Finding Invariant Predictors Efficiently via Causal Structure

Kenneth Lee[1] **Md Musfiqur Rahman**[1] **Murat Kocaoglu**[1]

[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana, USA

Abstract

One fundamental problem in machine learning is out-of-distribution generalization. A method named the surgery estimator incorporates the causal structure in the form of a directed acyclic graph (DAG) to find predictors that are invariant across target domains using distributional invariances via Pearl's do-calculus. However, finding a surgery estimator can take exponential time as the current methods need to search through all possible predictors. In this work, we first provide a graphical characterization of the identifiability of conditional causal queries. Next, we leverage this characterization together with a greedy search step to develop a polynomial-time algorithm for finding invariant predictors using the causal graph. Given the correct causal graph, our method is guaranteed to find at least one invariant predictor, if it exists. We show that our proposed algorithm can significantly reduce the run-time both in simulated and semi-synthetic data experiments and have predictive performance that is comparable to the existing work that runs in exponential time.

1 INTRODUCTION

One fundamental challenge in machine learning (ML) is to deploy an algorithm that generalizes well to unseen data. When the training data distribution and the target distribution differ, i.e., a distribution shift occurs, ML algorithms can make mistakes that have serious consequences in mission-critical applications in areas such as healthcare [21, 30]. Thus, an important goal in ML is to carefully select the features that can be used to train predictive algorithms that perform well in new environments.

There have been numerous studies to investigate distribution shifts using different tools. [6, 26] evaluates their predictor performance under mixture covariate shifts by modeling it as a distributionally robust optimization (DRO) problem([3]). In this approach, they consider a lower bound of the proportion of the minority sub-population from a mixture model and minimize their worst-case subpopulation loss. Another line of work deals with the distribution shift ([14, 19]) by developing learning models that are stable against shifts due to changes in the data-generating mechanisms. Researchers have considered the causal connection between features ($\mathbf{X}$) and the target variable (Y) to introduce methods to deal with different types of distribution shifts. Some examples include covariate shift where $P(\mathbf{X})$ changes ([9, 15]), target shift where $P(Y)$ changes ([31, 7]) and conditional shift where $P(Y|\mathbf{X})$ changes ([8]).

There are mainly two types of stable training algorithms in the literature that consider different forms of distribution shifts: reactive and proactive ([4]). Reactive approaches consider datasets from the deployment environment to train and adjust their training algorithm accordingly by re-weighting the training data so that it performs better in the deployment environment [27, 9]. However, in many sensitive applications, we do not have access to every possible domain dataset. Under these circumstances, proactive approaches are preferable, as they are trained without any deployment data and prepared to perform well for any possible distribution shift [28, 18]. Proactive algorithms train with stable information/features that are invariant across environments.

Some recent proactive algorithms such as [16, 10] find optimal conditioning sets containing causal, anti-causal, or confounded dependence (i.e., unobserved common cause) with the target by hypothesis testing their stability across multiple data domains. This makes the prediction invariant to the specific distribution shift. Such algorithms model the causal relations among the features as a causal graph and model distribution shift via an auxiliary node that captures the shift as an intervention. This setup allows utilizing conditional independence relations to obtain stable predictors.

Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023), PMLR 216:1196–1206.

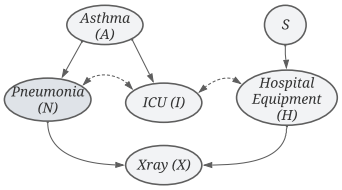


Figure 1: A causal graph representing causal relations among features. We use A, I, X, H from a hypothetical do(H) interventional distribution to predict *Pneumonia* severity such that it stays invariant to distribution shift.

Suppose we have access to the data-generating causal graph in Figure 1. The admission criteria for *ICU*(I) is caused by *Asthma (A)* having a confounding effect on *Pneumonia (N)* severity and *Hospital equipment* (H). The variable *Xray (X)* is caused by *Pneumonia* and the *Hospital equipment*. We assume the feature H is responsible for the distribution shifts and represent the shift with a discrete variable S pointing to it. For example, $S = 0$ could represent the training distribution, and $S = 1$ could represent the distribution during deployment. According to most existing algorithms in the literature ([9, 31]), we can use features A, X, and H to train a model that can predict N as valid stable predictors since they cut off any dependence from S. Along with these features, I might be a better predictor of N. Although N and H are independent for the mentioned predictors, they become dependent once we control for I in the dataset, and eventually, N becomes dependent on S. Therefore, if we wish to include I in the stable predictors, previous approaches cannot suggest any solution to achieve that without creating dependence between S and the target variable, i.e., any such predictor becomes domain-dependent.

Recently, [29] proposed an algorithm that removes the dependence on any mechanism that is sensitive to the distribution shift using hypothetical interventions to find stable predictors known as *graph surgery estimators*. Such interventions are not actually performed but simulated from the observational data via the identification algorithm [23]. In Figure 1, the query $P(N|\text{do}(H), A, I, X)$ can be uniquely calculated from training data and do(H) d-separates N from S. Thus we can train our model with predictors I, X, H, A from a hypothetical do(H) interventional distribution to predict N. However, in order to find such predictors, [29] iterates over the subsets of all variables and constructs an exponential number of conditional causal queries. Each such query requires one execution of the **ID** algorithm. This results in an exponential-time algorithm in the worst-case.

[24] proposed a solution to a similar problem in a different context, identifying the conditional independence statements in interventional distributions using only observational data. Such conditional independences are called *dormant independences*. They provide a complete algorithm for finding dormant independence between two sets of variables. Although very relevant to the surgery estimator problem, their approach cannot be directly applied to solve the causal invariant prediction problem. We establish a formal connection and propose a generalized solution to the invariant prediction problem. We provide several invariant predictors for any causal graph by starting from dormant independence.

In this paper, we propose a polynomial-time algorithm that outputs invariant predictors given the causal graph by leveraging a characterization of causal identifiability of conditional queries and systematically combining the ideas from dormant independence with a greedy feature selection step. Our algorithm is guaranteed to find at least one invariant predictor if it exists. We perform extensive experiments and the results illustrate that our algorithm gains significant computational efficiency compared to the existing work and has competitive predictive performance. Our contributions are summarized as follows:

1. We provide a graphical characterization of the identifiability of conditional causal queries and leverage it with greedy search to develop a sound algorithm called *ID4IP* for finding invariant estimators in polynomial time given the causal graph structure.
2. We show that ID4IP is sound. We also show that ID4IP outputs at least one graph surgery estimator anytime such an estimator exists.
3. We perform experiments on both synthetic and semi-synthetic data to illustrate that our algorithm has predictive performance that is comparable to a complete algorithm in the literature by [29], and outperforms it when the runtime is limited.

2 BACKGROUND

In this section, we describe the necessary definitions and background knowledge required to introduce our approach.

