MARLIN: Multi-Agent Reinforcement Learning for Incremental DAG Discovery

Dong Li[1], **Zhengzhang Chen**[2*], **Xujiang Zhao**[2], **Linlin Yu**[3], **Zhong Chen**[4], **Yi He**[5], **Haifeng Chen**[2], **Chen Zhao**[1*]

[1]Department of Computer Science, Baylor University
[2]NEC Labs America
[3]School of Computer and Cyber Sciences, Augusta University
[4]School of Computing, Southern Illinois University
[5]Department of Data Science, The College of William and Mary
{dong_li1, chen_zhao}@baylor.edu, {zchen, haifeng}@nec-labs.com

Abstract

Uncovering causal structures from observational data is crucial for understanding complex systems and making informed decisions. While reinforcement learning (RL) has shown promise in identifying these structures in the form of a directed acyclic graph (DAG), existing methods often lack efficiency, making them unsuitable for online applications. In this paper, we propose MARLIN, an efficient multi-agent RL-based approach for incremental DAG learning. MARLIN uses a DAG generation policy that maps a continuous real-valued space to the DAG space as an intra-batch strategy, then incorporates two RL agents—state-specific and state-invariant—to uncover causal relationships and integrates these agents into an incremental learning framework. Furthermore, the framework leverages a factored action space to enhance parallelization efficiency. Extensive experiments on synthetic and real datasets demonstrate that MARLIN outperforms state-of-the-art methods in terms of both efficiency and effectiveness.

Introduction

Discovering and understanding the causal mechanisms behind natural phenomena is crucial in many scientific fields (Zhu, Ng, and Chen 2019). Consequently, methods for discovering causality from observational data have gained significant attention. Understanding these causal relationships helps predict the outcomes of interventions and hypothetical scenarios, which is highly valuable in areas like econometrics (Wold 1954). Due to the constantly changing nature of these fields, there is an increasing need to develop and use causal discovery methods in online settings to improve real-time decision-making and adaptability.

Identifying causal structures involves finding a Directed Acyclic Graph (DAG) $\hat{\mathcal{G}}$ that minimizes a score function $\mathcal{S}$ based on observed data $\mathbf{X}$:

$$\min_{\hat{\mathcal{G}}} \mathcal{S}(\hat{\mathcal{G}}, \mathbf{X}), \text{ s.t. } \hat{\mathcal{G}} \in \text{DAGs}. \quad (1)$$

However, this process is NP-hard (Wang et al. 2021) due to the super-exponential growth of the DAG space with the number of nodes (Robinson 1977) and the acyclicity constraint. Numerous methods have been proposed (Zheng et al. 2018; Ng, Ghassami, and Zhang 2020; Zhu, Ng, and Chen 2019; Li et al. 2025; Wang et al. 2021) to tackle this problem, with continuous optimization methods recently receiving significant attention. Pioneered by NOTEARS (Zheng et al. 2018), these methods transform the problem into a continuous optimization task by introducing a smooth feature to ensure acyclicity. However, they still tend to get stuck in local optima.

Reinforcement learning (RL) has become a promising approach for DAG learning due to its effective search strategies and explainable rewards (Zhu, Ng, and Chen 2019; Wang et al. 2021; Zhao et al. 2025; Yang et al. 2023). RL-BIC (Zhu, Ng, and Chen 2019) explores the full graph space and penalizes cycles via rewards but can't fully ensure acyclicity and is resource-intensive (Wang et al. 2021). Ordering-based methods (Yang et al. 2023) operate in the or-

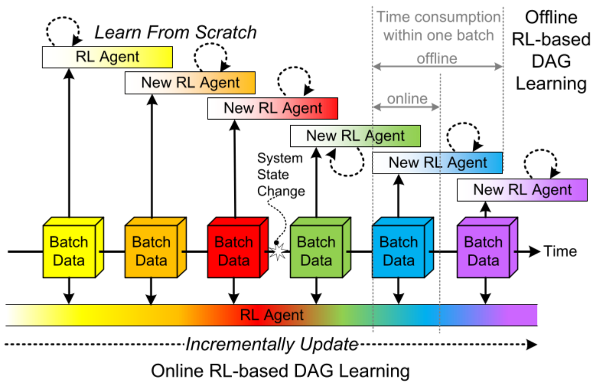


Figure 1: Comparison of the learning processes of offline (top) and online (bottom) RL-based DAG learning methods in an online data stream. The gradient colors on the RL agent represent its learning progress, with white indicating the initial state. When the color aligns with a data batch, it signifies that the agent has learned the current causal mechanism. Instead of learning from scratch, online DAG learning needs to incrementally and efficiently adapt to continuously arriving data batches and non-stationary data distributions.

[*]Corresponding authors

dering space to bypass acyclicity issues but rely on sequential decisions, limiting parallelization. These limitations hinder RL-based methods from scaling to real-world problems. Recent work ALIAS (Duong, Le, and Nguyen 2024) maps a real-valued vector to the DAG space and performs causal discovery without acyclicity constraints, enabling efficient RL.

Moreover, most existing RL-based methods are designed for offline settings, neglecting the more practical scenario of incremental DAG learning in online environments. Online DAG learning is particularly valuable for handling continuous data streams generated by modern applications, as it allows models to be updated incrementally with new data. This not only optimizes resource utilization but also enables real-time analysis and decision-making. When applied to online settings, RL-based methods encounter additional challenges due to stringent real-time requirements, as illustrated in Figure 1. First, online causal discovery demands that RL algorithms process incoming data batches efficiently, placing strict constraints on intra-batch performance. Second, unlike offline approaches that can be retrained from scratch, online methods must incrementally refine the model by integrating newly acquired information. This allows them to rapidly adapt to evolving system states and capture causal mechanisms under shifting data distributions, ensuring more effective and timely decision-making.

To address these challenges, in this paper, we propose MARLIN, an efficient multi-agent reinforcement learning framework for incremental DAG learning. We first develop an efficient intra-batch DAG learning method that maps from a continuous real-valued space to the DAG space without enforcing an acyclicity constraint. Building on this, we develop two RL agents to incrementally learn and disentangle state-invariant and state-specific causation from non-stationary online data across different batches, efficiently discovering the DAG continuously. Furthermore, we factor the action space to enable parallel DAG learning for efficiency improvement. Extensive experiments on both synthetic and real datasets validate the effectiveness and efficiency of MARLIN in incremental DAG learning.

