Causal Discovery via Adaptive Agents in Multi-Agent and Sequential Decision Tasks

Matteo Ceriscioli
Oregon State University
Corvallis (OR), USA
ceriscim@oregonstate.edu

Karthika Mohan
Oregon State University
Corvallis (OR), USA
karthika.mohan@oregonstate.edu

ABSTRACT

Understanding the connection between robustness to distribution shifts and learning the causal model of an environment is an important area of study in AI. While previous work has established this link for single agents in unmediated decision tasks, many real-world scenarios involve mediated settings where agents influence their environment. We demonstrate that agents capable of adapting to distribution shifts can recover the underlying causal structure even in these more dynamic settings. Our contributions include an algorithm for learning Causal Influence Diagrams (CIDs) using optimal policy oracles, with the flexibility to incorporate prior causal knowledge. We illustrate the algorithm's application in a mediated single-agent decision task and in multi-agent settings. We show that the presence of a single robust agent is sufficient to recover the complete causal model and derive optimal policies for all the other agents operating in the same environment. We also demonstrate how to apply these results to sequential decision-making tasks modeled as Partially Observable Markov Decision Processes (POMDPs).

KEYWORDS

Causal Discovery, POMDP, Multi-agent systems

1 INTRODUCTION

Understanding causal relationships is fundamental to developing AI systems that can robustly adapt to changing environments [12]. While traditional machine learning approaches excel at pattern recognition within fixed distributions, they often struggle when faced with distribution shifts or interventions that alter the underlying system dynamics. This problem has been extensively studied through diverse methodologies including domain adaptation [4], transfer learning [11, 21], federated learning [10], and transportability [13], each addressing distinct flavors of the problem. Modern AI systems are expected to meet several key requirements, including robustness to distribution shifts, reliable generalization, transparent decision-making processes, and avoidance of unintended consequences [1, 7]. Causal models offer a powerful framework that addresses these challenges by providing a formal representation of the mechanisms governing an environment [12]. This approach enables agents to generalize more effectively by understanding the underlying causal relationships that persist across different scenarios [16], and it enhances system explainability by supporting both interventional and counterfactual reasoning [3].

Proc. of the Adaptive and Learning Agents Workshop (ALA 2025), Avalos, Aydeniz, Müller, Mohammedalamen (eds.), May 19 – 20, 2025, Detroit, Michigan, USA, ala-workshop.github.io. 2025.

Recent work [15] has demonstrated that agents capable of adapting to distribution shifts must implicitly learn their environment's causal structure. However, these results focus on single-agent, non-sequential tasks, and rely on the strong assumption of no mediation [12], meaning the agent's actions cannot have an effect on the utility via environment states. In contrast, many real-world AI applications involve tasks where mediation exists. For example, an autonomous car navigating from point A to point B, may affect lane occupancy and, in turn, traffic flow and the behavior of other drivers. Similarly, a robot in an industrial plant might interact with tools, move through space, and transform products to complete its task.

This work makes the following key contributions:

(1) We extend the theoretical understanding of causal discovery through robust agents by demonstrating that the assumption of unmediated tasks is unnecessary [1].
(2) We present an algorithm to learn the Causal Influence Diagram (CID) describing mediated decision tasks, by querying optimal policy oracles. We outline how to incorporate prior knowledge into the causal model (Section 3.1).
(3) We offer insights into the implications of our findings for multi-agent environments (Section 3.3).
(4) We explain how this algorithm can be applied to sequential decision tasks modeled with Partially Observable Markov Decision Processes (POMDPs) (Section 3.4).

2 PROBLEM SETUP

Our goal is to recover the causal structure and the Conditional Probability Tables (CPTs) of the variables describing the environment in which the agents operate. This environment consists of both observable variables and hidden latent variables. We define a set of interventions modeling distribution shifts, and to learn the causal graph, we query an optimal policy oracle associated with one agent to get the optimal policy for that agent under the specified distribution shift. Observe that this setup is unsuitable for traditional causal discovery algorithms like PC [19] and FCI [18] because we do not have access to the joint probability distribution of the variables or any sample data.

To model the causal relationships in the environment, we use Causal Influence Diagrams (CIDs) [5, 6]. Similar to Influence Diagrams [8], CIDs are commonly used to reason about decision-making tasks. CIDs further assume that the graph encodes the causal relationships between the nodes. We denote the set of parents of a node X as

[1]The full proof with details can be found in the supplementary material.

Pa_X, the set of children as Ch_X, the set of ancestors as Anc_X, the set of descendants as $Desc_X$ and instantiations of random variables in lower-case.

Definition 1 (Causal influence diagram [5, 6])**.** A *causal influence diagram* (CID) is a Causal Bayesian Network $M = (G = \{V, E\}, P)$, where P is a joint probability distribution compatible with the conditional independences encoded in G. The variables in V are partitioned into decision, utility, and chance variables, $V = (D, U, C)$. Each utility node U_i is associated with a real function f_i of its parents $f_i : \mathrm{Im}(Pa_{U_i}) \to \mathbb{R}$.

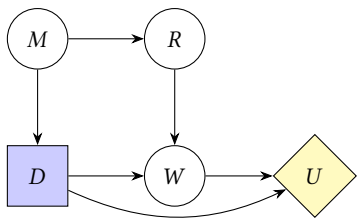


Figure 1: A CID that represents a mediated decision task where the set of utility nodes is U = $\{U\}$, the set of chance nodes is C = $\{M, R, W\}$, and the set of decision nodes is D = $\{D\}$. In this example, we have a sprinkler controlled by an AI agent. M represents the month of the year, R corresponds to rain, and W is 1 if the grass is wet and 0 otherwise. The utility node U is associated with a utility function that rewards the agent if the grass is wet but penalizes sprinkler usage. To compute the utility we only need to know the decision D and the state of W, this is why $Pa_U = \{D, W\}$.

