CORE: Towards Scalable and Efficient Causal Discovery with Reinforcement Learning

Andreas Sauter
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
a.sauter@vu.nl

Nicolò Botteghi
University of Twente
Enschede, The Netherlands
n.botteghi@utwente.nl

Erman Acar
IvI and ILLC, University of Amsterdam
Amsterdam, The Netherlands
e.acar@uva.nl

Aske Plaat
LIACS, Leiden University
Leiden, The Netherlands
a.plaat@liacs.leidenuniv.nl

ABSTRACT

Causal discovery is the challenging task of inferring causal structure from data. Motivated by Pearl's Causal Hierarchy (PCH), which tells us that passive observations alone are not enough to distinguish correlation from causation, there has been a recent push to incorporate interventions into machine learning research. Reinforcement learning provides a convenient framework for such an active approach to learning. This paper presents CORE, a deep reinforcement learning-based approach for causal discovery and intervention planning. CORE learns to sequentially reconstruct causal graphs from data while learning to perform informative interventions. Our results demonstrate that CORE generalizes to unseen graphs and efficiently uncovers causal structures. Furthermore, CORE scales to larger graphs with up to 10 variables and outperforms existing approaches in structure estimation accuracy and sample efficiency. All relevant code and supplementary material can be found at https://github.com/sa-and/CORE.

KEYWORDS

Causal Discovery, Reinforcement Learning

ACM Reference Format:
Andreas Sauter, Nicolò Botteghi, Erman Acar, and Aske Plaat. 2024. CORE: Towards Scalable and Efficient Causal Discovery with Reinforcement Learning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024,* IFAAMAS, 11 pages.

1 INTRODUCTION AND RELATED WORK

Causal discovery (CD) is the challenging task of inferring causal structure from data [10, 35]. Traditional approaches to causal discovery consider data from purely observational distributions. These are approaches such as constraint-based ones [14, 31], score-based ones [6], and more recently continuous optimization-based ones [38, 39].

Pearl's Causal Hierarchy (PCH) asserts that distinguishing between mere correlations and genuine causal relationships requires the integration of interventions in general [3]. As a response to this requirement, there has been a recent push to incorporate interventions into causal discovery research [12, 23] including machine learning [5, 18, 28], among others.

Reinforcement learning (RL) learns an optimal policy for sequential decision problems through interactions [32]. Therefore, RL is a promising framework for using interventions to investigate causal relationships. In particular, RL plays a dual role in the realm of causal discovery - it can be used not only to recover the causal structure of an environment [40], but also to learn causal discovery algorithms [28], thus representing a versatile tool for CD.

In particular, RL has also been used to search the space of causal structures more efficiently based on a fixed dataset [36, 40] with the possibility of incorporating prior knowledge [11]. Similarly, work on RL-related GFlow Nets [4] has been deployed to generate good estimates of the true causal structure [7, 17]. Furthermore, many integrations of RL with causal concepts have been investigated that restrict their CD process to supervised learning [9, 16, 21, 24, 34]. In addition to that, RL has also been used to learn policies that choose the best interventions to do for CD [1, 29, 33].

Although causal discovery has seen substantial progress with these works over the years, leading to a multitude of methodologies, challenges persist in areas such as scalability, generalization, and planning of interventions. In this context, this paper introduces CORE (**C**ausal Disc**O**very with **RE**inforcement Learning), a deep-RL-based algorithm designed for the task of learning a CD policy. CORE can learn a policy that sequentially reconstructs causal graphs from both observational and interventional data, while simultaneously performing informative interventions. This dual learning paradigm allows CORE not only to uncover causal structures efficiently, but also to identify interventions that enhance its causal models. The following lists our main contributions:

- We formalize the task of learning a CD algorithm as a partially observable Markov decision process (POMDP).
- We propose a dual Q-learning setup to learn intervention design and structure estimation simultaneously.
- We demonstrate that CORE can be successfully applied for causal discovery to previously unseen graphs of sizes of up to 10 variables.

In addition to those, we show the importance of jointly learning which interventions to perform and graph generation, and investigate the limitations of our approach regarding the applicability to the real world.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

The most distinctive feature of CORE is that it does not impose a specific algorithm for identifying causal models, but rather attempts to learn it. Among others, this can have positive effects on efficiency and transferability to new problem instances. While MCD [28] and AVICI [19] solve the same task, they run into pitfalls that hinder their application to realistic graph sizes or rely on offline data, respectively. We set steps to overcome these pitfalls by imposing additional structure on our policy, more efficient rewards, and learning to actively perform relevant interventions.

Our results show robust generalization to unseen graphs and the capability to scale to scenarios with up to ten variables, a step forward over the state of the art, and a crucial advancement towards addressing real-world complexities. The subsequent sections delve into the intricacies of CORE's architecture, its training methodologies, and empirical validations.

2 PRELIMINARIES AND NOTATION

In this section, we establish the necessary notation and provide an overview of key concepts and techniques used in the field of causal discovery with interventions and reinforcement learning.

2.1 Causal Models

Causal relations are often formalized through a *structural causal model* (SCM) which is a tuple $M = (\mathcal{X}, \mathcal{F}, \mathcal{U}, \mathcal{P})$ with a set of endogenous (random) variables (i.e., relevant variables for the problem) $\mathcal{X} = \{X_1, \ldots, X_n\}$, $\mathcal{U} = \{U_1, \ldots, U_n\}$ a set of exogenous (random) variables (often also called unobservable or noise variables), $\mathcal{F} = \{f_1, \ldots, f_n\}$ the set of functions (also called *structural equations*) whose elements are in the form of $X_i \leftarrow f_i(Pa(X_i), U_i)$ where $Pa(X_i) \subseteq \mathcal{X} \backslash \{X_i\}$ stands for endogenous parent variables of X_i, and $\mathcal{P} = \{P_1, ..., P_n\}$ the set of pairwise independent probability distributions defined over $\mathcal{U}$ with $U_i \sim P_i$.

Interpreting variables as nodes and the functional dependency between variables as directed edges, every SCM M induces a directed graph structure G, which we will call the corresponding causal graph. Directed edges represent direct causation from parent nodes to child nodes, hence absence of edges is as important as present edges. For the sake of simplicity, we shall follow the common assumption that no variable is its own cause i.e., there is no circular functional dependency, hence the induced causal graph is always a directed acyclic graph (DAG). Furthermore, each SCM M induces a joint distribution $P_M(\mathcal{X})$ over its endogenous variables, whose structural properties inherited from the corresponding induced graph G satisfy the Markov condition. That is, each X_i is independent of its non-descendants, given its parents $Pa(X_i)$. Along with the independence of the noise variables, this condition implies the following factorization [26]:

$$P_M(\mathcal{X}) = \prod_{X_i \in \mathcal{X}} P(X_i | Pa(X_i)) \tag{1}$$

We shall refer to this distribution as *observational distribution*.