Related Work

Traditional DAG Learning Methods. Existing DAG learning methods fall into four categories: **(i) Constraint-based methods**, like the PC algorithm (Spirtes, Glymour, and Scheines 2001), use conditional independence (CI) tests to recover DAG structures but depend heavily on CI test accuracy, making them unreliable when conflicts arise. **(ii) Score-based methods** evaluate DAGs using scoring functions (*e.g.*, BIC (Schwarz 1978)), but their combinatorial search complexity limits scalability. **(iii) Continuous optimization methods** reformulate DAG learning as a smooth optimization problem, as in NOTEARS (Zheng et al. 2018), but often struggle to escape local optima (Yu et al. 2019). **(iv) Sampling-based methods** estimate DAG posteriors but are computationally expensive, even with recent differentiable sampling techniques (Charpentier, Kibler, and Günnemann 2022). Traditional methods focus on offline settings and face efficiency challenges. In contrast, our approach, MARLIN, employs multi-agent reinforcement learning to efficiently search for global optima and adapt to online settings.

Reinforced DAG Learning. In recent years, RL has become a promising approach for combinatorial optimization, offering an intuitive search process and interpretable rewards to overcome local heuristic limitations. RL-BIC (Zhu, Ng, and Chen 2019) trains an RL agent to find high-reward DAGs with implicit acyclicity penalties but searches the entire directed graph space, making it inefficient and unable to strictly guarantee acyclicity. Ordering-based methods like CORL (Wang et al. 2021) and RCL-OG (Yang et al. 2023) reduce the search space by framing variable ordering as an MDP, mapping orderings to fully-connected DAGs (Teyssier and Koller 2012), and estimating DAGs via variable selection. While these methods avoid directly dealing with acyclicity, their sequential nature makes them unsuitable for parallelization and less efficient in online settings. Recent work ALIAS (Duong, Le, and Nguyen 2024) projects a real-valued vector into the DAG space, allowing causal discovery without enforcing acyclicity constraints, which facilitates efficient RL, making continuous and efficient DAG discovery possible. However, these methods focus solely on the offline setting, limiting their real-world applications. In contrast, our approach enables efficient and continuous DAG discovery through multi-agent RL.

Preliminaries

Structural Equation Model (SEM). Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a DAG, where each node $v_i \in \mathcal{V} = \{v_1, ..., v_d\}$ is associated with a random variable $X_i \in \mathcal{X} = \{X_1, ..., X_d\}$. Each directed edge $(v_i, v_j) \in \mathcal{E} = \{(v_i, v_j) | i, j = 1, ..., d \text{ and } i \neq j\}$ indicates that X_i is a direct cause of X_j. The DAG $\mathcal{G}$ can be equally represented by a binary adjacency matrix $\mathbf{A} \in \{0, 1\}^{d \times d}$, where the (i, j)-th entry is 1 if $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise. The joint distribution associated with $\mathcal{G}$ can be decomposed into $P(X_1, ..., X_d) = \prod_{i=1}^{d} P(X_i | \text{Pa}(X_i))$, where $\text{Pa}(X_i) = \{X_k \mid (v_k, v_i) \in \mathcal{E}\}$ is the set of parents of X_i in $\mathcal{G}$. We assume that the data generation process conforms to a SEM with additive noise:

$$X_i = f_i(\text{Pa}(X_i)) + \eta_i, i = 1, ..., d, \tag{2}$$

where $f_i(\cdot)$ represents the causal relationship between X_i and its parents $\text{Pa}(X_i)$, and the additive noise η_i is assumed to be jointly independent. We also assume *causal minimality*, meaning that each $f_i(\cdot)$ is not constant with respect to any of its arguments (Peters et al. 2014).

Non-Stationarity of Online Data. Real-world online data streams frequently exhibit non-stationarity (Shao et al. 2024), with the causal relationships among random variables being dynamic and subject to change over time. We assume that such changes are sufficient to induce partial changes in the underlying causal structure, leading to transitions in system state[1]. In this context, although the new system state

[1]We distinguish between "system state" and "state" as different concepts. The former refers to the underlying causal mechanisms of a system, while the latter is a concept used in RL.

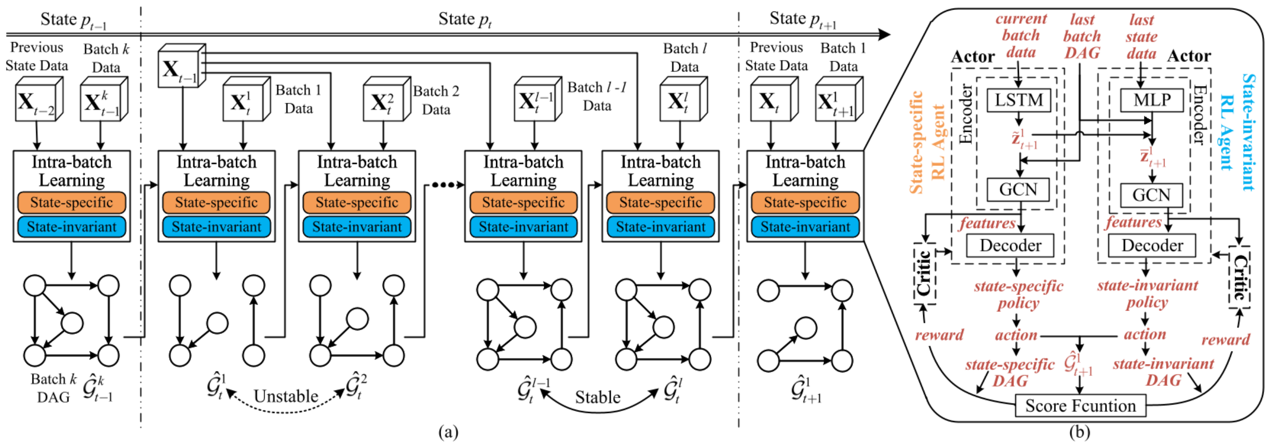


Figure 2: (a) The pipeline of MARLIN across three consecutive system states. For each batch, MARLIN learns the DAG using the intra-batch single-step RL algorithm, which includes state-specific and state-invariant RL agents optimizing their policies through an actor-critic approach. Detailed network architecture and variables are explained in Section . MARLIN facilitates efficient incremental DAG learning by disentangling state-specific and state-invariant causal relationships.

introduces causal relationships that are specifically dependent on it (*i.e.,* system **state-specific** causation), some inherent system causal relationships remain unchanged over time, reflecting the **state-invariant** aspects of the system's dynamics.