Definition 2.1 (Structural Causal Model (SCM) and Causal Graph)**.** An SCM is a tuple $\mathcal{M} = (\mathbf{V}, \mathbf{E}, \mathcal{N}, \mathcal{U}, \mathcal{F}, P(.))$ that contains a set of observable variables $\mathbf{V}$, a set of unobserved exogenous variables $\mathcal{N}$, a set of latent confounders $\mathcal{U}$ i.e., unobserved common causes of two observable variables, a set of functions $\mathcal{F}$ and a product probability distribution $\mathcal{P}(.)$ over $\mathcal{N}$ and $\mathcal{U}$. Each observed variable is generated as $V_i = f_i(Pa_i, E_i, U_{S_i})$, where $f_i \in \mathcal{F}$, $Pa_i \subset \mathbf{V}$, $E_i \in \mathcal{N}$ and $U_{S_i} := \{U_j : j \in S_i\}$ for some $S_i \subset \mathcal{U}$. Variables set $\mathbf{V}$ has a joint distribution $\mathcal{P}_{\mathbf{V}}$ implied by $\mathcal{F}$ and $\mathcal{P}(.)$.

An SCM induces a directed acyclic graph called a causal graph, $G = (\mathbf{V}, \mathbf{E})$. Here $\mathbf{V}$ is the set of observable nodes and $\mathbf{E}$ is the set of directed edges. For any pair V_i, V_j, a directed edge $V_i \to V_j \in \mathbf{E}$ indicates that V_i is a parent of V_j, i.e., $V_i \in Pa(V_j)$ and V_j is a child of V_i, i.e., $V_j \in Ch(V_i)$ if and only if V_j is in the domain of f_{V_i}. There exists a bi-directed edge $V_i \leftrightarrow V_j \in \mathbf{E}$ in G if V_i and

V_j share a latent confounder. $An(V)$ and $De(V)$ represent the ancestors and descendants of V respectively. $Nbr(V)$ represents the nodes that are either parent, children of V, or share a bi-directed edge with V. For a variable set $\mathbf{V}$, $Pa(\mathbf{V}) = \{Pa(V_i)\}_{V_i \in \mathbf{V}} \setminus \mathbf{V}$. $Ch(\mathbf{V})$ also follows the same. However, $An(\mathbf{V}), De(\mathbf{V})$ and $Nbr(\mathbf{V})$ has set $\mathbf{V}$ included. We let G_S to denote an induced subgraph of G over any subset S of node $\mathbf{V}$, $G_{\overline{S}}$ be the graph obtained by removing the incoming edges to S from G_S, and $G_{\underline{S}}$ be G_S with all outgoing edges of S removed. We define an intervention as $\text{do}(x)$ where $\text{do}(x)$ replaces f_X with the equation $X = x$ and in other functions where X occurs. We represent the observed distribution after such an intervention as $\mathcal{P}_x(\mathbf{V})$ and the causal graph as $G_{\overline{X}}$. Let $\langle X, Y, Z \rangle$ be any consecutive triple along a path p. Y is a *collider* on p if both edges are into Y. Otherwise, Y is a *non-collider* on p.

Definition 2.2 (d-separation)**.** In a DAG, a path p between vertices X and Y is *d-connecting (active)* relative to a set of vertices $\mathbf{Z} (X, Y \notin \mathbf{Z})$ if (i) every non-collider on p is not in $\mathbf{Z}$ and (ii) every collider on p is an ancestor of some $Z \in \mathbf{Z}$. If there is no d-connecting path between X and Y relative to $\mathbf{Z}$, we say X and Y are *d-separated* relative to $\mathbf{Z}$, denoted as $(X \perp\!\!\!\perp Y | \mathbf{Z})_G$.

Definition 2.3 (Causal Effect Identifiability [25])**.** Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be disjoint sets. The causal effect of an action $\text{do}(\mathbf{x})$ on a set of variables $\mathbf{Y}$ in a given context $\mathbf{z}$ is said to be identifiable from P in G if $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$ is (uniquely) computable from P in any causal model that induces the causal graph.

Distribution shifts: Distribution shifts refer to the changes between training conditions and deployment conditions that prevent the generalization of machine learning and statistical models. For a set of features $\mathbf{X}$ and a target variable Y, distribution shift can be categorized into sub-groups ([31]) based on assumptions about the training domain and test domain. For example: $i)$ $P(Y)$ changes while $P(Y|\mathbf{X})$ stays fixed (target shift), $ii)P(Y|\mathbf{X})$ changes while $P(Y)$ stays fixed (conditional shift), $iii)$only $P(\mathbf{X})$ changes (covariate shift). To prevent failure driven by distribution shifts, we observe the relationships among dataset variables and how the dataset is generated. One way is to model the underlying data-generating process as a causal graph and identify the sources where the shift occurs in the graph.

For example, in Figure 1, assume that we are given the knowledge that the distributions of H change between the training and testing data. We address this in the causal graph by adding an auxiliary variable called selection variable S ([13]) pointing to H. Suppose we want to predict N from A, X, and I. The Bayes-optimal predictor models the conditional distribution $P(N|A, I, X)$. It is easy to see that there exists a d-connecting path from S to N relative to the set $\{A, I, X\}$ as X is a child of H such that N is conditionally dependent on S given A, I, X. As a result, this conditional distribution changes in the target environment

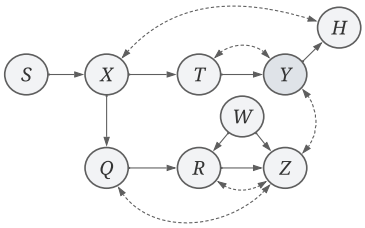


Figure 2: Selection diagram: Causal Graph with S

making the model prone to distribution shift. Therefore, a systematic approach to ensure invariant prediction is to intervene on H and use conditioning sets that render target N independent from S producing an identifiable conditional query. This method is known as graph surgery ([29]).

Definition 2.4 (Graph surgery estimator)**.** Let S be the shift variables, and Y be the target variable. For any subsets $\mathbf{Q}, \mathbf{W} \subseteq \mathbf{V}$, if $(Y \perp\!\!\!\perp S|\mathbf{W})_{G_{\overline{\mathbf{Q}}}}$ and $P(Y|\text{do}(\mathbf{Q}), \mathbf{W})$ is identifiable in G and $P(Y|\text{do}(\mathbf{Q}), \mathbf{W}) \neq P(Y)$, then $P(Y|\text{do}(\mathbf{Q}), \mathbf{W})$ is called a *graph surgery estimator*.

We will illustrate the concept of graph surgery estimator and other graphical definitions using the graph in Figure 2. In this example, $P(Y|\text{do}(X), H, T)$ is a graph surgery estimator because $P(Y|\text{do}(X), H, T)$ is identifiable and $(Y \perp\!\!\!\perp S|T, H)_{G_{\overline{X}}}$. Note that $P(Y|\text{do}(X), H, T) \neq P(Y|X, H, T) \neq P(Y)$, i.e., the intervention $\text{do}(X)$ has non-zero effect on the distribution of Y. The purpose of a graph surgery estimator is to find a predictor that is invariant across environments by shielding off the causal effects of S to Y. It uses interventional queries that can be computed from observational distribution as invariant predictors that are not available by only checking d-separation. In Figure 2, suppose we use H for predicting Y. Thus the target variable is distributed as $P(Y|H)$. After conditioning on H, we search for a feature set K to further use, keeping Y d-separated from S. However, there does not exist any such K for invariant predictors unless we utilize a graph surgery estimator. Similarly, in Figure 1, $P(N|\text{do}(H), A, I, X)$ is a graph surgery estimator since the query is identifiable and $(N \perp\!\!\!\perp S|A, I, X)_{G_{\overline{H}}}$. One crucial condition of the graph surgery estimator is that the interventional distributions have to be identifiable from observational training data. Next, we provide several definitions that are used in identifiability, which will also be useful for our algorithm.