Note that SCMs are generative models, i.e., we can sample values for $\mathcal{X}$ from them. We can sample the exogenous variables from $\mathcal{P}$ and determine the values of endogenous variables according to their functions in $\mathcal{F}$. This procedure effectively corresponds to sampling from the joint distribution over endogenous variables [26].

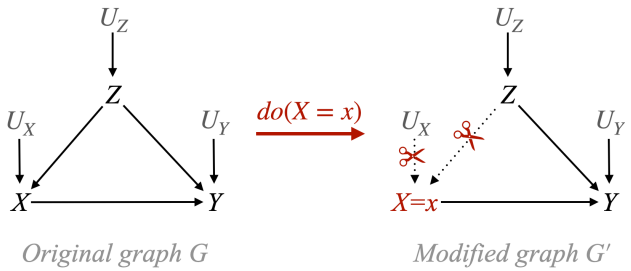

Figure 1: A simple graphical illustration of a (hard) intervention. Given the causal graph G with endogenous variables $\mathcal{X} = \{X, Y, Z\}$ and the corresponding noise variables $\mathcal{U} = \{U_X, U_Y, U_Z\}$, intervening on variable X (i.e., $do(X = x)$) results in modifying G into G' by pruning the incoming edges to node X and assigning the value x.

2.2 Interventions

Interventions play a crucial role in causal discovery, allowing us to investigate causal relationships by actively manipulating variables in a system. In general, imposed by Pearl's causal hierarchy [3], interventions are necessary to distinguish causation from correlation, and eventually to reason about causal effects.

Formally, an *intervention* on a variable X changes the variable's value to x (an arbitrary but fixed value), independently of X's actual causes. Then X is called the *intervention target*. Effectively, at the graph level, intervening on a variable X, removes all the edges incoming to X, resulting in $Pa(X) = \varnothing$. This operation is the so-called *do-operation* (denoted as $do(X = x)$), and allows us to distinguish the causal effect of variable X on variable(s) Y from the confounding influence of common parents of X and Y (Figure 1).

In an SCM, intervening on a variable X implies that the corresponding structural equation $f_X \in \mathcal{F}$ is replaced by $X \leftarrow x$, resulting in a modified SCM M'. Therefore, an intervention affects the distribution of the intervention target, since:

$$P_{M'}(X | Pa(X)) = P_{M'}(X | \varnothing) = P_{M'}(X) = \delta_x \tag{2}$$

where δ_x is the probability density function that has all mass on x. Put differently, an intervention replaces the factor associated with the intervened variable. We refer to the resulting joint distribution

$$P_M(\mathcal{X} | do(X = x)) = \prod_{X_i \in \mathcal{X} \setminus \{X\}} P_M(X_i | Pa(X_i)) \cdot P_{M'}(X = x) \tag{3}$$

as *post-interventional* distribution. To simplify the notation, we will sometimes use $P_{M_{do(X=x)}}(\mathcal{X})$ or $P_{M_{do(X)}}(\mathcal{X})$ to refer to the expression $P_M(\mathcal{X} | do(X = x))$ when the target variable or x is clear from the context.

2.3 Reinforcement Learning

Reinforcement learning (RL) is a general approach to learning through interaction with the world [32], especially in sequential decision problems. An RL agent aims to find the sequence of actions

that maximize the expected *return*, i.e., the cumulative (discounted) *reward*. How the RL agent selects its actions only relies on (possibly indirect) measures of how the world changes after each action is taken.

POMDP. In reinforcement learning problems, the relationship between an agent and the environment in which the states are not fully observable is often modeled as a Partially Observable Markov Decision Process (POMDP). Formally, a POMDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is a set of transition probabilities between states, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, Ω is a set of observations, $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$ is the set of conditional observation probabilities, and $\gamma \in [0, 1)$ is the discount factor.

RL:. The strategy of actions of an RL agent is called *policy*. A policy can be either deterministic or stochastic. A deterministic policy $\pi : \mathcal{S} \longrightarrow \mathcal{A}$ maps states to an action, whereas a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$ is characterized by a conditional distribution of actions given states. To maximize the return in the long run, RL agents often estimate the so-called *value* function $V : \mathcal{S} \longrightarrow \mathbb{R}$ or *action value* function $Q : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$. These functions determine the desirability of a state or a state-action pair, respectively. The optimal Q-function Q^* allows us to derive an optimal policy π^* that maximizes the return by greedily choosing the action that maximizes the value of each state, that is, $\pi^*(s) = \text{argmax}_a Q^*(s, a)$ [32].

Deep Q-Learning: The Q-learning algorithm [37] estimates the state-action value function Q using the *temporal difference* (TD). In particular, TD learning decomposes the problem of estimating the expected return of a given policy as the sum of the instantaneous reward and the value accumulated by following the optimal policy in the next step:

$$Q(s, a) = r(s, a) + \gamma \text{max}_{a'} Q(s', a') \tag{4}$$

where $r(s, a)$ is the instantaneous reward of the state-action pair. To estimate the future reward, we assume that the agent follows the optimal policy $\pi^*(s)$, i.e., $\text{argmax}_a Q^*(s, a)$. Especially in the first iterations of the algorithm, the estimate of Q does not correspond to the optimal value function Q^*. However, tabular Q-learning can still converge to the optimal solution [37]. The *deep Q-network (DQN)* algorithm [22] adapts the Q-learning algorithm to non-tabular settings, e.g., continuous state spaces, where the Q-function needs to be approximated via a neural network. DQN utilizes the TD-learning rule to generate a target for the training of the neural network that approximates the Q-value by means of the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(r(s, a) + \gamma \text{max}_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \tag{5}$$

where $Q(s, a; \theta)$ is the Q-function approximated by the neural network of parameters θ, while $Q(s, a; \theta^-)$ is the so-called target network, used to generate a fixed target and stabilize the training dynamics, and γ is the discount factor.

3 LEARNING A CAUSAL DISCOVERY POLICY WITH INFORMATIVE INTERVENTIONS

In this section, we present our algorithmic setup for learning causal discovery policies. We consider the classic agent-environment interaction scheme commonly used in RL. The goal is to learn a policy that represents a CD algorithm that uses observational and interventional data to sequentially estimate the true causal structures by performing informative interventions. Such modules can be applied to previously unseen causal structures through a few forward passes of a neural network without retraining, making them a highly efficient tool for causality [19, 28].

Following this line of research, we learn a policy that collects a stream of data by intervening on the environment to infer a causal structure estimate. This setup acknowledges the strong influence that informative interventions have, especially when there is a limited budget for interventions. We learn to perform these interventions by rewarding interventions that lead to the generation of a better structural update and limiting the budget for interventions by means of possible steps in an episode.

One key aspect of learned CD modules is that they need information about the ground truth causal structure only during training. This promises the possibility of (i) training the CD policy on synthetic data where the ground truth can easily be generated, and then (ii) applying it to estimate the causal structure of environments where the ground truth structure is potentially unknown, such as in the real world.