Problem Statement. Given a dataset $\mathcal{D} = \{\mathbf{X}_t\}_{t=1}^m$ comprising m sequentially continuous sets of observations $\mathbf{X}_t \in \mathbb{R}^{n_t \times d}$. Each $\mathbf{X}_t$ corresponds to a system state p_t, with n_t observations, and is associated with a DAG $\mathcal{G}_t$. In an online setting, data for each system state p_t arrives in batches of size b, denoted by $\mathbf{X}_t = [(\mathbf{X}_t^1)^\top, ..., (\mathbf{X}_t^L)^\top]^\top$, where $\mathbf{X}_t^l \in \mathbb{R}^{b\times d}, l = 1, ..., L$, represents the l-th batch of $\mathbf{X}_t$ and is associated with the DAG $\mathcal{G}_t^l$, which captures the causal mechanisms of the current batch data. Our goal is to **efficiently** perform DAG learning on each batch and **effectively** learn the estimated DAG $\hat{\mathcal{G}}_t$ for each system state p_t, aiming to achieve the best average score across all system states:

$$\min_{\hat{\mathcal{G}}_1,...,\hat{\mathcal{G}}_m} \frac{1}{m}\sum_{t=1}^{m} \mathcal{S}(\hat{\mathcal{G}}_t, \mathbf{X}_t), \text{ s.t. } \hat{\mathcal{G}}_t \in \text{DAGs}, \quad \hat{\mathcal{G}}_t \neq \hat{\mathcal{G}}_{t+1}, \forall t. \tag{3}$$

System state transition detection is beyond the scope of this work. Here, we use the Multivariate Singular Spectrum Analysis model (Alanqary, Alomar, and Shah 2021), an effective method for online change point detection, to identify state transition points in the online DAG learning process.

Methodology

We propose MARLIN, a multi-agent reinforcement learning framework designed for incremental DAG learning. As illustrated in Figure 2, MARLIN involves two modules: (i) Intra-batch Reinforced DAG Learning: We develop an intra-batch DAG generation method using two matrices derived solely from a real-valued vector, thereby mapping a continuous real space to the DAG space; (ii) Incremental Multi-agent Reinforced DAG Learning: This module leverages multi-agent reinforcement learning with state-specific and state-invariant agents to learn and disentangle state-invariant and state-specific causation across different batches. The approach is integrated into an incremental learning framework to efficiently uncover causal relationships. In addition, we explore the potential for parallel computation within this framework.

Intra-batch Reinforced DAG Learning

In this subsection, inspired by (Massidda et al. 2023; Duong, Le, and Nguyen 2024), we learn a DAG in a single batch through the score-based causal discovery method based on one-step reinforcement learning.

Sampling DAGs from a parameterized distribution is crucial for exploring the DAG space effectively. To ensure acyclicity, there is a well-established decomposition technique (Charpentier, Kibler, and Günnemann 2022) that decomposes a DAG into two binary matrices:

$$\mathbf{A} = \mathbf{P}^\top \mathbf{U}\mathbf{P}, \tag{4}$$

where $\mathbf{A} \in \{0,1\}^{d\times d}$ is the adjacency matrix of a DAG $\mathcal{G}$, $\mathbf{P} \in \{0,1\}^{d\times d}$ is a permutation matrix, and $\mathbf{U} \in \{0,1\}^{d\times d}$ is a strictly upper-triangular matrix. The matrix $\mathbf{U}$ represents the adjacency matrix of a graph that **ensures acyclicity**, with all directed edges (v_i, v_j) satisfying $i < j$. This captures all subsets of a specific fully-connected (FC) DAG corresponding to the "initial" ordering of nodes, where node $v_i \in \mathcal{V}$ is in the i-th position. The permutation matrix $\mathbf{P}$ changes the order of nodes in this initial ordering, resulting in a graph with the same topological structure. Consequently, $\mathbf{P}$ represents all subsets of the FC DAG corresponding to the altered ordering, allowing Eq. 4 to cover the entire DAG space. Based on this intuition, an arbitrary DAG $\mathbf{A} \in \{0,1\}^{d\times d}$ can be obtained from a FC DAG $\mathbf{H}$ and a binary mask matrix $\mathbf{S}$, as shown below:

$$\mathbf{A} = \mathbf{H} \odot \mathbf{S}, \tag{5}$$

where $\odot$ is the Hadamard product operator. Instead of time-consuming ordering-based methods to obtain a fully-connected matrix, we derives $\mathbf{H}$ from a single real-valued vector $\mathbf{h}$:

$$H_{ij} = \begin{cases} 1, & \text{if } h_i > h_j, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Since $\mathbf{S}$ can be easily obtained by filtering a real-valued matrix of the same shape with a simple threshold to produce a binary matrix, for any given real-valued vector $\mathbf{a}$ of dimension $d(d+1)$, we can generate a matrix $\mathbf{H}$ from its first d dimensions and a matrix $\mathbf{S}$ from the subsequent d^2 dimensions, thereby obtaining an arbitrary $\mathbf{A}$.

Based on the above idea, a single-step RL algorithm employs a stochastic policy π to learn an action a, which can be directly used to search the DAG space. The policy π selects a continuous action a from the real-valued space, which in turn determines the DAG of d nodes. After that, the reward $\mathcal{R}$ is calculated using a score function $\mathcal{S}$: $\mathcal{R}(\mathbf{a}, \mathbf{X}) = -\mathcal{S}(g(\mathbf{a}), \mathbf{X})$. Specifically, we use the Bayesian Information Criterion (BIC) score (Schwarz 1978) to enable straightforward comparisons with other RL-based methods.

Incremental Multi-Agent Reinforced DAG Learning

Incremental learning enables a model to update itself as new data arrives (Masana et al. 2022), eliminating the need for retraining from scratch. To tackle the problem outlined in Eq. 3, we propose a novel incremental learning framework based on multi-agent RL. Each agent builds upon the one-step reinforced DAG learning module described in Section and illustrated in Figure 2 (b). Specifically, we employ a state-invariant RL agent to incrementally learn causal relationships that remain consistent across different system states, and a state-specific RL agent to swiftly identify causal relationships unique to the current system state. This disentanglement mechanism enables MARLIN to efficiently capture state-specific causal mechanisms by leveraging the incrementally updated state-invariant information as acquired knowledge when encountering new data distributions, thus facilitating effective inter-batch incremental DAG learning.