Definition 2.5 (C-component)**.** A graph G where any pair of observable nodes is connected by a bidirected path is called a c-component (confounded component).

Definition 2.6 (C-tree)**.** Let G be a C-component such that each vertex of G has at most one child, and only one vertex Y (called the root) has no children. Then G is called a Y-rooted C-tree.

Definition 2.7 (C-forest)**.** Let G be a C-component such that each vertex of G has at most one child except a non-empty vertex set $\mathbf{Y}$ that has no children. Then G is called a $\mathbf{Y}$-rooted C-forest.

Note that every C-tree is also a C-forest, but the converse is not true. In Figure 2, $G_{T,Y,Q,R,Z}$ is a $\{Y, Z\}$-rooted C-forest, but it is not a C-tree. Additionally, $G_{Q,R,Z}$ is both a Z-rooted C-tree and a C-forest as it has at least one node with no children and other nodes with exactly one child. However, $G_{Z,R,W}$ fits neither definition as W does not belong to the same C-component of $\{R, Z\}$. We are now ready to use these concepts for understanding a particular graphical structure that is related to causal identifiability and is also used often in the causal discovery literature.

Definition 2.8 (Inducing paths for sets)**.** Let $\mathbf{X}, \mathbf{Y}$ be sets of variables in G. A path p between $\mathbf{X}$ and $\mathbf{Y}$ is called an inducing path if every non-endpoint vertex is a collider on the path and an ancestor of either $\mathbf{X}$ or $\mathbf{Y}$.

Definition 2.9 (Hedge)**.** Let $\mathbf{X}, \mathbf{Y}, \mathbf{W}$ be sets of variables in G. Let F, F' be $\mathbf{R}$-rooted C-forests in G such that $F \cap \mathbf{X} \neq \emptyset$, $F' \cap \mathbf{X} = \emptyset$ and $F' \subset F$, for some $\mathbf{R} \subset An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$. Then F and F' form a hedge for $P(\mathbf{Y}|\text{do}(\mathbf{X}))$.

For instance, in Figure 2, if $R = \{Z\}$ then $F = \{Q, R, Z\}$ and $F' = \{R, Z\}$ form a hedge for $P(Z|\text{do}(Q))$.

3 FINDING GRAPH SURGERY ESTIMATORS IN POLYNOMIAL TIME

In this section, we describe the details of our approach of finding graph surgery estimators. First, we introduce the theoretical results of causal identifiability that lead to the development of the algorithm. Then, we discuss the workings of our proposed algorithm. We leave most of the proofs to appendix Section A.

We extend the hedge condition to a generalized hedge condition for conditional queries.

Definition 3.1 (Generalized Hedge Condition)**.** Let $\mathbf{X}, \mathbf{Y}, \mathbf{W}$ be sets of variables in G. Let $\mathbf{Z} \subseteq \mathbf{W}$ be the maximal set such that $P(\mathbf{Y}|\text{do}(\mathbf{X}), \mathbf{W}) = P(\mathbf{Y}|\text{do}(\mathbf{X}, \mathbf{Z}), \mathbf{W} \setminus \mathbf{Z})$. Let F, F' be $\mathbf{R}$-rooted C-forests in G such that $F \cap (\mathbf{X} \cup \mathbf{Z}) \neq \emptyset$, $F' \cap (\mathbf{X} \cup \mathbf{Z}) = \emptyset$, and $F' \subset F$, and $\mathbf{R} \subset An(\mathbf{Y} \cup (\mathbf{W} \setminus \mathbf{Z}))_{G_{\overline{\mathbf{X} \cup \mathbf{Z}}}}$. Then F and F' is said to form a hedge for $P(\mathbf{Y}|\text{do}(\mathbf{X}), \mathbf{W})$.

We can use Figure 2 to illustrate Definition 3.1. Let $\mathbf{X} = \{Q, T\}, \mathbf{W} = \{R, W\}, \mathbf{Y} = \{Y, Z\}$. By rule 2 of do-calculus [12], $P(Y, Z|\text{do}(Q, T), R, W) = P(Y, Z|\text{do}(Q, T, W), R)$. We can let $F = \{Q, R, Z, T, Y\}$ and $F' = F \setminus \{Q, T, W\}$ so that F, F' are $\{Y, Z\}$-rooted C-forests to form a hedge for $P(Y, Z|\text{do}(Q, T), R, W)$. The following theorem describes the relationship between a hedge and causal identifiability.

Theorem 3.2. *There exists a hedge for $P(\mathbf{Y}|do(\mathbf{X}), \mathbf{W})$ according to the generalized hedge condition if and only if $P(\mathbf{Y}|do(\mathbf{X}), \mathbf{W})$ is unidentifiable in G.*

Definition 3.3 (Ancestral Confounded Set)**.** Let Y be a variable in G. A set $\mathbf{K}$ is ancestral confounded (ACS) for Y if $\mathbf{K} = An(Y)_{G_{\mathbf{K}}} = C(Y)_{G_{\mathbf{K}}}$. We call an ACS T_Y maximum ACS (MACS) if T_Y is the largest set such that $\mathbf{K} = An(Y)_{G_{\mathbf{K}}} = C(Y)_{G_{\mathbf{K}}}$.

In Figure 2, for variable Z, $\mathbf{K} = An(Z)_{G_{\mathbf{K}}} = C(Z)_{G_{\mathbf{K}}} = \{R, Z\}$ is an ACS for Z while $\mathbf{K}' = \{Q, R, Z\}$ is the largest set satisfying the same ACS condition. Thus, $T_Z = \mathbf{K}' = \{Q, R, Z\}$ is the MACS for Z. One special property about MACS is that it is unique for any variable in G by Theorem 4 in [24]. Throughout this work, we will denote the MACS of a set $\mathbf{K}$ in G as $T_{\mathbf{K}}$. The significance of the MACS is that it helps determine whether a causal query is identifiable in a given causal graph. Next, we need to define another graphical structure known as AC-component for finding MACS.

Definition 3.4 (AC-component)**.** A set $\mathbf{Y}$ of nodes in G is an ancestral confounded component (AC-component) if $\mathbf{Y}$ is a singleton e.g. $\mathbf{Y} = \{Y\}$ or $\mathbf{Y}$ is a union of two distinct AC-components $\mathbf{Y}_1, \mathbf{Y}_2$ which have ancestral confounded sets S_1, S_2, respectively, and S_1, S_2 are connected by a bidirected arc.