Each agent may correspond to a different set of decision nodes and have access to a distinct subset of observable variables. A variable observed by one agent may be latent for another. Additionally, considering a situation where an agent takes more than one decision, the set of variables that it observes when it takes one decision can differ from the one that it observes when taking another decision. An example of CID can be found in Figure 1. In this example, an AI agent controls a sprinkler (D) with the goal of keeping the grass wet ($W = 1$). The agent is aware of the current month ($M \to D$), which influences the probability of rain ($M \to R$), and rain in turn affects whether the grass becomes wet ($R \to W$). There is a utility function associated with the utility node U, which depends on both the grass's wetness ($W \to U$) and the sprinkler's usage ($D \to U$). The utility function rewards the agent for achieving $W = 1$ while discouraging sprinkler activations to avoid wasting water when the grass is already wet because of the rain.

An important concept in this setting is domain dependence [15]. Intuitively, domain dependence means that for the tasks and environments we consider, no single policy π can be optimal across all possible distribution shifts.

Definition 2 (Domain dependence [15])**.** There exists $P(C = c)$ and $P'(C = c)$ compatible with M such that $\pi^* = \arg\max_\pi \mathbb{E}^\pi_P[U] \implies \pi^* \neq \arg\max_\pi \mathbb{E}^\pi_{P'}[U]$.

Domain dependence is significant because when it holds it excludes trivial cases where the optimal policy remains unchanged under any distribution shift. In such scenarios, any optimal agent operating under a certain distribution remains optimal under any shift, so optimality automatically implies robustness. In these cases, the agent does not need to learn a causal model of the environment to be robust against distribution shifts.

Following the work of [15], we represent distribution shifts as mixtures of local interventions. Given a random variable X with $x_1, \ldots, x_n$ as possible observable values, a local intervention on X is a function $\sigma : x_i \mapsto f(x_i)$ that maps each observable value x_i to a new observable value $f(x_i)$. In other words, local interventions deterministically reassign a random variable's outcomes independently of other variables.

Definition 3 (Local intervention [15])**.** *Local intervention* σ on X involves applying a map to the states of X that is not conditional on any other endogenous variables, $x \mapsto f(x)$. We use the notation $\sigma = do(X = f(x))$ (variable X is assigned the state $f(x)$). Formally, this is a soft intervention on X that transforms the conditional probability distribution as:

$$P(x \mid \mathbf{pa}_X; \sigma) \coloneqq \sum_{x' : f(x') = x} P(x' \mid \mathbf{pa}_X) \tag{1}$$

In general, a local intervention has limited capacity to model distribution shifts. For instance, it cannot model the shift from a coin that always lands on heads to a fair coin because a local intervention must deterministically map the observable value 'head' to another observable value. Therefore, we now report the concept of a mixture of local interventions [15]. This mixture is a convex combination $\sigma^* = \sum_i p_i \sigma_i$ of local interventions σ_i, where each coefficient p_i represents the probability that σ_i is used to map the observable value for X.

Definition 4 (Mixture of interventions [15])**.** A *mixture of interventions* $\sigma^* = \sum_i p_i \sigma_i$ for $\sum_i p_i = 1$ performs intervention σ_i with probability p_i. Formally, $P(x \mid pa_x; \sigma^*) = \sum_i p_i P(x \mid pa_x; \sigma_i)$.

As an example of how a mixture of local interventions can represent a distribution shift, consider the following: we have a random variable X representing the outcome of a biased coin flip that always lands on heads, where $X \in \{H, T\}$ corresponds to heads and tails, respectively. Let $\sigma_i \coloneqq do(X = i)$ with $i \in \{H, T\}$, then define $\sigma^* = \sum_i p_i \sigma_i$. By changing the coefficients p_i, we can map the distribution of the fair coin to any distribution on the observable values set $\mathrm{Im}(X) = \{H, T\}$. For example, by setting $p_H = \frac{2}{3}$ and $p_T = \frac{1}{3}$, we can map the original distribution to a new one where heads is observed $\frac{2}{3}$ of the time and tails is observed $\frac{1}{3}$ of the time. Note that in this example, each local intervention was a hard intervention because, regardless of the value of the coin, each intervention mapped it to a specific value. In general, this is not required, as a local intervention can be any deterministic map from the set of observable values to itself.

Observe that there are local interventions that, unlike hard interventions, do not make a variable independent of its parents. For example, consider an unbiased coin flip Y and a die throw X. If the coin lands on heads, we throw a six-sided die, if it lands on tails,

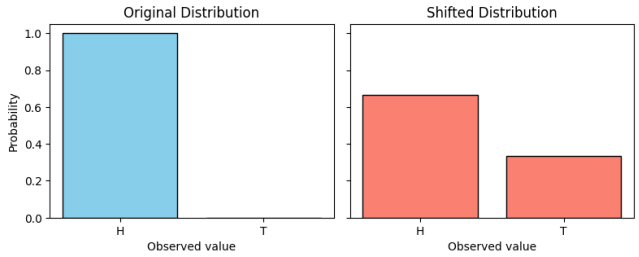


Figure 2: Histograms illustrating the distribution shift induced by a mixture of local interventions. The left histogram shows a biased coin distribution $P(X)$, where the coin always lands on heads. The right histogram represents a shifted distribution $P(X \mid \sigma^*)$ obtained by applying the mixture $\sigma^* = p_H\sigma_H + p_T\sigma_T$ for $\sigma_i \coloneqq do(X = i)$ with $p_H = \frac{2}{3}$ and $p_T = \frac{1}{3}$.