State-specific RL Agent. Suppose the incoming data represents the l-th batch for system state p_t. The state-specific RL agent's objective is to learn the new causal relationships introduced by the data batch $\mathbf{X}_t^l$. To capture information changes across different batches, the encoding component utilizes both $\mathbf{X}_t^l$ and the previous hidden state as inputs to a long short-term memory network (LSTM) (Hochreiter and Schmidhuber 1997), producing the embedding $\tilde{\mathbf{z}}_t^l$ for current batch. This embedding is then combined with the DAG from the previous batch $\mathcal{G}_t^{l-1}$, to create an attributed graph. This graph, which incorporates prior structural knowledge, is subsequently encoded using a graph convolutional network (GCN) (Kipf and Welling 2016).

Then, a decoder is used to learn a state-specific policy $\tilde{\pi}_t^l$ that samples an action $\tilde{\mathbf{a}}_t^l$ to generate the state-specific DAG $\tilde{\mathcal{G}}_t^l$. This action is then combined with the $\bar{\mathbf{a}}_t^l$, which is sampled from the state-invariant policy $\bar{\pi}_t^l$, to produce the fusion action $\hat{\mathbf{a}}_t^l = \beta\tilde{\mathbf{a}}_t^l + (1-\beta)\bar{\mathbf{a}}_t^l$, where $\beta \in [0, 1]$ is used to balance the importance of state-specific information and state-invariant information. And then, the complete DAG $\mathcal{G}_t^l$ for $\mathbf{X}_t^l$ is obtained based on Section 4.1 via $\hat{\mathbf{a}}_t^l$. By default, we set the β to be 0.5.

To ensure accurate discovery of state-specific information, we introduce a decoupling term in the reward function to encourage the estimated state-specific DAG $\tilde{\mathcal{G}}_t^l$ to be as distinct as possible from both the previous batch's state-invariant DAG $\bar{\mathcal{G}}_t^{l-1}$ and the DAG from the previous state $\mathcal{G}_{t-1}$. This term is defined as follows:

$$\mathcal{L}_{\tilde{\mathcal{G}}_t^l} = \left(\|\tilde{\mathbf{A}}_t^l - \complement\bar{\mathbf{A}}_t^{l-1}\|^2 + \|\tilde{\mathbf{A}}_t^l - \complement\mathbf{A}_{t-1}\|^2 \right)/d, \tag{7}$$

where d is the number of nodes in $\mathbf{A}$ and $\complement\mathbf{A}$ denotes the complement of the adjacency matrix $\mathbf{A}$, which involves converting 0s in $\mathbf{A}$ to 1s and 1s to 0s. The current reward is defined as $\tilde{\mathcal{R}}_t^l = -\mathcal{S}(\mathbf{A}_t^l, \mathbf{X}_t^l) + \lambda_1 \mathcal{L}_{\tilde{\mathcal{G}}_t^l}$, where λ_1 represents the weight balancing the decoupling term and the BIC score. Since state-specific information is highly dependent on the system state, the state-specific RL agent is reinitialized at the beginning of each new system state.

State-Invariant RL Agent. The state-invariant RL agent aims to learn the causal relationships that remain consistent across multiple system states. The encoding process begins by using a fully connected layer to transform the previous state data $\mathbf{X}_{t-1}$ into embedding $\bar{\mathbf{z}}_{t-1}$. Given that state-invariant causal relationships are influenced by both $\mathbf{X}_{t-1}$ and $\mathbf{X}_t^l$, we concatenate $\bar{\mathbf{z}}_{t-1}$ and $\tilde{\mathbf{z}}_t^l$ to form $\bar{\mathbf{z}}_t^l$. The subsequent steps are similar to those in the state-specific RL agent: after encoding the attributed graph formed by $\mathcal{G}_t^{l-1}$ and $\bar{\mathbf{z}}_t^l$ using a GCN, the state-invariant policy $\bar{\pi}_t^l$ is learned through a decoder to generate the state-invariant DAG $\bar{\mathcal{G}}_t^l$. This DAG is then combined with $\tilde{\mathcal{G}}_t^l$ to produce $\mathcal{G}_t^l$. Additionally, a decoupling term is introduced to ensure that the estimated state-invariant DAG $\bar{\mathcal{G}}_t^l$ remains as dissimilar as possible to the previous batch's state-specific $\tilde{\mathcal{G}}_t^{l-1}$ while staying similar to $\mathcal{G}_{t-1}$. This is defined as:

$$\mathcal{L}_{\bar{\mathcal{G}}_t^l} = \left(\|\bar{\mathbf{A}}_t^l - \complement\tilde{\mathbf{A}}_t^{l-1}\|^2 + \|\bar{\mathbf{A}}_t^l - \mathbf{A}_{t-1}\|^2 \right)/d. \tag{8}$$

The reward is then defined as $\bar{\mathcal{R}}_t^l = -\mathcal{S}(\mathbf{A}_t^l, \mathbf{X}_t^l) + \lambda_2 \mathcal{L}_{\bar{\mathcal{G}}_t^l}$, where λ_2 is a weight coefficient. Since state-invariant information remains constant over time, the state-invariant RL agent is continuously updated throughout learning.

Optimization. Both agents are trained using the Adam optimizer (Kinga, Adam et al. 2015). We introduce a baseline for more stable training (Sutton and Barto 2018), so that each agent's objective is to minimize the temporal difference (TD) error between the critic's predicted rewards $\hat{\mathcal{R}}$ plus a baseline $\mathcal{B}$ and its actual rewards $\mathcal{R}$. The baseline $\mathcal{B}$ is updated according to the formula: $\mathcal{B} = \gamma \cdot \mathcal{B} + (1-\gamma) \cdot \bar{R}$, where γ is the discount factor and $\bar{R}$ denotes the mean of the rewards $\mathcal{R}$. The policy gradient is given by $\nabla J(\psi) = \mathbb{E}_{\pi(\psi)}\{\nabla_{\psi} \log \pi(\psi)[\mathcal{R} - (b + \hat{\mathcal{R}})]\}$.