For example, in Figure 2, $\{Y, Z\}$ is an AC-component because $\{Z\}$ is an ACS for Z and $\{Y\}$ is an ACS for Y and Y and Z are connected by a bidirected arc. We can leverage the algorithm by [24] called **Find-MACS-on-set** (see Algorithm 2 in Section B.2) to find the MACS of a set in G. The following lemma describes the relationship between a MACS and an important graphical structure related to causal identifiability.

Lemma 3.5. *Let $\mathbf{Y} = \{Y\}$. The output of **Find-MACS-on-set**$(G, \mathbf{Y})$ is the MACS of Y. The MACS of Y is a Y-rooted C-tree.*

3.1 RELATIONSHIPS WITH GRAPH SURGERY ESTIMATORS

We now explain how the previous section relates to finding a graph surgery estimator. Theorem 3.6 and Theorem 3.7 imply that knowing the MACS for a target variable can help identify some causal queries that will not be graph surgery estimators. If selection variable S has a child W in a Y-rooted C-tree and W forms a hedge for $P(Y|do(W))$, then there is no graph surgery estimator in G.

Theorem 3.6. *For some $W \in Ch(S)$, if there exists a hedge for $P(Y|do(W))$, then for any $\mathbf{H}, \mathbf{J} \subseteq \mathbf{V}$, we have $(Y \not\perp\!\!\!\perp S|\mathbf{J})_{G_{\overline{H}}}$ or $P(Y|do(\mathbf{H}), \mathbf{J})$ is unidentifiable in G.*

Theorem 3.7. *If the selection variable S is a parent of MACS T_Y, then there is no graph surgery estimator in G.*

Furthermore, we can systematically leverage the MACS for the target variable to find graph surgery estimators. Theorem 3.8 says that we can find some graph surgery estimators by finding the union of the MACS of the target and the MACSs of some children of the target. The intuition is that we can find some graph surgery estimators by intervening on the parents of the MACs whenever the selection variable S is not a parent of those MACSs. Although Theorem 3.8 implies that we can find graph surgery estimators by using the MACS of the subsets of the children, we only use the largest subset i.e. picking $\mathbf{K} = \mathcal{H}$ (denoted in Theorem 3.8) to incorporate as many predictors as possible in our algorithm.

Theorem 3.8. *Let T_Y be the MACS of Y in G, $\mathcal{H} \coloneqq \{H : H \in Ch(Y),\ Pa(T_H) \not\ni S\}$ and $T_J \coloneqq \bigcup_{H\in\mathbf{K}} T_H$ for any $\mathbf{K} \subseteq \mathcal{H}$, where T_H is the MACS with respect to the variable H. Let $\mathbf{D} = Pa(T_Y \cup T_J)$. If S is not a parent of T_Y, then $P(Y|do(\mathbf{D}), \mathbf{K}, \mathbf{W})$ is identifiable in G and $(Y \perp\!\!\!\perp S|\mathbf{W}, \mathbf{K})_{G_{\overline{\mathbf{D}}}}$ for any $\mathbf{W} \subseteq (T_Y \cup T_J) \setminus (Y \cup \mathbf{K})$.*

Corollary 3.9. *Let T_Y be the MACS of Y in G and $\mathbf{D} = Pa(T_Y) \setminus T_Y$. If S is not a parent of T_Y, then $P(Y|do(\mathbf{D}), \mathbf{W})$ is identifiable in G and $(Y \perp\!\!\!\perp S|\mathbf{W})_{G_{\overline{\mathbf{D}}}}$ for any $\mathbf{W} \subseteq T_Y \setminus Y$*

In addition, we search for the bidirected neighbors of Y that are not in any MACS of the children of the target or in the MACS of the target. There are two reasons for doing so. First, we find the MACs of these bidirected neighbors to increase the number of graph surgery estimators output by our proposed algorithm. Second, finding the MACS of the bidirected neighbors of Y that are in any MACS of children of the target or the target itself can be inefficient due to duplicate searches for the same query.

Theorem 3.10. *Let T_Y be the MACS of Y in G, T_H be the MACS of any child H of Y in G. Define*

$$T_{\mathbf{C}} \coloneqq \bigcup_{H\in Ch(Y)} T_H \tag{1}$$

$$\mathcal{Z} \coloneqq \{Z : Z \in (C(Y) \cap Nbr(Y)) \setminus (T_Y \cup T_{\mathbf{C}}) \quad s.t. Pa(T_{Y\cup Z}) \not\ni S\} \tag{2}$$

$$T_B \coloneqq \bigcup_{Z\in\mathbf{M}} T_{Y\cup Z} \tag{3}$$

for any $\mathbf{M} \subseteq \mathcal{Z}$ where $T_{Y\cup Z}$ is the MACS for the set $(Y \cup Z)$. Let $\mathbf{D} = Pa(T_B)$. If S is not a parent of T_Y, then $P(Y|do(\mathbf{D}), \mathbf{M}, \mathbf{W})$ is identifiable in G and $(Y \perp\!\!\!\perp S|\mathbf{W}, \mathbf{M})_{G_{\overline{\mathbf{D}}}}$ for any $\mathbf{W} \subseteq (T_B) \setminus (Y \cup \mathbf{M})$.

3.2 ALGORITHM DETAILS

Our approach begins with Algorithm 3:**ID4IP**. It takes the selection variable S, the target variable Y, and the causal graph G as input, and outputs at least one invariant predictor if any exists. In the beginning, we initialize two sets P_{set} and L_{set} for storing the invariant predictors and their corresponding losses. At line 4, we call Algorithm 2 in Section B.2: **Find-MACS-on-set** as a sub-routine which finds the MACS T_Y (Definition 3.3) of the target variable Y. If S is a parent of T_Y, the algorithm returns **FAIL** at line 5-6. This indicates that there exists no graph surgery estimator in G.

At lines 7, 8 and 10, we call the sub-routine Algorithm 2: **addBestIP**. This sub-routine takes selection variable S, target Y, the C-tree T_Y, P_{set}, L_{set}, a variable set Z and roots R as inputs. The first five parameters stay the same for all these subroutine calls while $\mathbf{Z}$ and $\mathbf{R}$ change. Conditioning on the variable set $\mathbf{Z}$ allows us to find more invariant predictors. $\mathbf{R}$ is either empty or contains the target Y.