we throw a twelve-sided die. In this scenario, Y and X are clearly dependent and in the corresponding causal graph $Y \in Pa_X$. Now consider the local intervention $\sigma_m \coloneqq do(X = X \mod 12 + 1)$. This intervention increases the die result by one or sets it to 1 if the result was 12. This local intervention does not make X independent of Y because, for instance:

$$P(X = 4|Y = T; \sigma_m) = P(X = 3|Y = T) = \frac{1}{12} \tag{2}$$

However,

$$\frac{1}{12} \neq \frac{1}{8} = P(X = 3) = P(X = 4|\sigma_m) \tag{3}$$

We use optimal policy oracles to formalize the agent's understanding of optimal behavior under distribution shifts. Let D be a decision variable with observable values $d \in \mathrm{Im}(D)$, given a set of interventions Σ, an optimal policy oracle is a map $\Pi^*_\Sigma : \sigma \mapsto \pi_\sigma(d \mid pa_D)$ for $\sigma \in \Sigma$, where $\pi_\sigma(d \mid pa_D)$ is the optimal policy under the distribution shift induced by the intervention σ.

Definition 5 (Policy oracle)**.** A policy oracle for a set of interventions Σ is a map $\Pi^*_\Sigma : \sigma \mapsto \pi_\sigma(d \mid pa_D)\ \forall\, \sigma \in \Sigma$, where $\pi_\sigma(d \mid pa_D)$ is an optimal policy under the intervention σ.

In our work, we rely on Algorithm 1 from [15], which takes as input a utility function U, an optimal policy oracle, an intervention $\sigma \in \Sigma$, and a parameter N that controls the number of samples. For any local intervention $\sigma \in \sigma_Y$, let d be the deterministic optimal decision under the shift induced by σ. By domain dependence, there exists a hard intervention σ' such that d is no longer optimal. Let d_2 be the deterministic optimal decision under σ'. Considering the mixture $\sigma(q) \coloneqq q\sigma + (1 - q)\sigma'$, there exist a value q_{crit} for q such that d_2 and another decision d_1 are both optimal. The algorithm returns q_{crit}, d_1, and d_2.

3 MAIN RESULTS

In this section, we describe an algorithm for learning the CID using an optimal policy oracle. In Section 3.2, we show this algorithm can be applied to learn a simple environment with a single agent. Subsequently, in Section 3.3, we demonstrate how to adapt this approach to handle multi-agent environments. Finally, in Section 3.4 we discuss the relation between CIDs and POMDPs and explain how to apply our causal discovery algorithm to learn a causal model for POMDPs.

3.1 LearnCID Algorithm

Under specific assumptions detailed below, the LearnCID algorithm enables the reconstruction of the underlying causal model by identifying the CID structure and the CPTs for the variables corresponding to chance nodes. The correctness proof for Algorithm 1 (LearnCID) can be found in the supplementary material. The LearnCID algorithm operates under the following assumptions:

Assumption 1. Given the CID $M = (G = \{V, E\}, P)$ with $V = (D, U, C)$, the set of nodes and the partition (D, U, C) is known.

The set of nodes together with the node partition (D, U, C) is known, therefore we know all variables in the system and the type of each node (decision, utility, or chance).

Assumption 2. The CID is faithful [20] and sufficient [12].

Faithfulness implies that every conditional independence encoded in the graph G also holds in the joint probability P. A set of variables in a causal model is sufficient when it includes all common causes.

Assumption 3. The CID contains exactly one decision node D and one utility node U.

Despite Assumption 3, this algorithm can be applied to multi-decision CIDs. In Section 3.3 we explain how a CID containing multiple decision nodes can be converted to a single-decision CID to recover the full causal structure and derive optimal policies for all the decision nodes. For CIDs with multiple utility nodes, we can select one utility node and prune the others. Note that the optimal policy oracle depends on both the specific CID and decision node, so different utility node selections generally correspond to different optimal policy oracles. After selecting a utility node, we can also prune all chance nodes that are not ancestors of U.

Assumption 4. The Markov blanket of decision node D is known. We also know all the edges between these nodes. The CPTs of chance nodes that are children of D are known.

Motivations for Assumption 4 can be found in the supplementary material.

Assumption 5. D is a parent of U.

Assumption 6. All chance nodes are ancestors of U.

Chance nodes that are not ancestors of U (and consequently, since $D \in Pa_U$, not ancestors of D) have no influence on the decision task. LearnCID would ignore these nodes, and their associated causal structure and CPTs would not be recovered.

Assumption 7. The utility function f associated with the utility node U is fully specified.

The utility function's functional form is known, which tells us all the variables involved in calculating the utility. These variables appear in the causal graph as parents of the utility node.

Assumption 8. We have access to a set Σ of all possible mixtures of local interventions, along with the optimal policy oracle Π^*_Σ for decision node D.

Algorithm 1 LearnCID

Input:

- Nodes $V = \{\{D\}, \{U\}, C\}$
- Known set of edges $\hat{E}$
- Set of chance nodes with all known parents V_{kwn}
- Number of samples N to estimate q_{crit}

Output: The CID's structure E', and the set of CPTs P for all nodes in $C \setminus Ch_D$

```
while there are still unvisited chance nodes, starting from the parents of U that are not children of D do
    X ← Unvisited chance node with a known path to U
    Path ← Set of chance nodes on a directed internal path from X to U, or to D if U is unreachable
    C_X ← Chance nodes not in Path, and not known to be parents of X
    Z ← Pa_X if X ∈ V_kwn else Z ← C_X
    for each instantiation x of X do
        for each Y ∈ Z and each instantiation of variables c in C_X \ Y do
            σ_Y(c) ← As in Equation 5
            for each σ ∈ σ_Y do
                Estimate q_crit, d_1, d_2, pa'_U using ALGq_crit(U, Π*_Σ, N, σ)
                Compute P(X = x | pa_x; σ) using Equations 6 and 7
            if there exist two interventions σ, σ' in σ_Y such that P(x | pa_X; σ) ≠ P(x | pa_X; σ') then
                Y is a parent of X
        for each pa_X ∈ Im(Pa_X) do
            P(x | pa_X) ← P(x, pa_X; σ) for any hard intervention σ compatible with pa_X
Return the updated set of edges E', and the set of CPTs P
```

Assumption 9. There exist no decision $d^* \in \text{Im}(D)$ that is optimal for any instantiation of $Pa_U \setminus D$.