3.1 POMDP Formulation of Causal Discovery Through Interventions

To conveniently model the causal discovery process where there is partial observability, we will use a POMDP. However, since such a formalization is not entirely obvious, it constitutes our first contribution, which we will present in this section.

State Space: Our environment is determined by SCMs.[1] Therefore, each state will correspond to an SCM. We shall describe each SCM as the set of functions that determine the endogenous variables. Therefore, having n endogenous variables, a state is a set $s = \{f_0, ..., f_{n-1}\}$ of functions that define the current SCM. Furthermore, each state contains the ground truth observational graph G_s^* induced by the observational SCM $M_{do(\varnothing)}$.

Action Space: We model our action space as a multi-discrete space $A = [n + 1] \times [2n(n - 1) + 1]$ where the notation $[n] = \{1, 2, \ldots, n\}$ with n being the number of nodes in the graph. The first dimension of the action space represents the endogenous variables of the current SCM and determines the *intervention targets*. For each variable $X_i \in \mathcal{X}$, there is an action $do(X_i = c)$, where c is a predefined constant. In addition, the agent can do *nothing* and just collect observational data. In total, this dimension of the action space has $n + 1$ elements. The second dimension represents the *structural actions*. Each action in this space indexes the removal and addition of edges on the currently estimated graph. Additionally, the agent can perform a *void* structural update which leads to a total

[1] Our approach can be applied to any data-generating process that allows for sampling from and intervening on its variables.

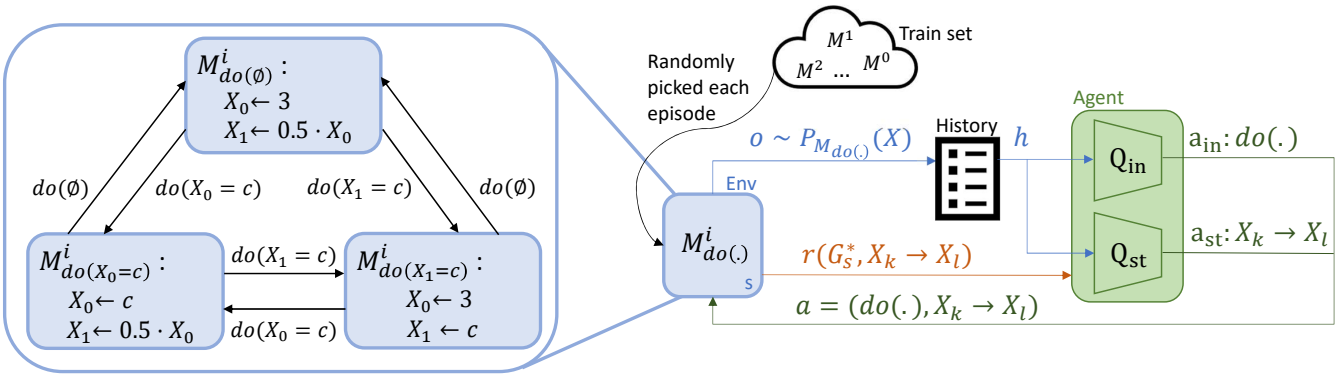

Figure 2: Overview of CORE's training setup (right) and a minimal example of the transition dynamics for an SCM with two endogenous variables (left). At each step, the agent picks the intervention/structural actions according to an ϵ-greedy policy on Q_{in} and Q_{st} respectively. The intervention is applied to the SCM M^i leading to a post-interventional distribution $P_{M_{do(.)}}$ from which an observation is sampled. The agent receives a reward based on the structure action and the induced graph of $M^i_{do(\emptyset)}$. The observation is added to the history of observations and serves as input to the agent. At the beginning of each episode a new M^j is drawn from the training set and the observation history is cleared.

size of $2n(n-1)+1$ actions in this dimension since we disallow reflective edges. The action space scales quadratically with the number of nodes. To address this problem, we mask the possible actions at every step such that the agent cannot add an already existing edge or remove a non-existing edge. This effectively halves the size of this action space dimension.

Transition Dynamics: Each episode starts in the observational SCM $M_{do(\emptyset)}$ where no intervention is performed. An intervention $do(X_i)$ on this SCM deterministically leads to a new SCM $M_{do(X_i)}$ where f_i is replaced by some constant c. This effectively replaces f_i in the state. With an intervention $do(X_j)$, we transition from $M_{do(X_i)}$ to $M_{do(X_j)}$, that is, $T(M_{do(X_i)}, do(X_j), M_{do(X_j)}) = 1$ or, equivalently, $T(M_{do(X_i)}, do(X_j)) = M_{do(X_j)}$. A minimal example of the transition dynamics of our approach can be seen in Figure 2.

Observations: At each step t, the agent collects the value of the endogenous variables $\{x_0, \ldots, x_{n-1}\}$ from the joint distribution $P_{M_{do(X)}}(\mathcal{X})$ induced by the current SCM $M_{do(X)}$. Therefore, the observation is $o_t \sim P_{M_{do(X)}}(\mathcal{X})$.

State Representation: The use of a single observation o_t is not sufficient to determine the best action in POMDPs. Thus, the agent has to build its own state representation h_t using the history of observations and actions [32]. We denote the history of observations and actions by $h_t = [x_0, a_0, ..., x_t, a_t]$.

Reward: The structural Hamming distance (SHD) measures the distance between two DAGs by counting the number of different edges. Since our goal is to minimize the distance between the generated and the ground truth observational graphs at every step, we consider the SHD as a natural candidate for our reward function. For a ground truth observational graph G^*_s, a graph estimate $\hat{G}_t$, and a graph estimate $\hat{G}_{t'}$ in the consecutive step t', we define the potential-based reward [15] as $r(s,a) = SHD(G^*_s, \hat{G}_t) - SHD(G^*_s, \hat{G}_{t'})$.

To simplify, we rewrite our reward function in the following way: Let $E(a)$ be the directed edge that is manipulated in action a. Then, when adding an edge $E(a)$, our reward becomes:

$$r(s,a) = \begin{cases} 1 & \text{if } E(a) \in G^*_s \\ -1 & \text{if } E(a) \notin G^*_s \end{cases} \tag{6}$$

When removing an edge $E(a)$ our reward becomes $-r(s,a)$. In all other cases ($E(a) = \varnothing$) the reward is 0.

This formulation has the computational advantage that only $E(a)$ must be compared to the edges of G^*_s instead of comparing the entire graph $\hat{G}_{s'}$. Furthermore, it makes the reward denser and depends only on s and a instead of relying on the entire history of structural actions that make up the current graph estimate. In Appendix A we demonstrate that this formulation is equivalent to $SHD(G^*_s, \hat{G}_s) - SHD(G^*_s, \hat{G}_{s'})$.

3.2 Data-Generation

We train our CORE agents using a training set of DAGs. In addition, we have an evaluation set of DAGs that the agent has not seen during training. To ensure that the evaluation set does not include any graphs from the training set, we first create a set of unique DAGs, shuffle it to ensure equal sparsity throughout the list, and then divide it into training and evaluation sets. As per common assumption in machine learning, we assume that having more graphs in the training set will help us to generalize better to the evaluation set.