Inside the **addBestIP** sub-routine, we initialize a set T_J with the Y rooted C-tree, $\mathcal{H}$ with the target variable and $T_{visited}$ as an empty set which will track all visited C-tree nodes so that the algorithm does not have to consider those nodes again. If $\mathbf{Z}$ is non-empty, for each variable $H \in \mathbf{Z}$, we call the sub-routine **Find-MACS-on-set** for $\mathbf{R} \cup H$ (line 6). This

1: **Input:** A set of targets $\mathbf{Y}$, an intervention set $\mathbf{X}$, a conditioning set $\mathbf{W}$
2: **Output:** P, a causal query that corresponds to the lowest training loss L among the searched queries.
3: $A_0 = \mathbf{Y}$ {A cumulative array s.t., $A_i \subset A_{i+1}$}
4: **for** $i \in 0 \dots (|\mathbf{W}| - 1)$ **do**
5: $\quad K = \underset{J\in\mathbf{W}\setminus A_i}{\arg\min}\ \mathbf{computeLoss}(A_i \cup \{J\}, \mathbf{X}) - \mathbf{computeLoss}(A_i, \mathbf{X})$
6: $\quad A_{i+1} = A_i \cup \{K\}$
7: **Return** $P(A_{|\mathbf{W}|}|do(\mathbf{X}))$ {Value of the last index.}

Algorithm 1: Greedy-Eval($\mathbf{Y}, \mathbf{X}, \mathbf{W}$)

1: **Input:** Selection variable S, Target Y, C-tree T_Y, Predictors and Loss terms P_{set}, L_{set}, Variable set $\mathbf{Z}$, Additional Roots $\mathbf{R}$.
2: **Output:** A set of predictors P_{set}, a set of losses L_{set}, a set of MACS $T_{visited}$
3: $T_J = T_Y$; $\mathcal{H} = \{Y\}$; $T_{visited} = \emptyset$
4: **if** $\mathbf{Z} \neq \emptyset$ **then**
5: $\quad$ **for** $H \in \mathbf{Z}$ **do**
6: $\quad\quad T_H =$ **Find-MACS-on-set** $(G, \mathbf{R} \cup \{H\})$
7: $\quad\quad T_{visited} = T_{visited} \cup T_H$
8: $\quad\quad$ **if** $S \notin Pa(T_H)$ **then**
9: $\quad\quad\quad T_J = T_J \cup T_H$
10: $\quad\quad\quad \mathcal{H} = \mathcal{H} \cup H$
11: $P, L =$ **Greedy-Eval**($\mathcal{H}$, $Pa(T_J), T_J \setminus \mathcal{H}$)
12: **if** $P \notin P_{set}$ **then**
13: $\quad P_{set}.append(P); L_{set}.append(L)$
14: **Return:** $P_{set}, L_{set}, T_{visited}$

Algorithm 2: addBestIP($S, Y, T_Y, P_{set}, L_{set}, \mathbf{Z}, \mathbf{R}$)

1: **Input:** Selection variable S, target Y, causal graph $G = (\mathbf{V}, \mathbf{E})$
2: **Output:** An invariant predictor $P^\star$ or **FAIL** to indicate that there is no graph surgery estimator in G.
3: $P_{set} = \emptyset$; $L_{set} = \emptyset$
4: T_Y = **Find-MACS-on-set**(G, Y)
5: **if** $S \in Pa(T_Y)$ **then**
6: **Return FAIL**
7: $P_{set}, L_{set}, T_\emptyset =$ **addBestIP**$(S, Y, T_Y, P_{set}, L_{set}, \emptyset, \emptyset)$
8: $P_{set}, L_{set}, T_{\mathbf{Ch}} =$ **addBestIP**$(S, Y, T_Y, P_{set}, L_{set}, Ch(Y), \emptyset)$
9: $\mathbf{T} = T_Y \cup T_{\mathbf{Ch}}$
10: $P_{set}, L_{set}, T_C =$ **addBestIP**$(S, Y, T_Y, P_{set}, L_{set}, (C(Y) \cap Nbr(Y)) \setminus \mathbf{T}, \{Y\})$
11: **if** $P_{set} \neq \emptyset$ **then**
12: **Return** the predictor P in P_{set} corresponding to the lowest loss L found in L_{set} that have been searched
13: **Return FAIL**

Algorithm 3: ID4IP(S, Y, G)

sub-routine returns a MACS T_H that contains a C-forest rooted at $\mathbf{R}$ and H. We store T_H in $T_{visited}$. However, we only consider those c-forests for which $S \notin Pa(T_H)$ holds (lines 8- 10). We combine the MACS T_H with the MACS of previous variables, in T_J, and save H in $\mathcal{H}$ (lines 9- 10). After the loop ends, we call the sub-routine Algorithm 1: **Greedy-Eval** at line 11 and send the three sets $\mathcal{H}, Pa(T_J)$ and $T_J \setminus \mathcal{H}$ as inputs. This sub-routine returns the causal query corresponding to the lowest validation loss among other queries found by our algorithm. If this is a new query that we have not found yet, we add it and its corresponding loss to P_{set}, L_{set} and return them along with $T_{visited}$ from the sub-routine.

Now, back to our initial Algorithm 3, all three calls at lines 7, 8 and 10 have common first five parameters. The function call at line 7 finds the invariant predictors that use ancestors of Y as inputs. For this case, lines 4- 10 of **addBestIP** sub-routine will get skipped. To better understand our algorithm till this step, we can look at Figure 3 where we have MACS $T_Y = \{E, Y\}$. Thus the step at line 7 will enlist the lowest loss causal query in the form of $P(Y|do(D), H), H \subseteq \{E\}$. However, if there exists a bidirected edge between D and E then T_Y would be $\{A, D, E, Y\}$, and S would be a parent of this set. Thus we would have to return fail (line 5). The reason is that we can not condition on any variables in T_Y since there would exist inducing paths from S to Y. And we can not intervene on any variables in T_Y as well since it would be non-identifiable. This illustrates a situation when there exists no invariant predictor. For the function call at line 8, we send $\mathbf{Z} = Ch(Y)$ as a parameter. At Algorithm 2, lines 5-10, we iterate over $\mathbf{Z}$ and find C-tree rooted at each of the children since $R = \emptyset$. We store these C-trees in $T_{visited}$. However, we can not utilize children for invariant

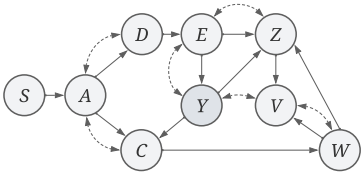


Figure 3: A causal graph inducing invariant predictors of the form $P(Y|do(D), H)$, $H \subseteq \{E\}$ when we consider the C-tree T_Y. Similarly, we condition on $Ch(Y) = \{Z\}$ and utilize T_Z for predictors as $P(Y|do(D, W), Z, H)$, $H \subseteq \{E\}$. Finally we condition on bi-directed neighbors of Y and utilize $T_{V,Y}$ for predictor $P(Y|do(C, Z), V, H)$, $H \subseteq \{W\}$

predictors if S is a parent of their C-tree. This way we can search for the invariant predictors that employ ancestors of the chosen $Ch(Y)$ for prediction since Y becomes dependent on those ancestors after conditioning on the chosen $Ch(Y)$. In our example in Figure 3, $Ch(Y) = \{C, Z\}$ and $T_C = \{A, C\}, T_Z = \{E, Y, Z\}$. We can not utilize child C for invariant predictors since $S \in Pa(T_C)$. However, $Pa(T_Z) = \{D, W\}$ and we have $\mathcal{H} = \{Y, Z\}$. **Greedy-Eval** will choose one query from all queries in the form of $P(Y|do(D, W), Z, H,), H \subseteq \{E\}$.

