \{A\}$ denotes the set of orderings $O(s) = \{[A, B, C], [A, C, B]\}$. Consider the transition from state s to its sub-state s', since s' is generated by s, the set of orderings represented by s' only includes the orderings corresponding to paths that pass through both s and s', namely $O(s') \subset O(s)$. In addition, we assume that there exists a definite path l from state s_0 to s, so that the size of ordering sets represented by s and s' is further reduced, because the path corresponding to the orderings in them must go through l. Under these assumptions, we define the reward of state s as the sum of the scores of all orderings it represents: $R(s) = \sum_{\mathcal{L}\in O(s)} R(\mathcal{L})$, where $R(\mathcal{L})$ is the score of ordering $\mathcal{L}$.

Detailed-balance condition: After encoding states as ordering sets, the probability of state s is given by:

$$P(s) = \sum\nolimits_{\mathcal{L}\in O(s)} P(\mathcal{L}) \tag{4}$$

where $O(s)$ is the set of orderings encoded by s and $P(\mathcal{L})$ is the probability of ordering $\mathcal{L}$. Under the assumption that there exists a definite path from state s_0 to s, since $O(s') \subset O(s)$, we get the following transition probability:

$$P(s'|s) = \frac{P(s's)}{P(s)} = \frac{P(s')}{P(s)} = \frac{\sum_{\mathcal{L}'\in O(s')} P(\mathcal{L}')}{\sum_{\mathcal{L}\in O(s)} P(\mathcal{L})} \tag{5}$$

Notice that $P(\mathcal{L}|\mathcal{D}) \propto R(\mathcal{L})$, we have $P(s'|s) = R(s')/R(s)$ and find the transition of states on the order graph satisfying the detailed-balance condition:

$$P(s'|s)R(s) = P(s|s')R(s') \tag{6}$$

where $P(s|s') = 1$, because s' is produced by s.

Actions and values: The calculation of state rewards $R(s)$ may be difficult because of the high complexity of ordering rewards $R(\mathcal{L})$. As such, we need to simplify the formula of transition probability. In the order graph, there is a unique action between each pair of parent and child states, which means putting a variable into the sorted set to make state s transiting to s'. We define the reward of action a as decomposed partial score function:

$$R(a) = S(s' - s, s|\mathcal{D}) \tag{7}$$

where $s' - s$ is the difference between two sets with sorted variables. We can express the ordering reward as the product of action rewards and then compute the sum of ordering rewards to obtain the rewards for states. We take the action reward into the detailed-balance condition and get the following formulas:

$$\begin{aligned} P(s'|s) &= \frac{Q(s,a)}{\sum_{a'\in\mathcal{A}_s} Q(s,a')} \\ Q(s,a) &= R(a) \sum_{a'\in\mathcal{A}_{s'}} Q(s',a'),\ Q(s_{d-1},a) = R(a) \end{aligned} \tag{8}$$

where a is the action between s and s', $\mathcal{A}_s$ is the set of all possible actions of s, and s_{d-1} stands for states in the second-to-last layer of the order graph with only one action. The above formulas consider that the transition probability from state s to s' is independent of the paths (or actions) before s and can be obtained by recursively calculating $Q(s, a)$. We call $Q(s, a)$ the value of doing action a in state s. By computing $Q(s, a)$ for all states and their actions, we can obtain the transition probabilities between each pair of states.