Since the space of DAGs grows superexponentially in the number of its nodes, it quickly becomes infeasible to generate all possible graph structures with n nodes. For this reason, we generate all possible graphs only for graphs with 3 nodes (for a total of 25 graphs) and graphs with 4 nodes (for a total of 543 graphs). For graphs with more than 4 nodes, we generate subsets of the possible graphs. Similarly to many works in the literature, each graph is generated as an Erdös-Rényi [8] graph with an edge probability of

0.2. We diversify the training data by generating SCMs based on these graphs by sampling a function $f_i(Pa_G(X_i))$ from a class of possible functions for every node X_i in the graph G at the beginning of each episode.

3.3 Learning Approach

The CORE agent is based on the DQN algorithm [22], but it maintains two multilayer perceptron Q networks simultaneously. One network, $Q_{st}(h, a_{st}; \Theta_{st})$, estimates the Q-values specific to the structural updates, the other $Q_{in}(h, a_{in}; \Theta_{in})$ maintains the values for the interventions. Each of the two networks comes with its own target network, and the loss is computed separately. Note that Q values are determined on the basis of the history of observations, as is often done when applying DQN to POMDP problems [22]. The networks are identical except for the output layers due to different dimensionalities of the action space. The overall output of the Q-function is the concatenation of the individual Q-values, and our greedy policy picks a_{in} and a_{st} in such a way that the corresponding Q-values are maximized.

At the procedural level, our algorithm generates a new SCM based on a random sample of the graph data at the beginning of each episode. Furthermore, we start every episode with an empty graph estimate. The inference time for a given SCM is fully determined by the fixed number of steps per episode. This puts a hard upper bound on the number of samples and makes inference highly efficient. Figure 2 gives an overview of our agent.

4 GENERALIZATION TO UNSEEN STRUCTURES

In this section, we empirically validate whether our learned policy constitutes a good causal discovery algorithm. To this end, we train our model on a training set of SCMs with known causal structures and evaluate it on SCMs with causal structures that were not seen during training.

4.1 Training Data

For this experiment, we generate graphs with 3, 4, 5, 8, and 10 variables as described in Section 3.2. We split the generated graphs into training and test sets as follows: We first generate the graphs (25, 543, 16000, 91000, 101000), and then split the final list into train and test sets with splits 19/6, 401/142, 15000/1000, 90000/1000, 100000/1000 for 3, 4, 5, 8 and 10 variables, respectively. We limit the number of test graphs to 1000 since the evaluation would otherwise slow down the training prohibitively.

At the beginning of each episode, a graph is sampled from the data set. To generate the SCMs in accordance with Section 3.2, we define a class of linear additive functions. For each node X_i in the graph G, we sample $\mid Pa_G(X_i) \mid$ weights from $Uniform(0.5, 2.0)$. The generated function for this node is then $X_i \leftarrow \Sigma_{X_j \in Pa(X_i)} w_j \cdot x_j$ for the current values x_j of the parents of X_i. If X_i is a root node, we assign a default value of 0. We use an intervention value of 20 to provide a strong signal about the causal structure w.r.t. to the true causal effect sizes. This value can be considered as a hyperparameter and we discuss its impact further in Section 6.

4.2 Experimental Setup

We evaluate the generalization capability of CORE w.r.t. the available baselines. While AVICI [19] learns a graph generator that estimates the causal structure of an offline dataset, CORE operates in a few-shot online data sample regime. Therefore, a meaningful comparison with AVICI is out of reach. Consequently, MCD [28] is, to the best of our knowledge, the only SOTA method that learns a CD algorithm that actively intervenes. Furthermore, we compare with the *random* baseline that generates random DAGs, and the *empty* baseline which represents the empty graph.

We train both MCD and our approach for the same amount of steps and align all relevant hyperparameters including the neural network sizes. MCD runs into difficulties when scaling up to graphs with more than 4 nodes. Due to such computational infeasibility, we cannot run experiments for MCD on SCMs with 8 or 10 variables. A detailed description of the hyperparameters and the architectures used can be found in Appendix B. We set a maximum compute budget via a timeout of 25 training hours. The precise hardware configuration can be found in Appendix C. We paid special attention to setting comparable episode lengths for both approaches. For the sake of fair comparison, we set the episode length of our approach to half the episode length used in MCD, since our approach can perform interventions and structure updates synchronously, while MCD can only perform them sequentially.

4.3 Results

The results in Table 1 are from applying the learned models to three SCMs randomly generated for each graph in the held-out test set. The model that achieved the best performance during training was chosen for each evaluation.

We can see the favorable performance of CORE compared to MCD and the trivial baselines. For all sizes of graphs tested, we observe that our approach estimates graphs that are closer to the ground-truth structures than the other approaches in less than 34 milliseconds per graph. We generate an average of 0.5, 0.5, 1.3, 2.0, and 5.2 wrong edges in graphs with 3, 4, 5, 8, and 10 nodes, respectively. Even in a set of graphs with 10 variables, our approach adds approximately 2 correct edges out of 90 potential edges while only observing 15 data points. For smaller graphs, the ratio of correctly identified edges is even higher.

We attribute the improvement over MCD to a variety of aspects. First, addressing the structural actions and the intervention actions with separate networks makes learning the corresponding Q-functions more efficient. This is because two separate networks have a greater degree of freedom to represent structural and intervention actions that are inherently different. Second, representing the reward densely instead of a summary at the end of each episode often improves performance [25]. Third, instead of learning to integrate observations from the environment via a long short-term memory (LSTM) [13], we directly input the history of samples into our policy. And, lastly, by not generating graphs at runtime, but rather having a pre-defined training set, we avoid a significant computational overhead.

Furthermore, we observe the rather unfavorable performance of MCD compared to the baselines. We partly attribute this to a lack of extensive hyperparameter tuning, since this would likely have

	3 variables	4 variables	5 variables	8 variables	10 variables
random	3.29 ± 0.97	5.92 ± 1.41	8.33 ± 1.22	11.08 ± 2.80	14.85 ± 4.01
empty	1.80 ± 0.90	3.80 ± 1.10	6.20 ± 0.42	5.10 ± 1.60	7.0 ± 2.50
MCD	1.85 ± 1.10	4.97 ± 1.61	6.18 ± 0.44	-	-
CORE	0.50 ± 0.50	0.54 ± 0.65	1.26 ± 1.06	2.04 ± 1.64	5.16 ± 2.69

Table 1: Average SHDs on the test set of SCMs with unseen causal structures.