At line 9 of **ID4IP**, we store the Y-rooted C-tree T_Y and the C-trees rooted at the chosen children T_{Ch}, returned from the call at the previous step in $\mathbf{T}$. Finally, at line 10, we only consider specific bi-directed neighbors $\mathbf{N}$ of Y that are not in $\mathbf{T}$, such that $\mathbf{N} = C(Y) \cap Nbr(Y) \setminus \mathbf{T}$. The goal is to avoid computation of the overlapping queries among the MACS found during these 3 steps. We send this set as a parameter of the **addBestIP** sub-routine so that we can find invariant predictors that utilize ancestors of the bi-directed neighbors in this set. Similar to children, we iterate over these neighbors to find more invariant predictors that can predict Y after conditioning on that bi-directed neighbor. For this purpose, we find c-forests rooted at both Y and some neighbor in $\mathbf{N}$ such that S is not a parent of the found C-forest. Thus, we send root $R = \{Y\}$ as a parameter of the sub-routine, unlike the previous case when we sent $R = \emptyset$. For Figure 3, V is the bi-directed neighbor of Y. Here $T_V = \{Y, V, W\}$ and $Pa(T_V) = \{C, Z\}$. Therefore, the last **addBestIP** call will return the query with the lowest validation loss from a set of causal queries of the form $P(Y|do(C, Z), V, H), H \subseteq \{W\}$

After these 3 function calls, we return the predictor from P_{set} with the minimum validation loss. If our algorithm can not find any predictors even after these 3 steps, i.e., P_{set} is empty, we return fail indicating that there exists no invariant predictor for this graph. This follows from Theorem 3.11.

During the execution of our algorithm, we call Algorithm 1: **Greedy-Eval** several times to find the query with the lowest validation loss among all found surgery estimators. The

Greedy-Eval sub-routine takes $\mathcal{H}, Pa(T_J)$ and $T_J \setminus \mathcal{H}$ as arguments, i.e., set of targets Y, set of interventions X and set of conditions W, respectively. Here we initialize an array of lists A with Y as the first element. Then the algorithm loops for $|W| - 1$ times and each time updates the list at $i+1$-th position of the array A with $A[i]$ and some new variable K (lines 4-6). The variable K is chosen from the conditioning set, which combined with $A[i]$, helps reduce the training loss (line 5). After the $i+1$-th iteration, $A[i+1]$ indicates the joint variable list in the causal query that we might return as the result. Finally at line 7, we return the query $P(A_{|\mathbf{W}|}|do(\mathbf{X}))$ where $A_{|\mathbf{W}|}$ indicates the joint variable list of the array after the loop ends. This query is received in Algorithm 3, and finally output as an invariant predictor with minimum validation loss among the queries that can be produced from the inputs of **Greedy-Eval**.

Next, we show the soundness of ID4IP. Furthermore, we also show that if there exists a graph surgery estimator, then **ID4IP** outputs a graph surgery estimator:

Theorem 3.11. *If there exists a graph surgery estimator in G, **ID4IP** outputs at least one graph surgery estimator.*

Theorem 3.12. ***(Soundness of Algorithm 3: ID4IP)** When **Algorithm 3: ID4IP** returns an estimator, it is a graph surgery estimator with respect to the given target and the selection variable in G.*

4 COMPLEXITY ANALYSIS

In this section, we compare the time complexity of ID4IP with that of the Graph Surgery Estimator (GSE) algorithm (see Algorithm 5 in Section B.4) [29]. GSE uses various sub-routines for converting conditional queries to unconditional queries by checking d-separations. Fortunately, [22] has provided an efficient algorithm for checking d-separation condition, which we will incorporate into the analysis of GSE algorithm's time complexity.

Theorem 4.1. ***(GSE Complexity)*** [1] *Let $|Ch(S)| = C$, $\mathbf{M} = Ch(S)$, $\mathbf{Q} = \mathbf{V} \setminus (\mathbf{M} \cup Y)$. Given a causal graph $G = (\mathbf{V}, \mathbf{E})$ and disjoint variables $\mathbf{X}, \mathbf{Y} \subset V$, the time complexity of Graph Surgery Estimator (GSE) (Algorithm 5 in Section B.4) is: $O(2^{2(|\mathbf{V}|-C)-1} \times B)$, where B represents the time complexity of **ID** algorithm.*

One of the major benefits of using MACS lies in its efficiency. Combining with greedy search, **ID4IP** enjoys a polynomial time complexity relative to the complexity of **ID** algorithm, whereas GSE has exponential time complexity relative to the complexity of **ID** algorithm.

[1]The algorithm presented in [29] does not explicitly search over supersets but the proofs of both soundness and completeness of the algorithm indicate that it should search over supersets, which is why we are taking this version as a baseline.

Theorem 4.2. *[24] **Find-MACS-on-set**$(G, \mathbf{Y})$ outputs the MACS of $\mathbf{Y}$ in polynomial time in the size of graph.*

Theorem 4.3. ***(ID4IP Complexity)** Given a causal graph $G = (\mathbf{V}, E)$ and disjoint variables $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{V}$, the complexity of **ID4IP** (Algorithm 3) is $O(|(C(Y) \cap Nbr(Y)) \setminus (T_Y \cup T_{\mathbf{C}})| + |Ch(Y)| + 1)K + (|T_Y| - 1 + |T_J| + |T_J^{'}| - |\mathcal{H}^{'}| - |\mathcal{H}|)B)$, where K represents the time complexity of **Find-MACS-on-set** and B represents the time complexity of **ID** algorithm, T_Y be the MACS of Y in G, T_H be the MACS of a child H of Y in G, and $T_{\mathbf{C}} \coloneqq \bigcup_{H \in Ch(Y)} T_H$.*

5 EXPERIMENT

In this section, we will show a comparison between Graph Surgery Estimator (Algorithm 5 in Section B.4) and ID4IP algorithms (Algorithm 3) in terms of accuracy, run time through both synthetic and semi-synthetic data sets. Throughout the experiment, we use a server that has 128 cores CPUs with 126 GB of memory. We conduct the experiment in Python programming language. The source code is available at: https://github.com/kenneth-lee-ch/id4ip

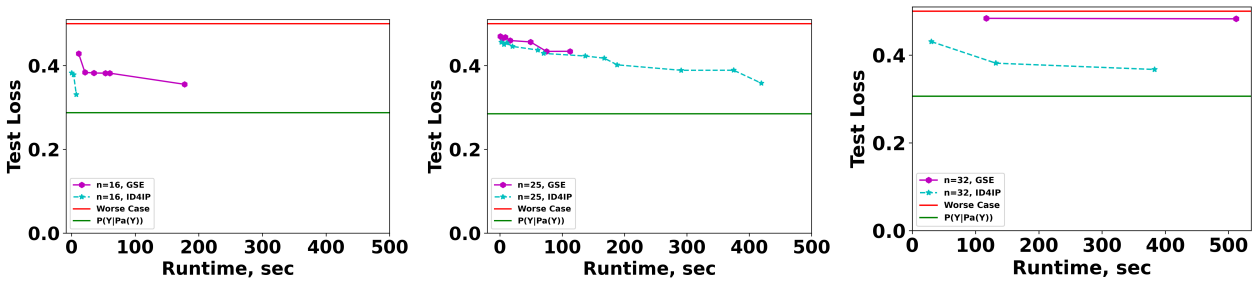
5.1 SYNTHETIC EXPERIMENT

In the synthetic experiment, we evaluate the performances of **ID4IP** and **GSE** to find graph surgery estimators within a given time limit for the program execution and their sensitivities to training sample size.