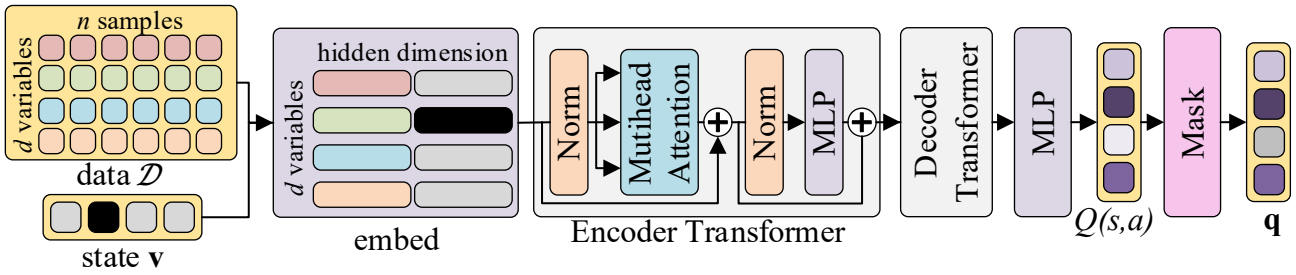


Figure 2: The architecture of Q-net. The input data is a data matrix of n samples represented by d variables. The input state is a binary vector of length d. Each variable is embedded through the concatenation of the embeddings of its corresponding sample values and the embeddings of its corresponding entry in the state vector. These embeddings are fed into a Transformer and encoded as features. The decoder has the same structure as the encoder, along with a multi-layer perception to decode the features into $Q(s, a)$ of the state s for each possible action a. Finally, $Q(s, a)$ is masked as $\mathbf{q}$ to bypass explored actions.

Learning Probability Transition Model

Network structure: Although the order graph greatly reduces the size of the state-action space compared to MCMC, it is still difficult to traverse all states and actions to compute the transition probability between each pair of states. RL simplifies the search process based on Markov decision and guides us to explore important parts of the state-action space on the order graph based on rewards and defined target function (which usually prefers higher rewards for states and actions). In the RL framework, we use neural networks to parameterize $\log Q(s, a)$ (we will explain why $\log$ later) and to borrow their capacity for generalizing to states and actions not explored by RL. In practice, we adopt an encoder-decoder structure, as shown in Figure 2.

The state is expressed as a binary vector $\mathbf{v} \in \{0|1\}^d$, whose entry is 1 if the corresponding variable is included in the set of sorted variables represented by the state, and 0 otherwise. We embed data $\mathcal{D}$ with the state vector $\mathbf{v}$ and map the embedding into a feature vector through a Transformer (Vaswani et al. 2017). We empirically find that the input $\mathcal{D}$ can improve the model performance. Next, we use another transformer to decode the feature vector into a value vector:

$$\mathbf{q} = \text{Transformer}_{\text{de}}(\text{Transformer}_{\text{en}}(\text{embed}(\mathcal{D}, \mathbf{v}))) \quad (9)$$

Each entry of $\mathbf{q}$ denotes $\log Q(s, a)$ of the corresponding action. But for a state s, not all actions are feasible. For example, it is not possible for $s = \{A\}$ to add variable A to the set again, so we mask impossible actions of $\mathbf{q}$ as $-9e15$. We call the entire network as *structure Q-net*.

Reinforcement learning based ordering search: As shown in Figure 1, starting from the initial state s_0 of the order graph, we first input the binary vector $\mathbf{v}_0$ of s_0 into Q-net to obtain the value vector $\mathbf{q}$. We perform softmax to process $\mathbf{q}$ into a probability vector $\mathbf{p}$, and then use the ϵ-greedy strategy to select the action. Specifically, we sample possible action by $\mathbf{p}$ with probability ϵ, and uniformly sample action with probability $1 - \epsilon$. We then perform the sampled action on the order graph to get the next state, and repeat the above process. When reaching the state s_d, we start the above sampling process from s_0 again until a certain number of iterations. We store the sampled state s, action a and next state s' into a replay buffer. At each iteration, we randomly draw a batch of samples from the buffer and compute their current and target value to update Q-net.

According to Eq. (8), we need to calculate action rewards to express $Q(s, a)$. We can easily compute the logarithm of the action reward by score function:

$$\log R(a) = \text{LocalScore}(s' - s, s|\mathcal{D}) \quad (10)$$

where the score function is Bayesian Information Criterion (BIC), which is not only consistent but also decomposable (Chickering, Heckerman, and Meek 2004). Since the score function computes the logarithm of reward $R(a)$, we rewrite the target function of Q-net as the logarithm of $Q(s, a)$ for ease calculation:

$$y(s,a) = \begin{cases} \log R(a) & s' = s_d \\ \log R(a) + & \\ \quad \log(\sum_{a' \in \mathcal{A}_{s'}} \exp(Q(s', a'))) & s' \neq s_d \end{cases} \quad (11)$$

For ease understanding, we still use $Q(s, a)$ to denote the output of Q-net. We use the mean square error between current output value and the calculated target value as the loss: $loss = \frac{1}{m}\sum_{j=1}^{m}(y(s,a) - Q(s,a))^2$, where m is the size of batched samples drawn from the replay buffer.