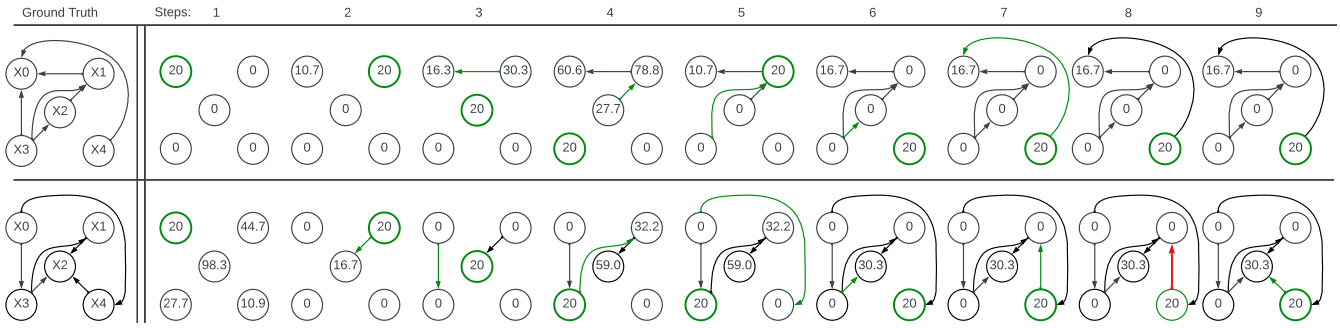

Figure 3: Two examples of how the learned CORE policy estimates the causal structure of two unseen SCMs described in Equations (7) and (8). Green elements indicate intervention (do (c = 20)) and structural update (adding an edge) in the current step, respectively. The red arrow indicates the deletion of an edge.

been needed to achieve the results in [28]. For the 4 and 5 variable cases, MCD reached the timeout of 25 hours.

Given these results, we conclude that CORE is capable of successfully learning a CD algorithm that can be applied to previously unseen causal structures. Even for these cases, our approach estimates the ground truth graph accurately without having to retrain on the new structure. Furthermore, we show that with CORE's novelties, we are able to scale towards graph sizes of more relevance for real-world applications, while simultaneously increasing training efficiency.

4.4 Examples

We present qualitative results on how our learned policy performs on the following two randomly selected example SCMs with unseen causal structures:

$$M^0_{do(\varnothing)} = \begin{cases} X_0 \leftarrow 0.54 \cdot X_1 + 0.91 \cdot X_3 + 0.83 \cdot X_4, \\ X_1 \leftarrow 1.52 \cdot X_2 + 1.84 \cdot X_3, \\ X_2 \leftarrow 1.38 \cdot X_3, \\ X_3 \leftarrow 0, \quad X_4 \leftarrow 0 \end{cases} \tag{7}$$

$$M^1_{do(\varnothing)} = \begin{cases} X_0 \leftarrow 0, \\ X_1 \leftarrow 1.61 \cdot X_3, \\ X_2 \leftarrow 0.83 \cdot X_1 + 1.60 \cdot X_3 + 1.5 \cdot X_4, \\ X_3 \leftarrow 1.39 \cdot X_0, \\ X_4 \leftarrow 0.54 \cdot X_0, \end{cases} \tag{8}$$

In Figure 3, we can see that in both cases the agent identifies the underlying causal structure almost correctly, with the exception of the missing edge $X_3 \rightarrow X_0$ in the first instance. In the second instance, it even recognizes an error and corrects it in Step 7.

5 ON THE IMPORTANCE OF JOINTLY LEARNING AN INTERVENTION POLICY

As described in Section 3, our method is designed to jointly learn a causal graph generator and an intervention policy. In this section, we show that learning an intervention policy, aimed at performing the interventions that are most informative for CD, helps in learning a CD policy.

It is worth noting that our agent does not receive any specific reward that represents the quality of the intervention performed in the environment. Instead, the reward function depends only on the structural update of the currently estimated causal structure (see Equation (6)). Since, for the full identification of the causal structure, interventions are generally needed [3], our agent has to learn to perform interventions to update the estimate of the causal structure. Therefore, our agent receives good rewards only if it performs interventions that are relevant to discover the current causal structure.

Although it is clear that interventions, in general, are helpful for CD, we argue that learning an intervention policy by measuring the usefulness for structure identification helps the overall learning process. Especially when the budget for performing interventions is very restrictive, as is the case in many real-world applications, it is crucial to perform the interventions that are most informative about the underlying causal structure [30, 33]. Which interventions are the most informative ones depends on the causal structure that is currently being discovered. This further motivates learning the intervention policy jointly with the structure generation policy. In this section, we empirically show that there is in fact a benefit in learning an intervention policy jointly with the CD policy.

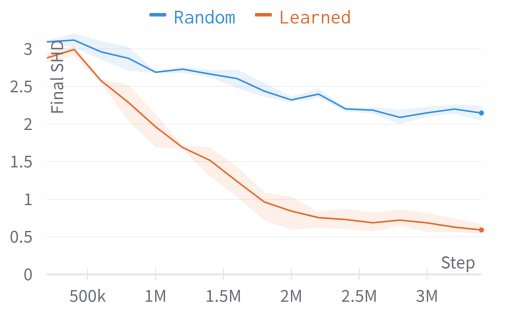

Figure 4: Plot of the average SHD on the test set (lower is better). We present the means over three training runs of CORE with random interventions (blue) and when jointly learning an intervention policy (red) over graphs with 4 variables.

5.1 Experimental Setup

To show the hypothesized performance gain, we compare the performance of the agent that learns the intervention policy and the structure generation policy jointly, to an agent that learns only the structure generation policy and randomly picks an intervention target at each step. We train the two agents 3 times each on environments with 4 endogenous variables. The graphs, function classes, and hyperparameters remain the same as described in Section 4.1.

5.2 Results

Figure 4 shows the aggregated results for the two types of agents. We can see that learning the intervention policy jointly with the CD policy results in a significantly better estimation of the ground truth causal structure. Additionally, it decreases the number of learning steps needed to reach a certain level of performance. Lastly, Figure 4 suggests that even with random interventions, our approach performs reasonably well (average SHD of ~ 2.3). This indicates the robustness of the CORE agent to less informative interventions.

Overall, we observe that the ability to learn the intervention policy is an integral part of learning a CD policy and that rewarding interventions leading to better structure estimates is sensible.

6 APPLICABILITY TO THE REAL-WORLD

Throughout this work, we acknowledge the capability of learning CD algorithms that can be applied to environments with previously unseen causal structures with up to 10 variables. This constitutes a substantial improvement over the SOTA when it comes to the application of learned CD algorithms in the real world, as many problems can be modeled with 10 variables [27, 41]. Therefore CORE, reaches graph sizes that are relevant in CD. However, we acknowledge that many applications with up to 5000 variables [20] are currently out of reach. In this section, we shed light on some of the limitations that this approach currently has with regard to applying it in a real-world scenario.

Here, we specifically investigate two design aspects that limit real-world applicability, and they are interconnected. First, during training, the functions of the SCMs are sampled from a specific function class, and for some function classes (e.g., non-linear functions), discovering the true causal structure can be harder than for others [10]. Consequently, as we will show in this section, learning a CD algorithm for these classes of functions is more difficult.