If Assumption 7 holds, we can verify Assumption 9 by testing different value combinations for the parent nodes of U. While Assumption 9 is equivalent to domain dependence (Definition 2) in unmediated decision tasks, this equivalence breaks down in the general mediated case. Nevertheless, Assumption 9 is sufficient to guarantee domain dependence.

Prior knowledge in the form of the causes of a subset of chance nodes can be provided. In particular, prior knowledge about direct causes (parents) can be specified for a subset of chance nodes, denoted as V_{kwn} i.e., for every $C_i \in V_{\text{kwn}}$, the set of parent nodes of C_i is known.

To properly formulate the algorithm, we need to introduce some concepts. We define a local intervention:

$$f(X) \leftarrow \begin{cases} x, & \text{if } X = x \\ x', & \text{otherwise} \end{cases} \tag{4}$$

where x' is an arbitrary observable value for X different from x. Let C_X represents the set of all chance nodes except X and those along a directed path from X to either U or D, we define the following family of local interventions:

$$\sigma_Y(c) \leftarrow \{do(Y = y, C_X = c, X = f(X)) \mid y \in \text{Im}(Y)\} \tag{5}$$

Let $C_1, \ldots, C_k$ be the chance nodes in a directed internal path from X to U or D. If $C_1 \in Pa_U$ let $\mathcal{C} \coloneqq \{C_1, \ldots, C_k\}$ otherwise let $\mathcal{C} \coloneqq \{C_2, \ldots, C_k\}$. For both x and x', we compute:

$$\beta(x) \coloneqq \sum_{c \in \text{Im}(C)} \prod_{i=1}^{k} P(c_i \mid pa_{C_i})[U(d_2, c) - U(d_1, c)] \tag{6}$$

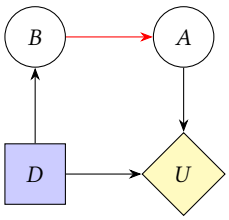
Observe that the right-hand side of Equation 6 depends on X for determining the set containing the path of chance nodes $\mathcal{C}$, and depends on the specific instantiation x of X because X is the parent of either a chance node in $\mathcal{C}$, the decision node D, or the utility node U. Using Equation 6, we can compute $P(x \mid pa_X; \sigma)$ as:

$$P(x \mid pa_X; \sigma) = \frac{(1 - \frac{1}{q_{crit}})[U(d_2, pa'_U) - U(d_1, pa'_U)] - \beta(x')}{\beta(x) - \beta(x')} \tag{7}$$

3.2 Example 1 - Single agent environment

Consider the CID in Figure 3, assume we know $B \in Ch_D$, $A \in Pa_U$, and all the variables are binary. We also know $U(d, a) \coloneqq 1$ if $d = a$ and 0 otherwise, and an optimal policy oracle Π^*_Σ where Σ is the set of all mixtures of local interventions. The CPT for A can be found in the table on the right side of Figure 3, but let us assume it is unknown. We want to use Algorithm 1 to learn whether there is an edge between B and A, and the CPT for A. For ease of comprehension we summarize the application of Algorithm 1 to the example CID in Figure 3, in the following steps:

(1) Insert unvisited chance nodes (i.e., chance nodes for which we do not know the parents or the CPT of the corresponding variable) with a known path to U into a queue, starting from the parents of U that are not children of D.



B	A	$P(A\|B)$
0	0	0.7
0	1	0.3
1	0	0.1
1	1	0.9

Figure 3: An example of a single-decision/single-utility CID. On the right, the CPT for variable A. The edge marked in red is unknown.

(2) Extract A from the queue and define local interventions on B as in Equation 5, used to obtain both the CPT for A and the set of its parents.
(3) Estimate q_{crit} using ALGq_{crit} (Algorithm 1 in Richens and Everitt [15]).
(4) Compute $P(A = a_i|do(B = b_i))$.
(5) Repeat steps 2 to 4 for all configurations of a_i and b_i.
(6) Deduce the set of parents of A and its CPT.

Following the aforementioned steps:

Step 1: In this example Pa_D is empty and $Pa_U = \{A\}$, so we start from node A. We also assume V_{knw} is empty. Observe that since A is the only chance node that is not children of D the process will stop after A. Since $A \in Pa_U$, "Path" is empty and $C_A = \{B\}$.

Step 2: Set $\sigma_0 \coloneqq do(B = 0)$, then in ALG$q_{crit}$ we use the oracle $\Pi^*_\Sigma(\sigma_0)$ to find the optimal decision $d_1 \coloneqq 0$. This is evident from the (unknown) CPT because for $B = 0$ the probability that $A = 0$ is higher than the one for $A = 1$, since the utility is an AND operation between A and D, the oracle returns the optimal decision $D = 0$ which will be equal to A more often than $D = 0$.