Inspired by Nature DQN (Mnih et al. 2015), we use different Q-nets to parameterize $Q(s, a)$ in different layers of the order graph. For an order graph with d variables, it has $d + 1$ layers, indexed from 1 to $d + 1$. We use the same architecture to initialize d Q-nets corresponding to layer 1 to d of the order graph, respectively. As shown in Figure 1, the state-action values of state s at j-th layer and its sub-state s' at $(j+1)$-th layer are respectively parameterized by the j-th Q-net and the $(j + 1)$-th Q-net to calculate $loss$, which not only decouples the current value and target value, but also reduces the number of state-action values that Q-net need to parameterize.

After Q-nets are trained, we can obtain the transition probability model and variable orderings by sampling on this model. We transform orderings into fully connected DAGs, get the weight of each edge on them by simple linear regression, and prune them by a threshold to get final DAGs. Algorithm 1 summarizes the whole procedure of RCL-OG.

Algorithm 1 RCL-OG: Reinforcement Causal Structure Learning on Order Graph

Input: Observed data $\mathbf{X} \in \mathbb{R}^{n\times d}$, ϵ-greedy parameter ϵ, number of iterations τ, number of variables m and batch size z

Initialization: initialize Q-nets[1:m], Initial state s_0 and replay buffer $\mathbb{B}$

```
Set state s = s_0 and index l = 0
for t = 1 : τ do
    Get q-value q = Q-nets[l](s, X);
    if random sample e ∈ [0, 1] and e < ε then
        Sample action a with q = softmax(q);
    else
        Uniformly sample action a;
    end if
    Get next state s' and store it in B along with s and a;
    if B.length > z then
        Sample z states and actions from B and update Q-nets[l] by Nature DQN strategy;
    end if
    if mod(t, m) = 0 then
        Set l = 0 and s = s_0
    else
        Set l = l + 1 and s = s'
    end if
end for
```

Results and Analysis

Comparison with Exact Transition Probability

Since the posterior probability is the product of transition probabilities, we compare the transition probability calculated by the networks against the exact ones to prove that RCL-OG can provide accurate posteriors of DAG topological orderings. However, computing the exact transition probability between states on the order graph still requires to exhaustively enumerate all possible orderings and compute their scores, which is only feasible for graphs with a few variables. The code of RCL-OG is shared at www.sduidea.cn/codes.php?name=RCL-OG. We use Python 3.9 with the MindSpore deep learning framework to implement RCL-OG and perform experiment on a DAG generated with 5 nodes and 2×5 edges under the Erdös-Rényi model (Erdos, Rényi et al. 1960). We generate 200 samples based on linear model with Gaussian noise. By enumerating all possible orderings of these variables and computing their scores, we can obtain the exact transition probability.

For state s and its sub-state s' on the order graph, we no longer assume that there exists a definite path from s to s'. So the reward for s is the sum of scores for all paths that pass through s, and this for s' is the sum of scores for all paths that go through both s and s'. The exact transition probability from s to s' is:

$$P(s'|s) = \frac{\sum_{\mathcal{L}'\in O_{all}(s')} R(\mathcal{L}')}{\sum_{\mathcal{L}\in O_{all}(s)} R(\mathcal{L})} \tag{12}$$

where $O_{all}(s')/O_{all}(s)$ include all orderings that go through state s'/s. We input the state vector of s to Q-nets and execute softmax on the value vector outputted by networks to obtain the probability vector, then the value of s' on the probability vector is the estimated transition probability from s to s'.

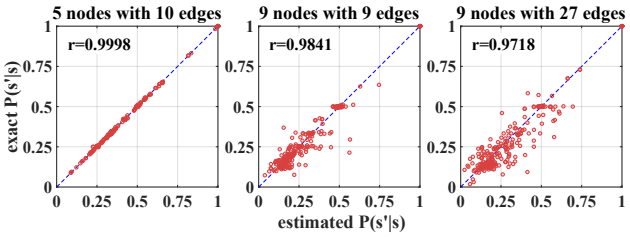


Figure 3: Distribution of exact transition probabilities (y-axis) and the ones estimated by RCL-OG (x-axis). We perform experiments with 5 different DAGs, randomly select 50 pairs of states and sub-states for each DAG. r is the Pearson correlation coefficient.

RCL-OG can accurately learn the transition probability between states on order graph, and give precise approximation of the posterior distribution over DAG topological orderings $P(\mathcal{L}|\mathcal{D})$. For validation, we randomly select 250 pairs of states and sub-states from the order graph of the experimental DAGs, plot the exact probabilities and the ones estimated by RCL-OG in Figure 3. We see the exact and estimated probabilities are strong linearly correlated.