Second, our approach is tailored to generate graph estimates for the function class on which it was trained. This means that when used for different causal structures, the same function class is expected during inference. Consequently, this can lead to a decrease in performance if the function class is altered. We expect an exception for this for function classes that are either very similar to the training functions or that subsume them. Therefore, when CORE is trained with the intention of being used in a real-world setting, the real-world function class has to be anticipated during training.

6.1 Transferability across Noise and Non-Linearity

Motivated by these aspects, we show the difficulty in training CD policies on some function classes and test CORE on function classes that it was not trained on.

6.1.1 Experimental Setup. For the data-generating processes in this section, we test how noise and non-linearities influence the performance and transferability of CORE. Therefore, we use three function classes that define each function class $f_i(Pa(X_i), U_i)$ in an SCM as follows:

- *linear:* $f_i = \Sigma_{X_j \in Pa(X_i)} w_j \cdot x_j$ (same as Section 4.1)
- *linear + noise:* $f_i = \Sigma_{X_j \in Pa(X_i)} w_j \cdot x_j + u_i$, where $u_i \sim \mathcal{N}(0, 0.5)$
- *interaction:* $f_i = \Sigma_{X_j \in Pa(X_i)} w_j \cdot x_j + x_k \cdot x_l$, where $X_j, X_k \in Pa(X_i)$

where the lowercase x_i represents the current value of the variable X_i. Whenever a node is a root node, we set a default value of 0.

We train two CORE models for various graph sizes for each of the two linear functions, one with an intervention value of 5 and the other with an intervention value of 20. This setup gives us further insight into how the signal-to-noise ratio affects our performance. For the interaction function, we train one model with an intervention value of 5. We then tested all the trained models in all three function classes.

6.1.2 Results. In Table 2, we show how models that were trained with one function class perform when applied to various function classes with previously unseen causal structures.

Our first observation is that, as hypothesized, the application of our learned CD algorithm on previously unseen function classes is problematic if the testing function class is very different from the training function class. While applying the linear models to their noisy/non-noisy counterparts still leads to good estimates, applying them to the interaction data mostly fails. This supports our claim that the function class chosen for training must be informative about the function class encountered during testing.

Looking at the model that was trained on interaction data, we observe two interesting aspects. First, for graphs with 4 and 5 variables, CORE fails to learn a CD policy that generalizes to unseen graphs. We believe that, given the right hyperparameters and a sufficient training budget, we can solve the task for larger graphs,

	3 Variables			4 Variables			5 Variables		
	lin	lin + noise	interaction	lin	lin + noise	interaction	lin	lin + noise	interaction
empty	1.8 ± 0.9			3.8 ± 1.1			6.2 ± 0.4		
lin 5	0.2 ± 0.4	1.2 ± 0.4	2.2 ± 0.7	0.6 ± 0.7	0.8 ± 0.8	3.4 ± 1.9	1.4 ± 1.1	1.8 ± 1.2	6.1 ± 2.4
lin 20	0.5 ± 0.5	0.8 ± 0.4	2.3 ± 0.5	0.5 ± 0.7	0.6 ± 0.6	3.8 ± 1.8	1.3 ± 1.1	1.2 ± 1.0	6.5 ± 2.2
lin+noise 5	0.7 ± 0.7	0.6 ± 0.7	1.3 ± 0.7	0.6 ± 0.7	0.8 ± 0.8	3.6 ± 2.0	2.0 ± 1.3	2.3 ± 1.3	5.9 ± 2.1
lin+noise 20	0.2 ± 0.4	0.8 ± 0.4	2.0 ± 0.8	0.6 ± 0.6	0.7 ± 0.7	4.3 ± 1.7	1.2 ± 1.0	1.2 ± 1.0	6.8 ± 2.2
interaction 5	0.8 ± 0.7	1.3 ± 0.7	1.0 ± 0.5	4.1 ± 1.3	4.1 ± 1.3	4.0 ± 1.3	7.9 ± 1.7	7.8 ± 1.7	7.9 ± 1.7

Table 2: We show the performance of trained CORE policies on unseen graphs with various function classes for their corresponding SCMs. Each row describes which function class the model was trained on and what intervention value it uses. Each column describes the function class it was tested on. Empty describes the baseline that generates the empty graph.

based on the results obtained from smaller graphs. However, these results demonstrate that the performance of a learned CD algorithm depends on the complexity of the SCM function that generates the data. Second, we observe that, for the interaction case, the learned policy can be successfully applied to the linear function. We argue that this is because the interaction data encompasses the linear data as well. This suggests that if the function classes used during training are broad enough, the CD algorithm that is learned will be relatively more applicable to real-world scenarios.

When comparing linear models with a higher intervention value with those with a lower intervention value, we observe that they tend to perform better on function classes that they successfully learned. We attribute this to a higher signal-to-noise ratio w.r.t. the data-generating process.

Overall, we can say that learning CD algorithms is limited by the function class that is observed during training. This currently obstructs their application to real-world scenarios, but finding more general functions on which CORE can be trained is a promising research direction.

6.2 Further Limitations

Apart from aspects related to the function classes on which CORE is trained, we point out the additional limitations of learning a model that is applicable to the real world.

One of them is the assumption of being able to intervene on any variable with any target value. In real-world scenarios, it might be that some variables cannot be manipulated (imagine changing the outside temperature) or a good target value is unknown during training. Furthermore, CORE-like algorithms might suffer from the presence of unobserved confounders. For both the unknown target value and unobserved confounders, there is hope that augmenting the learning procedure in the future will overcome these limitations.

7 SUMMARY AND CONCLUSION

In this paper, we introduce CORE, a deep RL-based approach to tackle the task of causal discovery. CORE learns a policy to sequentially perform informative interventions and generate candidate causal graphs from scratch. Moreover, the learned policy generalizes to previously unseen graphs of up to 10 variables in size. CORE outperforms the current SOTA baseline (i.e., MCD [28]) both in the number of variables it can deal with and in the accuracy of the estimated structure. Furthermore, it demonstrates that by learning to perform the most informative interventions, highly sample-efficient CD algorithms can be learned (~ 15 data samples for 10 variables).

Such improvement can be attributed to several key design features. One such feature is the imposed additional structure in the policy that separates the networks for interventions and structural updates. However, such separation is not completely isolated. As shown in our ablation study, CORE learns to perform relevant interventions outperforming random interventions. Learning which interventions are relevant is guided solely through a dense reward that assesses the accuracy of the generated graph.

Moreover, we outlined the real-world applicability of our approach in terms of the number of variables and generalizability across more complex function classes. For the former, while the number of variables that CORE can deal with matches some domains, for some other domains, usual practice is still out of reach. For the latter, it turns out that CORE delivers good estimates when it comes to linear functions, training on noisy functions, and testing on non-noisy counterparts, and vice versa. However, generalizing to more complex classes such as non-linear functions necessitates improvements. Furthermore, we empirically confirmed that training on more complex function classes and testing on simpler classes yields promising estimates. Such an observation can serve as a key idea along the road of applying these methods to real-world problems.