Step 3: We estimate q_{crit} using ALGq_{crit}. It works by finding σ' such that the optimal decision is no longer $D = 0$. Since in this example D is binary, we already know the new optimal decision must be $D = 1$, in general, for this we can use the optimal policy oracle. We can find σ' by hard intervening on the parents of the utility node U, which is A in this case, such that d_1 is no longer optimal. Specifically, we can define σ' as a hard intervention that sets A to 1, therefore the new optimal decision is $d_2 \coloneqq 1$. Mind that this is always possible thanks to Assumption 9. Then, we define the mixture of local interventions $\sigma(q) \coloneqq q\sigma_0 + (1 - q)\sigma'$. Figure 4 shows how the expected utility of both decisions varies with q. We can sample q uniformly in the interval $[0, 1]$ N times and each time query the optimal policy oracle. Each time the oracle returns an optimal decision for the intervention σ' we increment a counter θ. Then $\frac{\theta}{N}$ is an unbiased estimate for q_{crit}. For this example, $q_{crit} = \frac{5}{7}$.

Step 4: We now compute $P(A = 0|pa_A; \sigma_0)$. We start with $A = 0$ and $B = 0$, we first need to compute $\beta(A = 0)$ and $\beta(A = 1)$. This is the case where the node for which we are computing the CPT is the first on the directed path to the utility node. So the beta expressions

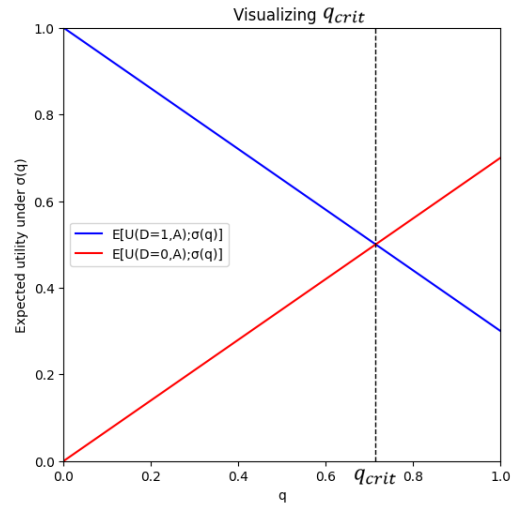


Figure 4: Following Example 1, let us examine two interventions: $\sigma_0 \coloneqq do(B = 0)$ with optimal decision d_1, and a hard intervention σ' with optimal decision d_2 where $d_1 \neq d_2$. We define a mixture of local interventions as $\sigma(q) = q\sigma_0 + (1-q)\sigma'$. The plot displays the expected utility of both decisions as q varies. The point where both decisions become simultaneously optimal is called q_{crit}.

are:

$$\beta(A = 0) = U(d_2, 0) - U(d_1, 0) = -1 \tag{8}$$
$$\beta(A = 1) = U(d_2, 1) - U(d_1, 1) = 1 \tag{9}$$

Following Equation 7:

$$P(A = 0|B = 0; \sigma) = \frac{(1 - \frac{7}{5})[U(d_2, 1) - U(d_1, 1)] - \beta(a')}{\beta(a) - \beta(a')} \tag{10}$$
$$= \frac{\frac{2}{5}[U(1, 1) - U(0, 1)] + 1}{2} = \frac{7}{10} \tag{11}$$

Which corresponds to the value in the table of Figure 3.

Step 5: We can repeat the same procedure for $\sigma \coloneqq do(B = 1)$ and find that $P(A = 0|do(B = 1)) = P(A = 0|B = 1) = 0.1$. The equivalence between intervention and conditioning follows the specific family of interventions we are using (i.e., hard interventions on all nodes that are not on the directed path to the utility node).

Step 6: Since $P(A = 0|do(B = 0)) \neq P(A = 0|do(B = 1))$, we can conclude that B is a parent of A. In the general case, this approach ensures that B is not just an ancestor but indeed a parent of A, because the intervention blocks all other paths from B to A. Thus, $P(A = 0|do(B = 0), pa_A)$ would equal $P(A = 0|pa_A)$ if B were not a parent.

As expected, this process allows us to learn both the correct graph structure and the CPT for A (and, more generally, for all chance nodes that are not children of D).

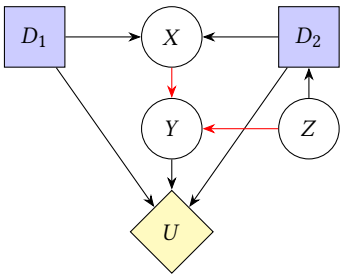


Figure 5: A multi-decision CID that represents an environment where two agents cooperate to maximize the utility U. The edges marked in red are unknown. Example 2 demonstrates how to adapt this CID to apply Algorithm 1 and recover the missing edges and CPTs for chance nodes.

3.3 Example 2 - Cooperative multi-agent environment

Examine the multi-decision CID in Figure 5. It represents a cooperative game between two agents, each controlling a distinct decision variable. Both agents aim to maximize a shared utility function, U, and operate in different contexts defined by the parent sets of their decision nodes ($Pa_{D_1} = \emptyset \neq \{Z\} = Pa_{D_2}$). Similarly to before, we assume knowledge of the children for the decision node D_1, their CPTs, their parents, and the utility function associated with the node U.

Overview of Methodology. Even if LearnCID can be applied only to single-decision CIDs, we propose two approaches to handle multi-decision CIDs. In both cases having access to an optimal policy oracle for one decision will let us recover the full causal structure and the CPT of all chance nodes. For both approaches there is one decision node that we will refer to as the primary decision node, which follows the same assumptions as the single-decision case, i.e., we assume to know its children, their CPTs, and their parents. We will refer to the other decision nodes as secondary decision nodes. The first approach requires knowing the parent sets of all secondary decision nodes. Then, we define a faithful policy for each of these nodes. This is feasible because we know the parent set for each decision node. Then we treat these nodes as chance nodes by using their policies as CPTs. With an optimal policy oracle for the primary decision node, we can apply Algorithm 1 to learn the CID structure and CPTs for all chance nodes except the children of the primary decision node. The second approach does not require knowledge of secondary decision nodes' parent sets, Instead, it assumes that each of these nodes corresponds to a fixed, unknown, faithful policy. Finally, we assume the availability of an optimal policy oracle and apply Algorithm 1 to recover the full CID. Once the CPTs for all chance nodes are learned, it becomes possible to determine the optimal policy for every decision node in the original graph under any distribution shift [2, 17].