We also repeat the above experiments on the DAGs with d=9 nodes and $1d, 3d$ edges, and reveal results in Figure 3. RCL-OG does not perfectly estimate all transition probabilities as $d = 9$, which we believe is caused by the size of states and actions space. With the increase of variables, the space of states and actions in the order graph super-exponentially increases, and it is impossible to search all possible states and actions in the sampling phase. Although RCL-OG cannot accurately approximate the transition probabilities that are infrequent or not searched, experimental results at multi-nodes graph show that this deficiency does not prohibit it from discovering the exact causal structure. For a state s in the order graph, the reward gap between different actions enlarges with the increase of variables, and the correct action tend to have a higher value than other actions, which also make the transition probability of the correct action much higher than other actions. In the sampling phase, the correct action is more likely to be sampled and used to train the networks, so RCL-OG is still able to find the exact causal structure in the multi-nodes graph.

Experiments on Simulated Data

We perform experiments on graph with 20 nodes, and sample the ground-truth graphs following an Erdös-Rényi model, with $2d$ edges in expectation (a setting referred as ER2). MCMC-based oMCMC (Friedman and Koller 2003), and traditional causal discovery methods, two variants of Bootstrapping (Lorch et al. 2021) based on the score-based algorithm GES (B-GES, (Chickering 2002)), and the constraint-based algorithm PC (B-PC, (Spirtes et al. 2000))

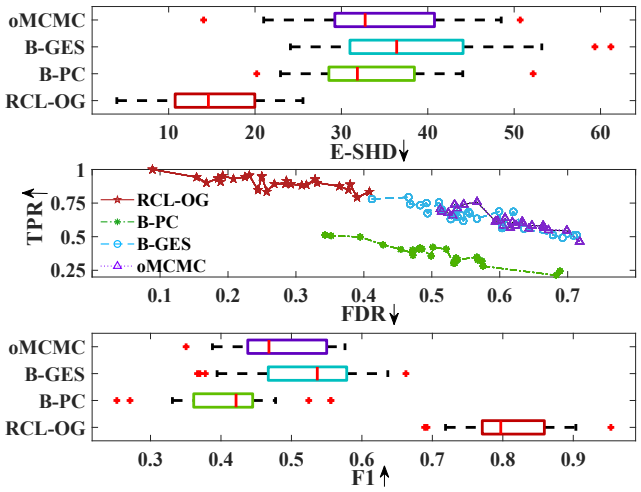


Figure 4: Results on DAGs with $d = 20$ nodes. ↑/↓ means the preferred direction of the metric value.

are used as baselines. We conduct experiments on 25 randomly generated different DAGs, and in each experiment, we generate a dataset of observations with 200 samples under the linear Gaussian model and sample 1000 DAGs from each method. We take the best 20 DAGs as the output of each method, and quantify the performance using the expected structural Hamming distance (E-SHD), which is approximately defined as the mean of SHD between the DAGs sampled by each method and the true graph:

$$\text{E-SHD} \approx \frac{1}{n}\sum_{k=1}^{n} \text{SHD}(\mathcal{G}_k, \mathcal{G}^*) \tag{13}$$

where n is the number of sampled graphs $\mathcal{G}_k$, $\mathcal{G}^*$ is the true graph and $\text{SHD}(\mathcal{G}_k, \mathcal{G}^*)$ is the structural Hamming distance between $\mathcal{G}_k$ and $\mathcal{G}^*$. We also report the line-plot of True Positive Rate (TPR) and False Discovery Rate (FDR) and the box-plot of F1-Score in Figure 4.