Model Convergence within State. The estimated DAG is expected to gradually converge as successive data batches are processed. To avoid wasting computational resources, we define the similarity between the estimated DAGs of two consecutive batches within the same system state p_t using the Jensen-Shannon (JS) divergence (Fuglede and Topsoe 2004):

$$\xi = 1 - \text{JS}(P_{\mathcal{G}}(\mathcal{G}_t^{l-1})||P(\mathcal{G}_t^{l})), \tag{9}$$

where $P_{\mathcal{G}}(\cdot)$ denotes the edge distribution of graph. A larger ξ indicates a greater similarity between the two graphs. When ξ exceeds a certain threshold, we consider the current estimated DAG to have stabilized and will terminate the learning process for the current system state early, until a new system state arrives.

Factored Action Space for Parallelization

Our method operates in a single step, in contrast to the multi-step decision processes of ordering-based methods. Each position in our action vector $\mathbf{a}$ has a specific role in forming the final DAG (*e.g.*, the first d elements of $\mathbf{a}$ are used as $\mathbf{h}$ to obtain $\mathbf{H}$). Thus, the action space can be decomposed into multiple subspaces, transforming our problem into one involving a factored action space (Tang et al. 2022), which can then be parallelized across multiple processing units. Each unit explores a subspace, and the combination of these subspaces forms the complete action space, significantly enhancing efficiency when applied to online applications. We refer to this variant of factored action space as MARLIN-M.

Experiments

Experimental Setup

Datasets We conduct extensive experiments with various synthetic datasets of differing scales and generation methods (see Section for details) to evaluate the algorithm's capacity for incremental learning of DAGs. Furthermore, to evaluate MARLIN's effectiveness in real-world scenarios, we apply it to causal discovery-based root cause analysis (RCA) tasks (Zheng et al. 2024b; Wang et al. 2023c,a,b; Zheng et al. 2024a) using three time series datasets from real systems: **(i) OnlineBoutique (OB)** (Yu et al. 2023) is a microservice system for e-commerce composed of 10 microservices, which experienced 18 system faults during the data collection period. **(ii) Secure Water Treatment (SWaT)** (Mathur and Tippenhauer 2016; Zheng et al. 2024c) is a scaled-down model of a real industrial water treatment plant, equipped with 51 sensors and actuators. The dataset includes 3 hours of SWaT operation under normal conditions and 1 hour during which 6 attacks were executed. **(iii) Water Distribution (WADI)** (Ahmed, Palleti, and Mathur 2017; Zheng et al. 2024c) consists of data collected from a water distribution testbed with 123 sensors and actuators over 16 days of continuous operation, including 14 days under normal conditions and 2 days with 15 fault cases.

Synthetic Data Generation We introduce a synthetic data generation strategy tailored to our online setting. Starting from a complete Erdős–Rényi (ER) DAG with d nodes, we iteratively construct incomplete DAGs by randomly deleting edges and injecting $e\%$ noisy edges while preserving acyclicity. Observations are then generated following RL-BIC (Zhu, Ng, and Chen 2019), ordered from the most incomplete to the complete DAG, yielding datasets that capture system state transitions for validating incremental DAG learning.

Here, we focus primarily on three factors: the scale of the DAG (number of nodes d), the observations generation procedure (the type of regression methods for the causal mechanisms and noise), and the rate of noise added during the system state transition process (referred to as the transition noise rate $e\%$). Based on variations in these factors, we generated multiple synthetic datasets.

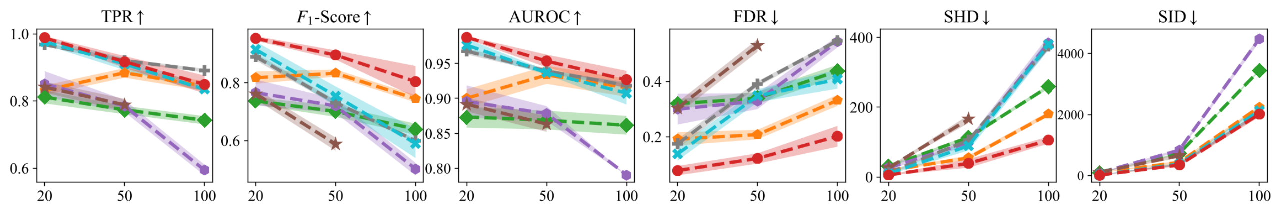
Baselines We compare MARLIN with seven DAG learning algorithms: the classical constraint-based algorithm PC (Spirtes, Glymour, and Scheines 2001), continuous optimization algorithms NOTEARS (Zheng et al. 2018), GOLEM (Ng, Ghassami, and Zhang 2020), and DAG-GNN (Yu et al. 2019), as well as RL-based algorithms RL-BIC (Zhu, Ng, and Chen 2019), CORL (Wang et al. 2021), and RCL-OG (Yang et al. 2023). Since these algorithms are designed to learn DAGs in offline settings, they lack incremental learning mechanisms, which prevents them from inheriting causations from previous data batches. Consequently, each new data batch requires learning from scratch to adapt to an online setting.

Evaluation Metrics We assess the performance using six common metrics: True Positive Rate (TPR), F_1-score, Area Under the Receiver Operating Characteristic Curve (AUROC), False Discovery Rate (FDR), Structural Hamming Distance (SHD), and Structural Intervention Distance (SID) (Peters and Bühlmann 2015). For the estimated graph, higher values of TPR, F_1-score, and AUROC are preferred, whereas lower values of FDR, SHD, and SID are desirable. We use the first state as historical offline data to train the initial model and then comprehensively evaluated the estimated DAGs learned by the algorithm on all subsequent states. We average the results across all states to evaluate the algorithm's performance over the entire dataset. Additionally, the average running time per batch (ATB) is calculated to measure the algorithm's efficiency.