Example Analysis. Let Π^*_Σ be the optimal policy oracle for D_1 and $\pi(D_2 \mid Z)$ be any given policy that governs D_2. The nodes for which we need to learn the parents are Y and Z. Node Y's potential parents are X and Z, whereas node Z's only potential parent is Y. Using Algorithm 1, we determine parental relationships as follows: consider the instantiation $Y = 0$. With Algorithm 1 we can compute $P(Y = 0 \mid pa_Y; \sigma_0)$ and $P(Y = 0 \mid pa_Y; \sigma'_0)$ using $\sigma_0 := do(X = 0, Z = 0)$, and $\sigma'_0 := do(X = 0, Z = 1)$ respectively. We observe that these two probabilities are equal. We repeat this process with $\sigma_1 := do(X = 1, Z = 0)$ and $\sigma'_1 := do(X = 1, Z = 1)$, and again, $P(Y = 0 \mid pa_Y; \sigma_1) = P(Y = 0 \mid pa_Y; \sigma'_1)$. Performing the same procedure for $Y = 1$, we find that all pairs of interventions yield the same probabilities. Therefore, Z is not a parent of Y. Next, we check whether X is a parent of Y. Comparing $P(Y = y \mid pa_Y; \sigma_0)$ with $P(Y = y \mid pa_Y; \sigma_1)$, and $P(Y = y \mid pa_Y; \sigma'_0)$ with $P(Y = y \mid pa_Y; \sigma'_1)$ for all $y \in \{0, 1\}$, we find that at least one of these pairs of probabilities differs. Given the faithfulness of the CID. This confirms that X is a parent of Y. Finally, we consider Z. Since the only potential parent of Z was Y, and Y was found to be a child of X, Y can not be a child of Z because this would introduce a cycle in the graph. Z has no other potential parents and therefore we have learned the full CID.

Algorithmic Complexity. First consider that the complexity of LearnCID depends on the complexity of querying the policy oracle, let us call this complexity K. As a worst-case scenario there is no prior knowledge about the graph, so V_{knw} is empty. Let n be the number of variables, and $b := \max_{X \in C} |\operatorname{Im}(X)|$ be the maximum number of observable values of any chance variable. The most computationally expensive steps correspond to computing q_{crit} with $ALG_{q_{crit}}$ (line 10) and the CPTs' entries for all the chance nodes. Each call to $ALG_{q_{crit}}$ costs $O(N(K + |\operatorname{Im}(D)|))$, filling one CPT's entry costs $O(nb^n)$. Overall the algorithm's time complexity is $O(n^2 b^n N(K + |\operatorname{Im}(D)|) + n^3 b^{2n})$.

3.4 Applying LearnCID to Partially Observable Markov Decision Processes

In this section, we demonstrate that under mild assumptions, given a Partially Observable Markov Decision Process (POMDP) [9] with unknown state-transition function T and states described by a finite set of discrete variables, the availability of an optimal policy oracle allows Algorithm 1 to be applied to learn both the intra- and inter-temporal causal structure, as well as the CPTs of all variables and the state-transition function.

Let $M = (S, \mathcal{A}, T, R, \Omega, O, \gamma)$ be a POMDP where S is the set of states, A the set of actions/decisions, $T : S \times A \to \Pi(S)$ is the state-transition function where $\Pi(S)$ is the set of probability distributions over the set of states, $R : S \times A \to \mathbb{R}$ is a reward function, Ω is the set of observables (values of observable variables), O is the set of conditional observation probabilities, and $\gamma \in [0, 1)$ is the discount factor.

We assume time-homogeneity, meaning the transition probabilities do not change over time. Moreover, we assume the set of observable variables is fixed and the agent gets an observation from these variables at every timestep. Formally, each state s in S is described by a finite set of variables V (i.e., there exists a bijective function $f : \operatorname{Im}(V) \to S$), there exists a subset of observable variables $\mathbf{V_o} \subset \mathbf{V}$, and for every state s in S and every action a in A it holds that $O(s, a, v_o) = 1$, where v_o is the instantiation of the observable variables at the current state. Then we can model the state

space of a POMDP using a CID. Mind that under our assumptions about the set of conditional observation probabilities, if the set of observable variables coincides with the set of all state variables then we get a Markov Decision Process (MDP) as a special case of a POMDP. By identifying the observable variables of the POMDP as the parents of decision nodes, the reward function as the utility function associated with a utility node, and all other unobserved variables as chance nodes that are not parents of decision nodes, we can observe that a POMDP can be seen as a CID unrolled over time.

The mapping from a POMDP to a CID with unknown structure is described in Algorithm 2. Since we want to apply Algorithm 1 to this CID, we further need to assume each variable is discrete, the POMDP reward directly depends on the action (equivalent to $D \in Pa_U$). In this case, we have a slightly weaker condition for the Markov blanket of the decision node: other than the parents of D, we only need to know which chance nodes are children of D at the same timestep, their CPTs, and their parents, while we can ignore the chance nodes that are children of D in the following timesteps (e.g., considering the POMDP described in Figure 6, we do not need to know that X_{t+1} is a child of D_t because they belong to different timesteps). Due to the Markov property of POMDPs, direct causal relationships are limited to occurring either within a single timestep or from variables at one timestep to those at the next. This temporal constraint provides an advantage over standard LearnCID applications: when analyzing POMDPs, we only need to identify the decision node's children within the same timestep, and can safely ignore children in subsequent timesteps.