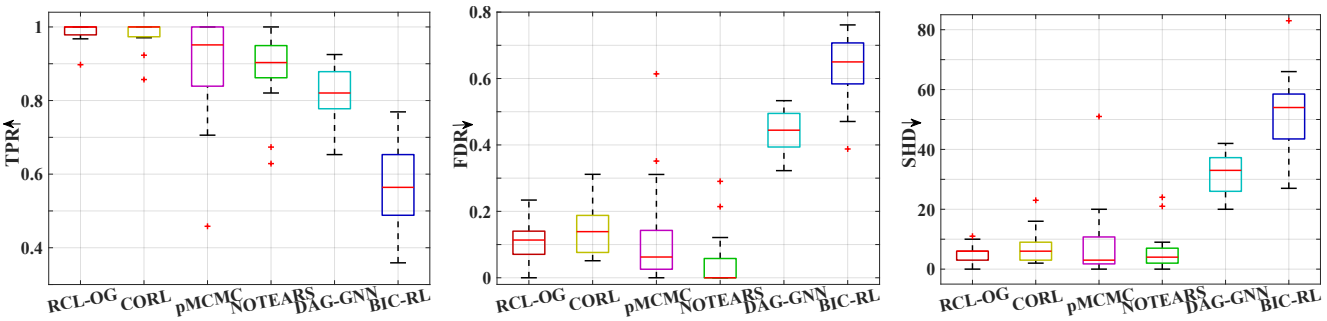
RCL-OG can more credibly sample the correct causal structure than other methods. The performance of B-PC and B-GES is restricted by their base algorithms PC and GES, so they cannot sample the exact graph structure. Compared with oMCMC, the state-action space of RCL-OG on order graph is much smaller. Therefore, with the trained transition model, RCL-OG is more likely to sample the correct order without expensive sampling cost. These experiments prove that RCL-OG can produce posterior probabilities close to those of topological orderings of ground-truth graph $\mathcal{G}^*$.

Besides sampling possible orderings in the order graph space, RCL-OG can also get an exact single ordering. Starting from the initial state s_0 of the order graph, we can obtain a single ordering by selecting action with the maximum transition probability at each transition until the target state s_d. We process this ordering into a DAG in the same way as before and call it the final output of RCL-OG. We compare RCL-OG against with state-of-the-art causal discovery methods, each of which also outputs a single DAG. The baselines include NOTEARS (Zheng et al. 2018), DAG-GNN (Yu et al. 2019), BIC-RL (Zhu, Ng, and Chen 2019) and CORL (Wang et al. 2021). We also include an MCMC-based baseline pMCMC (Kuipers and Moffa 2017). We use the same setup as the sampling experiment and report the TPR, FDR and SHD of each method in Figure 5. We have the following observations:

(i) RCL-OG clearly outperforms most compared methods across different evaluation criteria. BIC-RL has the worst performance in each metric, because it needs enough iterations to hit the correct causal structure in the whole DAGs space. DAG-GNN also performs poorly, because it uses graph neural network to reconstruct data, which requires a large amount of data for training. As a result, DAG-GNN maybe not able to learn a good causal structure with limited samples. NOTEARS has a better performance than BIC-RL and DAG-GNN, since the adopted linear regression favors the loss function of NOTEARS. pMCMC can also get the best results, but the nature of searching through probability-based sampling makes its results unstable. RCL-OG and CORL yield a similar performance, and achieve the best results on TPR, but slightly lose to NOTEARS on FDR. This is because they both prune the fully connected DAG induced from the ordering, but pruning often cannot remove all wrong edges. Even so, the results still prove that RCL-OG and CORL can find the correct topological ordering of causal graph.

(ii) RCL-OG has the highest TPR with ensured stability. By taking three metrics together, we can see that RCL-OG is slightly better than CORL in accuracy and stability. CORL searches the whole orderings space by RL and outputs the searched optimal ordering. In contrast, RCL-OG only needs to sample actions on the transition model with the maximum probability to get better (or similar) results. The smaller search space and the deterministic sampling makes RCL-OG more stable than CORL.

(iii) RCL-OG not only can find accurate causal topological orderings on a larger graph, but also has a higher search efficiency. We further compare RCL-OG against with CORL and pMCMC on larger graphs with d=40 nodes. With the same experimental settings as before, we record the total number of searched orderings of each method in the training phase and in the sampling phase for attaining the similar TPR. We fix the maximum number of searches as 10^5 to prevent them from indefinite search and record the counts in Table 1. We find the results of pMCMC are less stable, and it occasionally fails to find an accurate DAG using the maximum number of searched orderings. We record its average number of searches when its TPR≈0.97. For the case its TPR cannot reach 0.97 within the maximum number of searches, we record its best TPR. We set TPR≈0.97, because it is a common and good enough result for the three methods under the maximum number of searches. We see that both TPR and the number of used searches of pMCMC are worse than those of RCL-OG and CORL. The lower FPR of pMCMC is due to its detailed neighborhood selection, while RCL-OG and CORL simply use linear pruning. CORL can basically find a good enough ordering within the maximum number of searches, and it needs about 7.4×10^4 searches, while RCL-OG uses about 3.6×10^4 searches. That is be-



(a) Results of TPR (b) Results of FDR (c) Results of SHD

Figure 5: Results on DAGs with $d = 20$ nodes.