5.1.1 Finding graph surgery estimators in a given time limit

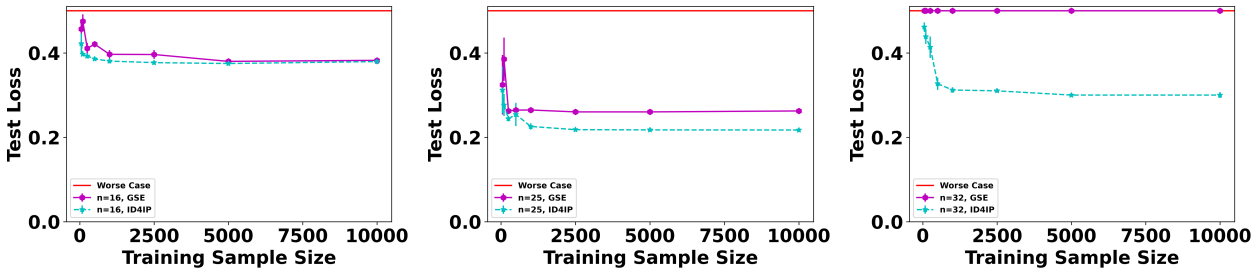
We randomly generate DAGs of size n by using PyAgrum library in Python [5], where $n \in \{16, 25, 32\}$ with $n/2$ number of latent confounders. The number of directed edges is set to be $3n$. Each variable follows a Bernoulli distribution with a randomly generated probability between 0 and 1. We assign the selection variable to be an ancestor of the target variable. The selection variable is generated by changing the marginal distribution of the assigned variable before generating the test set while fixing the same target variable. For the purposes of showing the efficiency of finding graph surgery estimator, we generate 10000 training samples and both **GSE** and **ID4IP** directly use the population distribution of the training data to learn which graph surgery estimator is best based on the lowest zero-one loss between their predicted labels and actual labels from the training data.

Based on Figure 4, we can see that **ID4IP** finds graph surgery estimators more efficiently as the number of observed variables increases from 16 to 32. Additionally, the graph surgery estimators found by **ID4IP** often result in lower test loss since **GSE** cannot find the best graph surgery estimator within the time limit. From Figure 4(c), we see



(a) 16 observed variables (b) 25 observed variables (c) 32 observed variables

Figure 4: Comparison between **ID4IP** (Cyan dashed lines) and **GSE** (Purple solid lines) in terms of finding graph surgery estimators within a given time limit of 600 seconds. The horizontal axis represents the time allowed to execute to find the predictors. The vertical axis represents zero-one test loss. Each point represents a graph surgery estimator found by the models. **Green**: the test loss evaluated by Bayes optimal $P(Y|Pa(Y))$, where $Pa(Y)$ includes the latent confounders as observed. **Red**: the worst predictive performance by a dummy classifier.



(a) 16 observed variables (b) 25 observed variables (c) 32 observed variables

Figure 5: Comparison between **ID4IP** (Cyan dashed lines) and **GSE** (Purple solid lines) in terms of sensitivities to training sample size. The horizontal axis represents the number of training samples used to find the graph surgery estimators. The vertical axis represents zero-one loss averaged by using three graph surgery estimators found by three randomly generated training samples. The time limit is set to be 60 seconds. Graphs in which no graph surgery estimators exist are excluded.

that **GSE** fails to find a graph surgery estimator in the first 100 seconds. This is possible since **GSE** needs to check whether a causal query is unidentifiable by calling the ID algorithm repeatedly during the search, whereas each causal query found by **ID4IP** is identifiable and it only calls on ID algorithm for deriving the estimands of the found graph surgery estimators.

5.1.2 Sensitivity to training sample size

In addition, we evaluate the predictive performance of both **GSE** and **ID4IP** by varying the number of training samples. Similar to the previous experiment, we randomly generate DAGs of size n, where $n \in \{16, 25, 32\}$ with $n/2$ number of latent confounders. The number of directed edges is set to be $3n$. Each variable follows a Bernoulli distribution with a randomly generated probability between 0 and 1. We evaluate the sensitivity by a range of training sample sizes e.g. $50, 100, 250, 500, 1000, 2500, 5000, 10000$. For each training sample size, we randomly generate three different training samples and report the zero-one test loss averaged and the standard errors based on potentially three different graph surgery estimators respectively found by the models while fixing the same test sample of size 10000 for all training sample sizes. We fix the time limit to be 60 seconds for both algorithms to find graph surgery estimators. We also use 30% of the training data for validation. We set the test loss to 0.5 if the model fails to find a graph surgery estimator in the given time limit. Furthermore, we adopt a heuristic for learning the training distribution from the observed data. We turn every bidirected edge in the given causal graph to a directed edge while maintaining acyclicity. If the children of the latent confounder already have a directed edge among them, we simply remove that bidirected edge. We use the *BNLearner* function in PyAgrum, which uses a greedy hill climbing algorithm by default, to learn the conditional probability tables from the observed data with a smoothing prior. We use this approximated training distribution to evaluate any graph surgery estimator found by both models.

From Figure 5, we see as the training sample size increases,

Algorithm	Sachs (11 nodes)	Alarm (37 nodes)
GSE	0.80	0.57
ID4IP	0.80	0.83
LR	0.53	0.52

Table 1: Algorithms performance comparison based on micro-averaged F1 score within time limit of 120 seconds

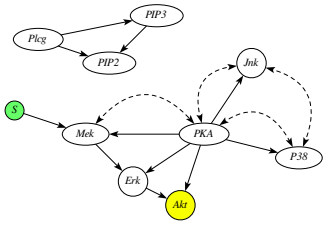


Figure 6: Modified Sachs causal graph

both algorithms achieve lower test loss in general. We also see that the difference in test loss between **GSE** and **ID4IP** increases as the number of nodes increases. This is expected as we set the time limit to be 60 seconds which restricts **GSE** to find the best graph surgery estimator.

5.2 SEMI-SYNTHETIC DATA

There are several practical scenarios where the computational demand of the graph surgery estimator is a bottleneck. For example, some researchers investigate the distribution shifts problem due to medical record transfer [1, 20]. Although the causal graph mentioned in [20] is not large, it is evident that causal graphs learned from medical data can be well over 100 nodes [11]. In this experiment, we further demonstrate the use case of having causal graphs given as a priori information and the utility of our proposed algorithm in invariant prediction by using causal graphs and semi-synthetic data provided motivated by real-world settings.