Algorithm 2 POMDPtoCID

Input: POMDP $(S, \mathcal{A}, T, R, \Omega, O, \gamma)$ with unknown state-transition function T, and states described by a finite set of variables $\mathbf{V}$.
Output: Correspondent CID's graph G without causal arcs involving chance nodes.

Add decision node D_t to CID's graph G with $\mathcal{A}$ as set of decisions.
Add chance node D_{t-1} to CID's graph G.
for each variable $V \in \mathbf{V}$ **do**
    Add chance node corresponding to random variable V_t and V_{t-1} to CID's graph G.
    if $V \in \mathbf{V_o}$ **then**
        Add edge (V_t, D_t) and (V_{t-1}, D_{t-1}) to CID's graph G.
Define $\mathcal{U} : \mathrm{Im}(V) \times \mathrm{Im}(D) \to \mathbb{R}$ as $\mathcal{U}(v, d) \mapsto \mathcal{R}(f(v), d)$ for all $d \in \mathrm{Im}(D)$ and $v \in \mathrm{Im}(V)$
Add utility node U_t with utility function $\mathcal{U}$ to G.
Add edge (D_t, U_t) with utility function $\mathcal{U}$ to G.
Return CID's graph G.

We can run LearnCID on the CID produced by Algorithm 2 to learn the causal relationships for variables both at the same time step and at different ones. The algorithm maps the variables, decisions, and rewards related to two consecutive timesteps to nodes of a CID. We now refer to these timesteps as $t-1$ and t. Therefore, for each variable X partially describing the state of the POMDP, this newly defined CID contains two variables X_{t-1} and X_t. Similarly, there will be two decision nodes D_{t-1} and D_t and two utility nodes U_{t-1} and U_t. Observe that we can prune U_{t-1}, and, similarly to what we proposed for multi-agent settings, we convert D_{t-1} to a chance node and assign any faithful policy as its CPT. This is possible because the chance nodes that are parents of decision nodes correspond to the observable variables which are known. Then, if the assumptions of LearnCID are satisfied, including having an optimal policy oracle available, then we can use it to learn all the missing causal relationships. For time-homogeneous POMDP, like in the example illustrated in Figure 6, this is sufficient to learn all the intra- and inter-temporal causal relationships of variables in $Anc_{U_t} \cup Anc_{D_t}$ for any timestep since they do not change when varying t because this would imply a change in the transition probability which contradicts the time-homogeneity assumption.

Learning the state-transition function. Once we have learned the CPTs for all the chance nodes and we have fixed a policy π for the agent, it is possible to recover the state-transition function T, i.e., for all states $s \in S$ and actions $a \in A$ we can find a probability distribution over the state of the process at the following timestep. Let $V_1^{(t)}, \dots, V_n^{(t)}$ be the variables $\mathbf{V}$ associated with the chance nodes at timestep t do that we observe that:

$$\begin{aligned} T(s^{(t-1)}, a^{(t-1)}) &= P(s^{(t)} | s^{(t-1)}, a^{(t-1)}) \\ &= P(V_1^{(t)}, \dots, V_n^{(t)} | v_1^{(t-1)}, \dots, v_n^{(t-1)}, a^{(t-1)}) \\ = \prod_{i | V_i \in Ch_D} \sum_{d \in \mathcal{A}} P(V_i^{(t)} | pa_{V_i}, d) \pi(d | pa_D) &\prod_{i | V_i \in \mathbf{V} \setminus Ch_D} P(V_i^{(t)} | pa_{V_i}) \end{aligned} \tag{12}$$

Since all the CPTs are known and the policy is fixed, we can compute the state-transition function for all states and actions.

Non time-homogeneous case. For non-time-homogeneous POMDPs, applying this approach directly only allows us to learn the causal relationships for two consecutive timesteps, since the time-dependance of the state-transition function may modify the mechanisms governing the interactions between variables at different timesteps. For instance, suppose each timestep represents a day of the year, and an agent controls an irrigation system for a farm. During the dry season, activating the irrigation system directly increases soil moisture, but during the rainy season, rainfall already saturates the soil, making the irrigation system's effect negligible. Applying Algorithm 2 followed by Algorithm 1 to two consecutive rainy-season days would suggest that irrigation and soil moisture are independent, a conclusion that would not hold during the dry season. One solution is to augment the state space of the POMDP to make it time-homogeneous [14]. Alternatively, to comprehensively learn the causal graph across all timesteps, we would need to apply this procedure repeatedly for each timestep. This repetition may be practical for POMDPs that are non-time-homogeneous only over a finite time period or that exhibit periodic behavior, as in the given example. For cases where time homogeneity doesn't hold across a finite number of timestep intervals, we can use an alternative approach. First, map each interval to a single CID. Then, prune all