	RCL-OG	CORL	pMCMC
TPR↑	0.978	0.971	0.964
FPR↓	0.049	0.055	0.016
#search↓	3.6×10^4	7.4×10^4	8.6×10^4

Table 1: Results on DAGs with d=40 nodes. #search is the number of searched orderings for attaching the target TPR≈0.97 in the training and sampling phase.

cause RCL-OG can explore a smaller state-action space on the order graph than CORL and pMCMC.

Experiments on Real-world Data

We evaluated RCL-OG on real-world flow cytometry data (Sachs et al. 2005) to learn a protein signaling causal network based on expression levels of proteins and phospholipids. This dataset is a benchmark for graphical model and widely used in the biological community. It contains continuous measurements of 11 phosphoproteins in individual T-cells. We select the observational data with 853 samples to infer causal structure, and take the DAG given by (Sachs et al. 2005) as the ground-truth graph, which contains 11 nodes and 17 edges.

We again use BIC as the score function, but due to the nonlinear nature of real data, we apply CAM pruning (Bühlmann, Peters, and Ernest 2014) to obtain sparse graph structure. We adopt the same setup as the simulation experiments to evaluate the sampling performance of each algorithm and the capability of finding a single causal structure, respectively. We use more evaluation metrics on the found graph: number of total edges, number of correct edges, SHD and Structural Intervention Distance (SID) (Peters and Bühlmann 2015). For the sampling performance, we report the average of these metrics. The experiment results are given in Table 2.

The sampling performance gaps between compared methods on real data are similar to those on simulated data, and RCL-OG shows obvious advantages over other sampling methods. As to the results of causal structure learning, CORL and pMCMC have a poor performance, since they only search the high score structure, which may not be able to find the correct causal graph in real data. DAG-GNN gets a lower SID than CORL and pMCMC, but it outputs cycles. The results of NOTEARS and BIC-RL are similar, but NOTEARS has a lower SID, which indicates that NOTEARS finds a graph closer to the causal graph in terms of causal mechanisms. RCL-OG holds a better (or comparable) performance than any other compared method, and offers a good trade-off between more correct edges and total edges. In addition, the lowest SID indicates that RCL-OG can identify a more accurate causal mechanism.

	E-total	E-correct↑	E-SHD↓	E-SID↓
oMCMC	7.00	2.70	16.85	55.75
B-GES	12.55	3.95	16.85	59.05
B-PC	11.95	4.95	13.10	59.10
RCL-OG	**10.30**	**6.60**	**11.70**	**41.40**
	total	correct↑	SHD↓	SID↓
pMCMC	7	3	14	57
CORL	10	5	13	50
DAG-GNN	12	6	13	48
BIC-RL	14	7	11	58
NOTEARS	15	7	11	44
RCL-OG	10	7	11	**39**

Table 2: Results of probability sampling (top) and causal structure learning (bottom) on the real dataset.

Conclusions

We study how to accurately characterize the whole distribution over DAGs for effective causal structure learning using limited data. We proposed a new reinforcement learning based solution on variable order graphs with reduced spaces to effectively approximate the posterior distribution of orderings. Experiments on simulated and real datasets prove that RCL-OG not only effectively approximates the posterior distribution of ground truths, but also stably and accurately finds the single causal structure.

Acknowledgements

This work is supported by NSFC (No. 62031003, 62072380 and 62272276), National Key Research and Development Program of China (No. 2022YFC3502101), and CAAI-

Huawei MindSpore Open Fund.

References

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Bernstein, D.; Saeed, B.; Squires, C.; and Uhler, C. 2020. Ordering-based causal structure learning in the presence of latent variables. In *AISTATS*, 4098–4108.

Bühlmann, P.; Peters, J.; and Ernest, J. 2014. CAM: Causal additive models, high-dimensional order search and penalized regression. *Ann. Stat.*, 42(6): 2526–2556.

Chickering, D. M. 2002. Optimal structure identification with greedy search. *JMLR*, 3(11): 507–554.

Chickering, M.; Heckerman, D.; and Meek, C. 2004. Large-sample learning of Bayesian networks is NP-hard. *JMLR*, 5(10): 1287–1330.