5.2.1 Sachs dataset

Sachs dataset measures the expression level of numerous proteins and phospholipids in human cells [17]. It consists of the concurrent measurements of 11 phosphorylated proteins and phospholipids derived from thousands of individual primary immune system cells. Each variable has 3 states. We set the target to be the variable *Akt*. The original causal graph of this dataset is shown in the supplementary material. For the purpose of this experiment, we modified it to be the graph as Figure 6. We treat the variable *Raf* as the selection variable by converting all of its class 2 to be class 1 and using only samples that have *Raf* being class 0 to train the models. The test sample is then generated by taking all the samples that have *Raf* being class 1. The training sample size is 3632 and the test sample size is 3368. We further use 30% of the training data to validate the models before testing. We also train a logistic regression model (**LR**) with the predictors *PIP3*, *PIP2*, *Plcg* only. We report micro-averaged F1 score for **GSE**, **ID4IP**, and **LR** in table 1. We see that both **GSE**, **ID4IP** outperform **LR** as expected as **LR** suffers from the distribution shift. We can also see that **GSE** achieves the same test loss as **ID4IP** since the causal graph is reasonably small.

5.2.2 Alarm dataset

We also consider a larger causal graph provided by [2], which consists of 37 observed variables. We modified the graph such that it includes 7 latent confounders and 28 observed variables by treating some of its original observed variables as latent. Each variable has a number of states varying from 2 to 4. We provide the original causal graph and its modified version in the supplementary material. We picked the binary variable *DIFFICULTY* to be the selection variable and *BP* to be the target. All models are trained on 1462 training samples while the shifted test sample size is 13538. **LR** only uses the features *HISTORY*, *LVEDVOLUME*, *STROKEVOLUME*, *CVP*, and *PCWP*. From table 1, we see that **ID4IP** achieves the lowest test loss among all models. It is because **GSE** fails to find the best graph surgery estimator within the time limit and **LR** also suffers from distribution shifts.

6 CONCLUSION

We presented an algorithm that efficiently finds predictors invariant to distribution shifts and guarantees to find at least one if there exists any for a data-generating process that is correctly modeled as a causal graph. In particular, we utilize a graphical characterization of the identifiability of conditional causal queries with greedy search to increase the efficiency of finding invariant predictors. Our algorithm is sound that runs in polynomial time in contrast to the existing work that requires exponential time. As shown by the numerical experiments, our proposed algorithm has significantly reduced run time to reach predictive performance similar to the existing work within given time limits. In the future, it is worthwhile to develop the completeness of the algorithm. Another direction is to understand the approximation guarantees for greedy-search methods for invariant causal prediction. We also want to explore the idea of finding invariant predictors with weak confounding variables.

7 ACKNOWLEDGEMENTS

This research has been supported in part by NSF Grant CAREER 2239375.

REFERENCES

[1] Denis Agniel, Isaac S Kohane, and Griffin M Weber. Biases in electronic health record data due to processes within the healthcare system: retrospective observational study. *Bmj*, 361, 2018.

[2] Ingo A Beinlich, Henri Jacques Suermondt, R Martin Chavez, and Gregory F Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *AIME 89: Second European Conference on Artificial Intelligence in Medicine, London, August 29th–31st 1989. Proceedings*, pages 247–256. Springer, 1989.

[3] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton university press, 2009.

[4] J Quinonero Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. Dataset shift in machine learning. *The MIT Press*, 1:5, 2009.

[5] Gaspard Ducamp, Christophe Gonzales, and Pierre-Henri Wuillemin. aGrUM/pyAgrum : a Toolbox to Build Models and Algorithms for Probabilistic Graphical Models in Python. In *10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 609–612, Skørping, Denmark, September 2020.

[6] John Duchi, Tatsunori Hashimoto, and Hongseok Namkoong. Distributionally robust losses for latent covariate mixtures. *Operations Research*, 2022.

[7] Mingming Gong, Kun Zhang, Biwei Huang, Clark Glymour, Dacheng Tao, and Kayhan Batmanghelich. Causal generative domain adaptation networks. *arXiv preprint arXiv:1804.04333*, 2018.

[8] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *International conference on machine learning*, pages 2839–2848. PMLR, 2016.

[9] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

[10] Sara Magliacane, Thijs Van Ommen, Tom Claassen, Stephan Bongers, Philip Versteeg, and Joris M Mooij. Domain adaptation by using causal inference to predict invariant conditional distributions. *Advances in neural information processing systems*, 31, 2018.

[11] Galia Nordon, Gideon Koren, Varda Shalev, Benny Kimelfeld, Uri Shalit, and Kira Radinsky. Building causal graphs from medical literature and electronic medical records. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1102–1109, 2019.

[12] Judea Pearl. *Causality*. Cambridge university press, 2009.

[13] Judea Pearl and Elias Bareinboim. Transportability of causal and statistical relations: A formal approach. In *Twenty-fifth AAAI conference on artificial intelligence*, 2011.

[14] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[15] Ashkan Rezaei, Anqi Liu, Omid Memarrast, and Brian D Ziebart. Robust fairness under covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9419–9427, 2021.

[16] Mateo Rojas-Carulla, Bernhard Schölkopf, Richard Turner, and Jonas Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.

[17] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[18] Suchi Saria and Adarsh Subbaswamy. Tutorial: safe and reliable machine learning. *arXiv preprint arXiv:1904.07204*, 2019.

[19] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012.

[20] Jessica Schrouff, Natalie Harris, Sanmi Koyejo, Ibrahim M Alabdulmohsin, Eva Schnider, Krista Opsahl-Ong, Alexander Brown, Subhrajit Roy, Diana Mincu, Christina Chen, et al. Diagnosing failures of fairness transfer across distribution shift in real-world medical settings. *Advances in Neural Information Processing Systems*, 35:19304–19318, 2022.

[21] Peter Schulam and Suchi Saria. Reliable decision support using counterfactual models. *Advances in neural information processing systems*, 30, 2017.

[22] Ross D Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). *arXiv preprint arXiv:1301.7412*, 2013.

[23] Ilya Shpitser and Judea Pearl. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9:1941–1979, 2008.

[24] Ilya Shpitser and Judea Pearl. *Dormant independence*. AAAI Press, 2008.

[25] Ilya Shpitser and Judea Pearl. Identification of conditional interventional distributions. *arXiv preprint arXiv:1206.6876*, 2012.

[26] Agnieszka Słowik and Léon Bottou. On distributionally robust optimization and data rebalancing. In *International Conference on Artificial Intelligence and Statistics*, pages 1283–1297. PMLR, 2022.

[27] Amos Storkey et al. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 30:3–28, 2009.

[28] Adarsh Subbaswamy and Suchi Saria. Counterfactual normalization: Proactively addressing dataset shift using causal mechanisms. In *UAI*, pages 947–957, 2018.

[29] Adarsh Subbaswamy, Peter Schulam, and Suchi Saria. Preventing failures due to dataset shift: Learning predictive models that transport. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3118–3127. PMLR, 2019.

[30] John R Zech, Marcus A Badgeley, Manway Liu, Anthony B Costa, Joseph J Titano, and Eric Karl Oermann. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study. *PLoS medicine*, 15(11):e1002683, 2018.

[31] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International conference on machine learning*, pages 819–827. PMLR, 2013.