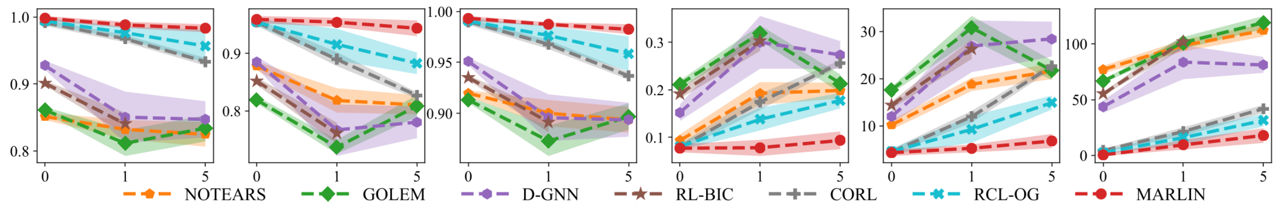
Performance Evaluation

Linear Model with Gaussian Noise We first evaluate the effectiveness of MARLIN using Linear-Gaussian (LG) synthetic datasets. These datasets are generated by applying a linear model with Gaussian noise, varying both the DAG scale ($d = \{20, 50, 100\}$) and the transition noise rate ($e = \{0, 1, 5\}$). The PC algorithm's results are excluded from the main discussion due to its suboptimal performance on dense graphs.

Figure 3 reports results on these datasets, yielding four insights: (1) MARLIN consistently surpasses nearly all baselines across metrics, highlighting its strong DAG learning and robustness to state transitions. (2) CORL achieves higher TPR than MARLIN at $d = 100$, but due to overly dense graphs with spurious edges, reflected in poor FDR and SHD—a pattern also seen in RCL-OG. Both methods degrade notably under increasing transition noise. (3) RL-BIC,



(a) DAG learning performance across different DAG scales (with d as x-axis).



(b) DAG learning performance across different transition noise rates (with e as x-axis).

Figure 3: Average performance of DAG learning across all states on synthetic Linear-Gaussian datasets, varying (a) DAG scale ($d = \{20, 50, 100\}$) and (b) transition noise rate ($e = \{0, 1, 5\}$) for MARLIN and other baselines. The shaded area represents the standard deviation; ↑ indicates that higher values are better, while ↓ indicates that lower values are better.

though RL-based, struggles with incremental DAG learning, worsening as graph size grows and failing at $d = 100$. (4) NOTEARS shows some promise in dynamic settings but still trails MARLIN, while GOLEM and DAG-GNN perform poorly, with DAG-GNN especially unstable as scale and complexity increase.

Non-Gaussian or Nonlinear Models To further assess the robustness of the algorithms under non-Gaussian noise and nonlinear models, we employ three distinct generation procedures, maintaining $d = 20$ and $e = 1$ constant: (i) a linear model with exponential noise (LE), (ii) a nonlinear model with quadratic regression (QR), and (iii) a nonlinear model utilizing Gaussian processes (GP). Performance and runtime efficiency of MARLIN, MARLIN-M, and the baseline methods on the remaining two synthetic datasets are presented in Table 1.

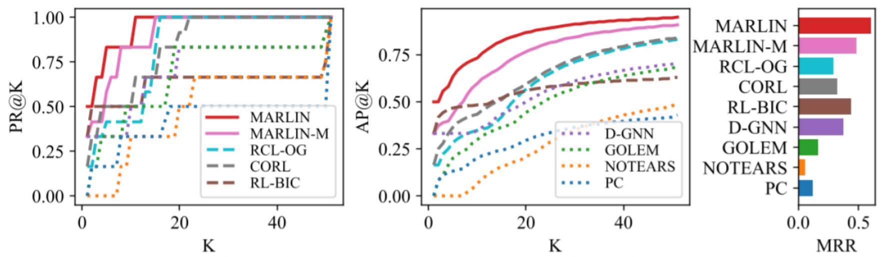
We make four key observations: First, MARLIN surpasses all baselines in both efficiency and accuracy across all datasets, showing strong effectiveness on nonlinear data. Second, MARLIN-M learns DAGs close in quality to MARLIN while offering greater stability and faster runtime, indicating that parallel computation enables real-time use with minimal performance loss. Its slight accuracy drop likely arises from the decomposition and reconstruction of the action space, which limits holistic causal modeling but remains an acceptable efficiency trade-off. Third, non-RL methods perform poorly because they struggle with nonlinear causal structures. Finally, although other RL-based methods handle nonlinear models well, their high computational cost makes them unsuitable for real-time systems with strict efficiency demands.

Application to Root Cause Analysis

To validate MARLIN's effectiveness, we evaluate its performance on real data. Due to the lack of ground-truth DAGs in complex real-world data, we assess its incremental DAG learning capability through root cause analysis (RCA), a key task in causal discovery (Wang et al. 2023a,b,c; Zheng et al. 2024b). After learning the DAG, we perform random walk with restarts (Tong, Faloutsos, and Pan 2006) to generate a rank list and compute three widely-used metrics—PR@K, AP@K, and MRR—where higher values indicate better. Efficiency is measured by the average running time per fault case (ATC). All methods use the same configurations as in the synthetic QR dataset experiment.

Microservice Data. Table 2 shows the RCA performance on the OB dataset. MARLIN outperforms all baselines in terms of runtime, coming second only to the PC algorithm, demonstrating the success of its intra-batch learning approach and efficiency enhancements. MARLIN consistently ranks the root cause among the top-3 in nearly all fault cases, indicating its ability to adapt well and learn causal mechanisms amid different system changes. The findings for MARLIN-M are consistent with previous results, significantly enhancing efficiency while learning high-quality DAGs.

Secure Water Treatment System Data. SWaT and WADI, being larger in scale, pose greater challenges than OB. SWaT results are in Figure 4. Non-RL methods struggle with noise and nonlinear causalities, while RL-based methods, though performing better, are too time-consuming for real-time systems. MARLIN and MARLIN-M surpass all baselines in performance and efficiency, identifying root causes faster with fewer rankings. Their balance between performance and efficiency ensures adaptability. Overall, our methods learn high-quality DAGs faster and with less data, even in complex settings.



(a) SWaT PR@K result. (b) SWaT AP@K result. (c) SWaT MRR result.

Figure 4: Overall performance on the SWaT dataset across (a) PR@K, (b) AP@K, and (c) MRR metrics.