Future work includes investigating the limitations of our approach in the presence of confounders, soft interventions, and determining the right target value for interventions. In particular, it would be interesting to understand how robust our approach is when it comes to different topologies regarding such non-intervenable variables and unobservable confounders. Another avenue that we plan to address is an extensive study of the transferability of our approach across different function classes, such that learned CD algorithms that autonomously plan interventions can be applied to real-world data.

ACKNOWLEDGMENTS

We thank Frank van Harmelen for his valuable input throughout this project and the anonymous reviews for helping to improve the final version of this work. This research was partially funded by the Hybrid Intelligence Center, a 10-year programme funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl, grant number 024.004.022

REFERENCES

[1] Amir Amirinezhad, Saber Salehkaleybar, and Matin Hashemi. 2022. Active learning of causal structures with deep reinforcement learning. *Neural Networks* 154 (10 2022), 22–30. https://doi.org/10.1016/J.NEUNET.2022.06.028

[2] Henri Bal, Dick Epema, Cees de Laat, Rob van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. 2016. A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term. *Computer* 49, 5 (5 2016), 54–63. https://doi.org/10.1109/MC.2016.127

[3] Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. 2022. On Pearl's Hierarchy and the Foundations of Causal Inference. In *Probabilistic and Causal Inference*. ACM, New York, NY, USA, 507–556. https://doi.org/10.1145/3501714.3501743

[4] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. 2023. GFlowNet Foundations. *Journal of Machine Learning Research* 24, 210 (2023), 1–55. http://jmlr.org/papers/v24/22-0364.html

[5] Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. 2020. Differentiable Causal Discovery from Interventional Data. In *Advances in Neural Information Processing Systems*, H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin (Eds.), Vol. 33. Curran Associates, Inc., 21865–21877. https://proceedings.neurips.cc/paper_files/paper/2020/file/f8b7aa3a0d349d9562b424160ad18612-Paper.pdf

[6] David Maxwell Chickering. 2003. Optimal Structure Identification with Greedy Search. *J. Mach. Learn. Res.* 3, null (3 2003), 507–554. https://doi.org/10.1162/153244303321897717

[7] Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. 2022. Bayesian structure learning with generative flow networks. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research, Vol. 180)*, James Cussens and Kun Zhang (Eds.). PMLR, 518–528. https://proceedings.mlr.press/v180/deleu22a.html

[8] P. Erdós and A. Rényi. 1959. On random graphs. *Publicationes Mathematicae* (1959), 290–297.

[9] Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. 2022. Factored Adaptation for Non-Stationary Reinforcement Learning. In *Advances in Neural Information Processing Systems*, S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho, and A Oh (Eds.), Vol. 35. Curran Associates, Inc., 31957–31971. https://proceedings.neurips.cc/paper_files/paper/2022/file/cf4356f994917177213c55ff438ddf71-Paper-Conference.pdf

[10] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review of Causal Discovery Methods Based on Graphical Models. *Frontiers in Genetics* 10 (6 2019). https://doi.org/10.3389/fgene.2019.00524

[11] Uzma Hasan and Md Osman Gani. 2022. Kcrl: A prior knowledge based causal discovery framework with reinforcement learning. In *Machine Learning for Healthcare Conference*. 691–714.

[12] Alain Hauser and Buhlmann@stat Math Ethz Ch. 2012. Characterization and Greedy Learning of Interventional Markov Equivalence Classes of Directed Acyclic Graphs Peter B ¨ uhlmann. *Journal of Machine Learning Research* 13 (2012), 2409–2464.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (11 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[14] Antti Hyttinen, Frederick Eberhardt, and Matti Järvisalo. 2014. Constraint-based Causal Discovery: Conflict Resolution with Answer Set Programming. In *Conference on Uncertainty in Artificial Intelligence*. 340–349.

[15] Erik Jenner, Herke van Hoof, and Adam Gleave. 2022. Calculus on MDPs: Potential Shaping as a Gradient. (8 2022). https://arxiv.org/abs/2208.09570v2

[16] Anson Lei, Bernhard Schölkopf, and Ingmar Posner. 2022. Causal Discovery for Modular World Models. In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*. https://openreview.net/forum?id=VfkjQzdGCH

[17] Wenqian Li, Yinchuan Li, Shengyu Zhu, Yunfeng Shao, Jianye Hao, and Yan Pang. 2022. GFlowCausal: Generative Flow Networks for Causal Discovery. (2022). http://arxiv.org/abs/2210.08185

[18] Phillip Lippe, Taco Cohen, and Efstratios Gavves. 2021. Efficient Neural Causal Discovery without Acyclicity Constraints. In *International Conference on Learning Representations*.

[19] Lars Lorch, Scott Sussex, Jonas Rothfuss, Andreas Krause, and Bernhard Schölkopf. 2022. Amortized Inference for Causal Structure Learning. In *Advances in Neural Information Processing Systems*, S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho, and A Oh (Eds.), Vol. 35. Curran Associates, Inc., 13104–13118. https://proceedings.neurips.cc/paper_files/paper/2022/file/54f7125dee9b8b3dc798bb9a082b09e2-Paper-Conference.pdf

[20] Marloes H. Maathuis, Diego Colombo, Markus Kalisch, and Peter Bühlmann. 2010. Predicting causal effects in large-scale systems from observational data. *Nature Methods 2010 7:4* 7, 4 (4 2010), 247–248. https://doi.org/10.1038/nmeth0410-247

[21] Arquímides Méndez-Molina, Eduardo F.Morales, and L Enrique Sucar. 2022. Causal Discovery and Reinforcement Learning: A Synergistic Integration. In *Proceedings of The 11th International Conference on Probabilistic Graphical Models (Proceedings of Machine Learning Research, Vol. 186)*, Antonio Salmerón and Rafael Rumi (Eds.). PMLR, 421–432. https://proceedings.mlr.press/v186/mendez-molina22a.html

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015). https://doi.org/10.1038/nature14236

[23] Joris M Mooij, Sara Magliacane, and Tom Claassen. 2020. Joint Causal Inference from Multiple Contexts. *Journal of Machine Learning Research* 21 (2020), 1–108. http://jmlr.org/papers/v21/17-123.html.

[24] Suraj Nair, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. 2019. Causal Induction from Visual Observations for Goal Directed Tasks. (2019).

[25] Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. *ICML* (1999), 278–287.

[26] Judea Pearl. 2009. Introduction to Probabilities, Graphs, and Causal Models. In *Causality*. Cambridge University Press, 1–40. https://doi.org/10.1017/CBO9780511803161.003

[27] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. 2005. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science* 308, 5721 (4 2005), 523–529. https://doi.org/10.1126/science.1105809

[28] Andreas W M Sauter, Erman Acar, and Vincent Francois-Lavet. 2023. A Meta-Reinforcement Learning Algorithm for Causal Discovery. In *Proceedings of the Second Conference on Causal Learning and Reasoning (Proceedings of Machine Learning Research, Vol. 213)*, Mihaela van der Schaar, Cheng Zhang, and Dominik Janzing (Eds.). PMLR, 602–619. https://proceedings.mlr.press/v213/sauter23a.html

[29] Nino Scherrer, Olexa Bilaniuk, Yashas Annadani, Anirudh Goyal, Patrick Schwab, Bernhard Schölkopf, Michael C Mozer, Yoshua Bengio, Stefan Bauer, and Nan Rosemary Ke. 2022. Learning Neural Causal Models with Active Interventions. (2022).