Deleu, T.; Góis, A.; Emezue, C. C.; Rankawat, M.; Lacoste-Julien, S.; Bauer, S.; and Bengio, Y. 2022. Bayesian Structure Learning with Generative Flow Networks. In *UAI*.

Erdos, P.; Rényi, A.; et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1): 17–60.

Friedman, N.; and Koller, D. 2003. Bayesian approach to structure discovery in Bayesian networks. *Mach. Learn.*, 50(1–2): 95–125.

Goudet, O.; Kalainathan, D.; Caillou, P.; Guyon, I.; Lopez-Paz, D.; and Sebag, M. 2018. Learning functional causal models with generative neural networks. In *Explainable and interpretable models in computer vision and machine learning*, 39–80. Springer.

Huang, B.; Zhang, K.; Lin, Y.; Schölkopf, B.; and Glymour, C. 2018. Generalized score functions for causal discovery. In *KDD*, 1551–1560.

Kalainathan, D.; Goudet, O.; Guyon, I.; Lopez-Paz, D.; and Sebag, M. 2018. Structural agnostic modeling: Adversarial learning of causal graphs. *arXiv preprint arXiv:1803.04929*.

Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; and Song, L. 2017. Learning combinatorial optimization algorithms over graphs. In *NeurIPS*, 6351–6361.

Kuipers, J.; and Moffa, G. 2017. Partition MCMC for inference on acyclic digraphs. *JASA*, 112(517): 282–299.

Lachapelle, S.; Brouillard, P.; Deleu, T.; and Lacoste-Julien, S. 2019. Gradient-Based Neural DAG Learning. In *ICLR*.

Lorch, L.; Rothfuss, J.; Schölkopf, B.; and Krause, A. 2021. Dibs: Differentiable bayesian structure learning. In *NeurIPS*, 24111–24123.

Madigan, D.; York, J.; and Allard, D. 1995. Bayesian graphical models for discrete data. *Inter. Stat. Rev.*, 63(2): 215–232.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Niinimäki, T.; Parviainen, P.; and Koivisto, M. 2016. Structure discovery in Bayesian networks by sampling partial orders. *JMLR*, 17(1): 2002–2048.

Pearl, J. 2009. *Causality*. Cambridge University Press.

Peters, J.; and Bühlmann, P. 2015. Structural intervention distance for evaluating causal graphs. *Neural Comput.*, 27(3): 771–799.

Peters, J.; Janzing, D.; and Schölkopf, B. 2017. *Elements of causal inference: foundations and learning algorithms*. MIT Press.

Peters, J.; Mooij, J. M.; Janzing, D.; and Schölkopf, B. 2014. Causal Discovery with Continuous Additive Noise Models. *JMLR*, 15: 2009–2053.

Raskutti, G.; and Uhler, C. 2018. Learning directed acyclic graph models based on sparsest permutations. *Stat*, 7(1): e183.

Sachs, K.; Perez, O.; Pe'er, D.; Lauffenburger, D. A.; and Nolan, G. P. 2005. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529.

Schwarz, G. 1978. Estimating the dimension of a model. *Ann. Stat.*, 6(2): 461–464.

Spirtes, P.; Glymour, C. N.; Scheines, R.; and Heckerman, D. 2000. *Causation, prediction, and search*. MIT Press.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT Press.

Teyssier, M.; and Koller, D. 2005. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *UAI*, 584–590.

Tsamardinos, I.; Brown, L. E.; and Aliferis, C. F. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, 65(1): 31–78.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*, 6000–6010.

Wang, X.; Du, Y.; Zhu, S.; Ke, L.; Chen, Z.; Hao, J.; and Wang, J. 2021. Ordering-Based Causal Discovery with Reinforcement Learning. In *IJCAI*, 3566–3573.

Yu, Y.; Chen, J.; Gao, T.; and Yu, M. 2019. DAG-GNN: DAG structure learning with graph neural networks. In *ICML*, 7154–7163.

Zheng, X.; Aragam, B.; Ravikumar, P. K.; and Xing, E. P. 2018. DAGs with no tears: Continuous optimization for structure learning. In *NeurIPS*, 9492–9503.

Zheng, X.; Dan, C.; Aragam, B.; Ravikumar, P.; and Xing, E. 2020. Learning sparse nonparametric dags. In *AISTATS*, 3414–3425.

Zhu, S.; Ng, I.; and Chen, Z. 2019. Causal Discovery with Reinforcement Learning. In *ICLR*.