		TPR↑	FDR↓	SHD↓	AUROC↑	SID↓	ATB↓
QR	PC	0.30(0.01)	0.65(0.02)	83.2(1.3)	0.58(0.00)	330.0(6.8)	–
	NOTEARS	0.28(0.01)	0.15(0.03)	78.0(1.9)	0.63(0.00)	270.0(6.5)	–
	GOLEM	0.36(0.01)	0.35(0.02)	78.2(1.4)	0.65(0.01)	245.0(8.6)	–
	D-GNN	0.31(0.00)	0.38(0.05)	86.5(3.1)	0.63(0.01)	264.0(5.9)	–
	RL-BIC	0.84(0.02)	0.30(0.02)	26.3(2.1)	0.89(0.01)	102.0(6.1)	330(60)
	CORL	0.88(0.01)	0.25(0.01)	21.1(0.7)	0.92(0.01)	78.1(11.1)	416(38)
	RCL-OG	0.90(0.03)	0.18(0.07)	15.6(6.2)	0.94(0.02)	68.9(18.1)	266(42)
	MARLIN	**0.94(0.01)**	**0.08(0.01)**	**7.0(0.7)**	**0.96(0.00)**	**49.6(7.1)**	81(9)
	MARLIN-M	0.90(0.01)	0.15(0.01)	14.2(0.4)	0.92(0.00)	65.1(5.6)	**32(3)**
GP	PC	0.19(0.01)	0.71(0.00)	74.8(1.2)	0.55(0.01)	315.2(3.2)	–
	NOTEARS	0.29(0.00)	0.58(0.01)	63.4(1.7)	0.61(0.01)	265.0(4.2)	–
	GOLEM	0.36(0.01)	0.51(0.01)	43.4(0.9)	0.65(0.00)	241.8(4.8)	–
	D-GNN	0.25(0.01)	0.62(0.06)	70.0(2.8)	0.58(0.01)	295.4(6.4)	–
	RL-BIC	0.80(0.01)	0.35(0.04)	31.3(2.2)	0.86(0.02)	159.8(8.5)	415(88)
	CORL	0.86(0.01)	0.27(0.01)	26.3(3.2)	0.88(0.01)	105.1(11.6)	455(34)
	RCL-OG	0.87(0.01)	0.23(0.04)	20.4(2.2)	0.92(0.02)	98.9(15.4)	293(30)
	MARLIN	**0.92(0.01)**	**0.15(0.02)**	**13.2(1.6)**	**0.95(0.02)**	**78.9(10.3)**	85(6)
	MARLIN-M	0.87(0.01)	0.20(0.01)	18.8(0.9)	0.91(0.01)	102.8(9.3)	**33(4)**

Table 1: DAG learning performance on synthetic QR and GP datasets with standard deviations reported in parentheses. A dash (–) for ATB indicates methods with notably poor performance. Bold indicates the best performance; ↑ indicates higher is better, and ↓ indicates lower is better.

	PR@1	PR@3	PR@5	AP@1	AP@3	AP@5	MRR	ATC↓
PC	16.7%	33.3%	50.0%	16.7%	27.8%	36.7%	35.2%	51
NOTEARS	38.9%	55.6%	72.2%	38.9%	50.0%	57.8%	54.4%	88
GOLEM	11.1%	27.8%	61.1%	11.1%	18.5%	32.2%	30.1%	227
D-GNN	33.3%	50.0%	72.2%	33.3%	40.7%	51.1%	47.9%	315
RL-BIC	33.3%	44.4%	72.2%	33.3%	40.7%	52.2%	49.7%	171
CORL	27.8%	72.2%	88.9%	27.8%	51.9%	65.6%	52.8%	141
RCL-OG	22.2%	77.8%	**100%**	22.2%	51.9%	68.9%	51.3%	122
MARLIN	**61.1%**	**94.4%**	**100%**	**61.1%**	**77.8%**	**86.7%**	**76.4%**	63
MARLIN-M	44.4%	88.9%	**100%**	44.4%	70.4%	82.2%	67.6%	**25**

Table 2: RCA Performance on OB dataset. All metrics except ATC are better when their values are higher.

		TPR↑	SHD↓	AUROC↑	SID↓	ATB↓
$d = 20$	**MARLIN**	**0.99(0.01)**	**5.24(0.8)**	**0.99(0.00)**	**9.6(2.4)**	26(2)
	MARLIN-S	0.96(0.01)	14.5(1.1)	0.97(0.00)	22.4(1.5)	**16(2)**
$d = 50$	**MARLIN**	**0.92(0.02)**	**38.9(8.4)**	**0.95(0.01)**	**302(41.9)**	**182(23)**
	MARLIN-S	0.88(0.02)	82.1(11.8)	0.90(0.01)	401.6(46.1)	195(18)
$d = 100$	**MARLIN**	**0.85(0.03)**	**105.5(10.1)**	**0.93(0.01)**	**1713.0(85.9)**	**1321(105)**
	MARLIN-S	0.82(0.02)	180.6(6.2)	0.86(0.01)	2240.3(67.7)	1687(111)

Table 3: Ablation study results on synthetic LG datasets with varying DAG scales ($d = 20, 50, 100$).

Ablation Study

To further explore the roles of state-specific and state-invariant RL agents in the continuous updating of DAGs, we have designed a single-agent variant, MARLIN-S, for an ablation study. MARLIN-S utilizes only one RL agent, which takes the current batch data as input and directly learns the complete DAG using the intra-batch learning approach from Section , bypassing the disentanglement process. Table 3 presents the empirical results of MARLIN and MARLIN-S on synthetic LG datasets with varying DAG scales.

MARLIN outperforms MARLIN-S, indicating that incremental disentangled DAG learning enhances causal structure learning. For small graphs with simple causal relations, a single agent is more efficient, but as scale and complexity grow, the advantages of a disentangled multi-agent design emerge. MARLIN rapidly captures state-specific information from new data, whereas MARLIN-S adapts more slowly. This underscores the need for both state-specific and state-invariant agents to improve performance and efficiency in fast, incremental DAG learning.

Conclusion

In this paper, we investigate the challenging problem of learning DAGs in an online setting. We propose MARLIN, an efficient multi-agent reinforcement learning (RL) framework designed for incremental DAG learning. MARLIN leverages an efficient intra-batch DAG learning method to learn a policy that maps from a continuous real-valued space to the DAG space. Building on this, MARLIN incorporates two RL agents—state-specific and state-invariant—to uncover causal relationships and integrates these agents into an incremental learning framework. Additionally, we explore the potential for parallel computation within this framework. Extensive experiments on both synthetic and real-world datasets demonstrate the effectiveness and efficiency of MARLIN for incremental DAG learning.