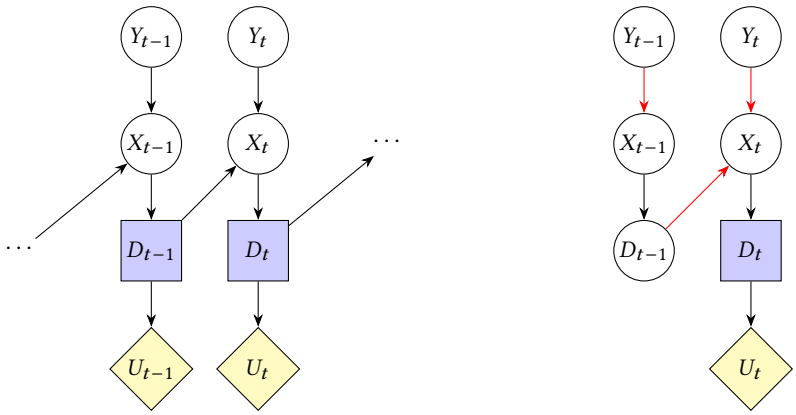


Figure 6: Example of an application of Algorithm 2 and Algorithm 1 (LearnCID) to a time-homogeneous POMDPs. On the left, a time-homogeneous POMDP where each state is described by a CID. On the right, the corresponding CID with unknown edges marked in red. To apply LearnCID we consider two timesteps $t-1$ and t, and the decision node D_t with utility node U_t. We pick any faithful policy for D_{t-1} and turn it into a chance node, we can prune U_{t-1}. Since the POMDP is time-homogeneous both the intra- and inter-temporal causal relationships are preserved for any timestep thus we learn the causal graph for the entire POMDP.

utility nodes except those corresponding to the last timestep of each interval. Finally, apply our proposed solution for multi-decision CIDs to each individual interval. However, in general, all of these approaches impose different requirements on the optimal policy oracle. Specifically, using the augmented POMDP requires an optimal policy oracle capable of handling interventions on the time variable, whereas the other approaches generally require a distinct optimal policy oracle for every pair or interval of timesteps.

4 CONCLUSIONS

In this work, we addressed the challenge of understanding the relationship between robustness to distribution shifts and an agent's causal understanding of the environment in which it operates. While previous work established that robust agents encode the causal model in single-agent, unmediated tasks, we demonstrated that this connection also holds in mediated tasks and multi-agent settings. We presented an algorithm for learning CIDs using optimal policy oracles, which allows the integration of prior causal knowledge. Our results show that even for mediated tasks, where agents' actions affect the environment, it is possible to recover the underlying causal structure. Moreover, in multi-agent systems, we showed how a single robust agent enables the discovery of the complete causal model, making it possible to learn optimal policies for all the other agents under any distribution shift. Finally, we applied our approach to POMDPs, demonstrating its ability to recover intra- and inter-temporal causal relationships and the state-transition function, even in non-time-homogeneous settings. These findings provide a solid foundation for the development of AI systems that can adapt and learn in complex settings. To bridge the gap between theory and practical applications, we are actively extending our research to approximate settings, where regret-bounded policies are employed instead of optimal ones.

REFERENCES

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete Problems in AI Safety. (06 2016). https://doi.org/10.48550/arXiv.1606.06565

[2] Elias Bareinboim, Juan D. Correa, Duligur Ibeling, and Thomas Icard. 2022. *On Pearl's Hierarchy and the Foundations of Causal Inference* (1 ed.). Association for Computing Machinery, New York, NY, USA, 507–556. https://doi-org.oregonstate.idm.oclc.org/10.1145/3501714.3501743

[3] Elias. Bareinboim, Junzhe Zhang, and Sanghack Lee. 2024. *Towards Causal Reinforcement Learning*. Technical Report R-65. Causal Artificial Intelligence Lab, Columbia University.

[4] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *Proceedings of the 19th International Conference on Neural Information Processing Systems* (Canada) *(NIPS'06)*. MIT Press, Cambridge, MA, USA, 137–144.

[5] Tom Everitt, Ryan Carey, Eric D. Langlois, Pedro A. Ortega, and Shane Legg. 2021. Agent Incentives: A Causal Perspective. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 13 (May 2021), 11487–11495. https://doi.org/10.1609/aaai.v35i13.17368

[6] David Heckerman. 1995. A Bayesian approach to learning causal networks. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (Montréal, Qué, Canada) *(UAI'95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 285–295.

[7] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2021. Unsolved Problems in ML Safety. https://doi.org/10.48550/arXiv.2109.13916

[8] Ronald A. Howard and James E. Matheson. 1984. Influence Diagrams. *Readings on the Principles and Applications of Decision Analysis, Vol. II* (1984).

[9] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101, 1 (1998), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

[10] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*. https://arxiv.org/abs/1610.05492

[11] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[12] Judea Pearl. 2009. *Causality: Models, Reasoning and Inference* (2nd ed.). Cambridge University Press, USA.

[13] Judea Pearl and Elias Bareinboim. 2011. Transportability of Causal and Statistical Relations: A Formal Approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*. 540–547. https://doi.org/10.1109/ICDMW.2011.169

[14] Martin L. Puterman. 2005 - 1994. Markov decision processes : discrete stochastic dynamic programming.

[15] Jonathan Richens and Tom Everitt. 2024. Robust agents learn causal world models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=pOoKI3ouv1

[16] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. 2021. Toward Causal Representation Learning. *Proc. IEEE* 109, 5 (2021), 612–634. https://doi.org/10.1109/JPROC.2021.3058954

[17] Ross D. Shachter. 1986. Evaluating Influence Diagrams. *Oper. Res.* 34, 6 (Dec. 1986), 871–882.

[18] Peter Spirtes. 2001. An Anytime Algorithm for Causal Inference. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. R3)*, Thomas S. Richardson and Tommi S. Jaakkola (Eds.). PMLR, 278–285. https://proceedings.mlr.press/r3/spirtes01a.html Reissued by PMLR on 31 March 2021.

[19] Peter Spirtes and Clark Glymour. 1991. An algorithm for fast recovery of sparse causal graphs. *Social science computer review* 9, 1 (1991), 62–72.

[20] Peter Spirtes, Clark Glymour, N Scheines, et al. 1991. Causation, Prediction, and Search. (1991).

[21] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* 109, 1 (2021), 43–76. https://doi.org/10.1109/JPROC.2020.3004555