[30] Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G Dimakis, and Sriram Vishwanath. 2015. Learning Causal Graphs with Small Interventions. In *Advances in Neural Information Processing Systems*, C Cortes, N Lawrence, D Lee, M Sugiyama, and R Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/b865367fc4c0845c0682bd466e6ebf4c-Paper.pdf

[31] Peter Spirtes, Clark N Glymour, and Richard Scheines. 2000. *Causation, prediction, and search*. MIT press.

[32] R.S. Sutton and A.G. Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[33] Panagiotis Tigas, Yashas Annadani, Andrew Jesson, Bernhard Schölkopf, Yarin Gal, and Stefan Bauer. 2022. Interventions, Where and How? Experimental Design for Causal Models at Scale. *Advances in Neural Information Processing Systems* 35 (12 2022), 24130–24143. https://github.com/yannadani/cbed

[34] Jean-François Ton, Dino Sejdinovic, and Kenji Fukumizu. 2021. Meta learning for causal direction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9897–9905.

[35] Matthew J. Vowels, Necati Cihan Camgoz, and Richard Bowden. 2023. D'ya Like DAGs? A Survey on Structure Learning and Causal Discovery. *Comput. Surveys* 55, 4 (4 2023), 1–36. https://doi.org/10.1145/3527154

[36] Xiaoqiang Wang, Yali Du, Shengyu Zhu, Liangjun Ke, Zhitang Chen, Jianye Hao, and Jun Wang. 2021. Ordering-Based Causal Discovery with Reinforcement Learning. (2021). https://arxiv.org/abs/2105.

[37] Christopher J C H Watkins and Peter Dayan. 1992. Q-Learning. 8 (1992), 279–292.

[38] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. DAG-GNN: DAG Structure Learning with Graph Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 7154–7163. https://proceedings.mlr.press/v97/yu19a.html

[39] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P Xing. 2018. DAGs with NO TEARS: Continuous Optimization for Structure Learning. (2018). https://github.com/xunzheng/notears.

[40] Shengyu Zhu, Ignavier Ng, and Zhitang Chen. 2019. Causal Discovery with Reinforcement Learning. (2019). http://arxiv.org/abs/1906.04477

[41] Matjaz Zwitter and Milan Soklic. 1988. Breast Cancer. UCI Machine Learning Repository.

A EQUIVALENCE OF OUR REWARD AND THE DIFFERENCE IN SHDS

Let G_s^* be the observational ground-truth graph of the current SCM $M^i_{do(\varnothing)}$. Furthermore, let $\hat{G}_s$ be the estimated graph at state s, $\hat{G}_{s'}$ the estimated graph in the consecutive state and $E(a)$ the edge that is manipulated by action a. We show that

$$SHD(G_s^*, \hat{G}_s) - SHD(G_s^*, \hat{G}_{s'}) = \begin{cases} 1 & if E(a) \in G_s^* \\ -1 & if E(a) \notin G_s^* \\ 0 & if E(a) = \varnothing \end{cases} \tag{9}$$

if a adds an edge and

$$SHD(G_s^*, \hat{G}_s) - SHD(G_s^*, \hat{G}_{s'}) = \begin{cases} -1 & if E(a) \in G_s^* \\ 1 & if E(a) \notin G_s^* \\ 0 & if E(a) = \varnothing \end{cases} \tag{10}$$

if a deletes an edge. In the following, we derive this equality. For a simplified notation, we write E_G for the set of edges in G. We start by decomposing the difference in its set-theoretic components.

$$\begin{aligned} SHD(G_s^*, \hat{G}_s) - SHD(G_s^*, \hat{G}_{s'}) = & \mid E_{G_s^*} \setminus E_{\hat{G}_s} \mid + \mid E_{\hat{G}_s} \setminus E_{G_s^*} \mid \\ & - \mid E_{G_s^*} \setminus E_{\hat{G}_{s'}} \mid - \mid E_{\hat{G}_{s'}} \setminus E_{G_s^*} \mid \end{aligned}$$

We now distinguish the two cases of adding and deleting, and edge. In these cases, the terms become:

Case 1; an edge is added *to* $\hat{G}_s$*:* Then

$$\mid E_{G_s^*} \setminus E_{\hat{G}_s} \mid - \mid E_{G_s^*} \setminus E_{\hat{G}_{s'}} \mid = \begin{cases} 1 & if e \in E_{G_s^*} \\ 0 & if e \notin E_{G_s^*} \end{cases} \tag{11}$$

and

$$\mid E_{\hat{G}_s} \setminus E_{G_s^*} \mid - \mid E_{\hat{G}_{s'}} \setminus E_{G_s^*} \mid = \begin{cases} 0 & if e \in E_{G_s^*} \\ -1 & if e \notin E_{G_s^*} \end{cases} \tag{12}$$

Finally, considering the case where the edge was not manipulated ($E(a) = \varnothing$), we add Equations (11) and (12) and arrive at Equation (9).

Case 2; an edge is removed *from* $\hat{G}_s$*:* In this case, the results in Equation (11) and (12) are multiplied by -1. After again adding the inverted parts, we arrive at Equation (10).

B HYPERPARAMETERS

Table 3 describes the main hyperparameters that we used throughout this paper. For MCD [28], we additionally used the default hyperparameters that were used in the original paper.

C HARDWARE REQUIREMENTS

We ran training and evaluation on the DAS-6 compute cluster [2]. At inference time, CORE generates a graph estimate in about 18-34 milliseconds, depending on the size of the policy network and the episode length.

For training, CORE took approximately 51min, 1h40min, 2h5min, 23h, 30h to train for 3, 4, 5, 8, 10 variables, respectively. For the 3, 4, and 5 variable case these are the results on a 24 core machine with an RTX4000 GPU, for 8 and 10 variables the results are on a 48 core A100 machine.

		shared layers	shared layers size	policy layers	policy layers size	episode length	total training steps
3 variables	MCD	1 LSTM	64	2	128	10	2000000
	CORE	0	-	3	128	5	2000000
4 variables	MCD	1 LSTM	64	2	128	16	3500000
	CORE	0	-	3	128	8	3500000
5 variables	MCD	1 LSTM	128	2	256	20	4500000
	CORE	0	-	3	256	10	4500000
8 variables	MCD	-	-	-	-	-	-
	CORE	0	-	2	1024	12	45000000
10 variables	MCD	-	-	-	-	-	-
	CORE	0	-	3	1024	15	90000000

Table 3: The hyperparameters that we used to obtain our results. These parameters are the same for all experiments we performed in this paper.