Acknowledgments

This work done by Yi He has been supported in part by the National Science Foundation (NSF) under Grant Nos. IIS-2505719, IIS-2441449, IOS-2446522, and the Commonwealth Cyber Initiative (CCI). The work done by Zhong Chen has been supported in part by an Illinois Innovation Network (IIN) sustaining Illinois seed funding grant. The work done by Chen Zhao has been supported in part by the National Science Foundation (NSF) under Grant No. CNS-2515265.

References

Ahmed, C. M.; Palleti, V. R.; and Mathur, A. P. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*, 25–28.

Alanqary, A.; Alomar, A.; and Shah, D. 2021. Change point detection via multivariate singular spectrum analysis. *Advances in Neural Information Processing Systems*, 34: 23218–23230.

Charpentier, B.; Kibler, S.; and Günnemann, S. 2022. Differentiable dag sampling. *arXiv preprint arXiv:2203.08509*.

Duong, B.; Le, H.; and Nguyen, T. 2024. ALIAS: DAG Learning with Efficient Unconstrained Policies. *arXiv preprint arXiv:2408.13448*.

Fuglede, B.; and Topsoe, F. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *International symposium onInformation theory, 2004. ISIT 2004. Proceedings.*, 31. IEEE.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Kinga, D.; Adam, J. B.; et al. 2015. A method for stochastic optimization. In *International conference on learning representations (ICLR)*, volume 5, 6. San Diego, California;.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Li, D.; Zhao, X.; Yu, L.; Liu, Y.; Cheng, W.; Chen, Z.; Chen, Z.; Chen, F.; Zhao, C.; and Chen, H. 2025. SolverLLM: Leveraging Test-Time Scaling for Optimization Problem via LLM-Guided Search. *arXiv preprint arXiv:2510.16916*.

Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A. D.; and Van De Weijer, J. 2022. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5): 5513–5533.

Massidda, R.; Landolfi, F.; Cinquini, M.; and Bacciu, D. 2023. Constraint-Free Structure Learning with Smooth Acyclic Orientations. *arXiv preprint arXiv:2309.08406*.

Mathur, A. P.; and Tippenhauer, N. O. 2016. SWaT: A water treatment testbed for research and training on ICS security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, 31–36. IEEE.

Ng, I.; Ghassami, A.; and Zhang, K. 2020. On the role of sparsity and dag constraints for learning linear dags. *Advances in Neural Information Processing Systems*, 33: 17943–17954.

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

Peters, J.; Mooij, J. M.; Janzing, D.; and Schölkopf, B. 2014. Causal discovery with continuous additive noise models.

Robinson, R. W. 1977. Counting unlabeled acyclic digraphs. In *Combinatorial Mathematics V: Proceedings of the Fifth Australian Conference, Held at the Royal Melbourne Institute of Technology, August 24–26, 1976*, 28–43. Springer.

Schwarz, G. 1978. Estimating the dimension of a model. *The annals of statistics*, 461–464.

Shao, M.; Li, D.; Zhao, C.; Wu, X.; Lin, Y.; and Tian, Q. 2024. Supervised algorithmic fairness in distribution shifts: A survey. *arXiv preprint arXiv:2402.01327*.

Spirtes, P.; Glymour, C.; and Scheines, R. 2001. *Causation, prediction, and search*. MIT press.

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

Tang, S.; Makar, M.; Sjoding, M.; Doshi-Velez, F.; and Wiens, J. 2022. Leveraging factored action spaces for efficient offline reinforcement learning in healthcare. *Advances in Neural Information Processing Systems*, 35: 34272–34286.

Teyssier, M.; and Koller, D. 2012. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. *arXiv preprint arXiv:1207.1429*.

Tong, H.; Faloutsos, C.; and Pan, J.-Y. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*, 613–622. IEEE.

Wang, D.; Chen, Z.; Fu, Y.; Liu, Y.; and Chen, H. 2023a. Incremental causal graph learning for online root cause analysis. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 2269–2278.

Wang, D.; Chen, Z.; Ni, J.; Tong, L.; Wang, Z.; Fu, Y.; and Chen, H. 2023b. Hierarchical graph neural networks for causal discovery and root cause localization. *arXiv preprint arXiv:2302.01987*.

Wang, D.; Chen, Z.; Ni, J.; Tong, L.; Wang, Z.; Fu, Y.; and Chen, H. 2023c. Interdependent causal networks for root cause localization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 5051–5060.

Wang, X.; Du, Y.; Zhu, S.; Ke, L.; Chen, Z.; Hao, J.; and Wang, J. 2021. Ordering-based causal discovery with reinforcement learning. *arXiv preprint arXiv:2105.06631*.

Wold, H. 1954. Causality and econometrics. *Econometrica: Journal of the Econometric Society*, 162–177.

Yang, D.; Yu, G.; Wang, J.; Wu, Z.; and Guo, M. 2023. Reinforcement causal structure learning on order graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 10737–10744.

Yu, G.; Chen, P.; Li, Y.; Chen, H.; Li, X.; and Zheng, Z. 2023. Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 553–565.

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

Zhao, Q.; Li, D.; Liu, Y.; Cheng, W.; Sun, Y.; Oishi, M.; Osaki, T.; Matsuda, K.; Yao, H.; Zhao, C.; et al. 2025. Uncertainty propagation on llm agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6064–6073.

Zheng, L.; Chen, Z.; Chen, H.; and He, J. 2024a. Online multi-modal root cause analysis. *arXiv preprint arXiv:2410.10021*.

Zheng, L.; Chen, Z.; He, J.; and Chen, H. 2024b. MULAN: multi-modal causal structure learning and root cause analysis for microservice systems. In *Proceedings of the ACM Web Conference 2024*, 4107–4116.

Zheng, L.; Chen, Z.; Wang, D.; Deng, C.; Matsuoka, R.; and Chen, H. 2024c. Lemma-rca: A large multi-modal multi-domain dataset for root cause analysis. *arXiv preprint arXiv:2406.05375*.

Zheng, X.; Aragam, B.; Ravikumar, P. K.; and Xing, E. P. 2018. Dags with no tears: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31.

Zhu, S.; Ng, I.; and Chen, Z. 2019. Causal discovery with reinforcement learning. *arXiv preprint arXiv:1906.04477*.