Finding Invariant Predictors Efficiently via Causal Structure

Kenneth Lee, Md Musfiqur Rahman, Murat Kocaoglu

1School 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. 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. Evaluates their predictor performance under mixture covariate shifts by modeling it as a distributionally robust optimization (DRO) problem. 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 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 (X) and the target variable (Y) to introduce methods to deal with different types of distribution shifts. Some examples include covariate shift where P(X) changes, target shift where P(Y) changes, and conditional shift where P(Y|X) changes.

There are mainly two types of stable training algorithms in the literature that consider different forms of distribution shifts: reactive and proactive. 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. 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. Proactive algorithms train with stable information/features that are invariant across environments.

Some recent proactive algorithms such as 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.
interventions are not actually performed but simulated from the finding dormant independence between two sets of variables. In this context, identifying the conditional independence state-

s 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 ([29,31]), 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 $P(N \mid \text{do}(H), A, I, X)$ can be uniquely calculated from training data and $\text{do}(H)$ d-separates $N$ from $S$. Thus we can train our model with predictors $I, X, H, A$ from a hypothetical $\text{do}(H)$ 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-


dependence between two sets of variables.


dependence between $N$ and $H$ i.e., unobserved common causes of two observable variables, a set of functions $\mathcal{F}$ and a product probability distribution $\mathcal{P}_{(\cdot)}$ over $\mathcal{N}$ and $\mathcal{U}$. Each observed variable is generated as $V_i \equiv f_i(Pa_i, E_i, U_{S_i})$, where $f_i \in \mathcal{F}$, $Pa_i \subseteq \mathcal{V}$, $E_i \in \mathcal{N}$ and $U_{S_i} \equiv \{U_j : j \in S_i\}$ for some $S_i \subseteq \mathcal{U}$. Variables set $\mathcal{V}$ has a joint distribution $\mathcal{P}_\mathcal{V}$ implied by $\mathcal{F}$ and $\mathcal{P}_{(\cdot)}$.

An SCM induces a directed acyclic graph called a causal graph, $G = (\mathcal{V}, \mathcal{E})$. Here $\mathcal{V}$ is the set of observable nodes and $\mathcal{E}$ is the set of directed edges. For any pair $V_i, V_j$, a directed edge $V_i \rightarrow V_j \in \mathcal{E}$ indicates that $V_i$ is a parent of $V_j$, i.e., $V_i \in Pa(V_j)$. $V_j$ 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 $f_i$. There exists a bi-directed edge $V_i \leftrightarrow V_j \in \mathcal{E}$ in $G$ if $V_i$ and $V_j$ are independent given their parents.


dependence on any mechanism that is sensitive to the distribution shift. Among features, we use $A, I, X, H$ and represent the shift with a discrete variable $S$ pointing to $H$. 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 ([29,31]), 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 $P(N \mid \text{do}(H), A, I, X)$ can be uniquely calculated from training data and $\text{do}(H)$ d-separates $N$ from $S$. Thus we can train our model with predictors $I, X, H, A$ from a hypothetical $\text{do}(H)$ 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-


dependence between two sets of variables.


dependence between $N$ and $H$ i.e., unobserved common causes of two observable variables, a set of functions $\mathcal{F}$ and a product probability distribution $\mathcal{P}_{(\cdot)}$ over $\mathcal{N}$ and $\mathcal{U}$. Each observed variable is generated as $V_i \equiv f_i(Pa_i, E_i, U_{S_i})$, where $f_i \in \mathcal{F}$, $Pa_i \subseteq \mathcal{V}$, $E_i \in \mathcal{N}$ and $U_{S_i} \equiv \{U_j : j \in S_i\}$ for some $S_i \subseteq \mathcal{U}$. Variables set $\mathcal{V}$ has a joint distribution $\mathcal{P}_\mathcal{V}$ implied by $\mathcal{F}$ and $\mathcal{P}_{(\cdot)}$.

An SCM induces a directed acyclic graph called a causal graph, $G = (\mathcal{V}, \mathcal{E})$. Here $\mathcal{V}$ is the set of observable nodes and $\mathcal{E}$ is the set of directed edges. For any pair $V_i, V_j$, a directed edge $V_i \rightarrow V_j \in \mathcal{E}$ indicates that $V_i$ is a parent of $V_j$, i.e., $V_i \in Pa(V_j)$. $V_j$ 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 $f_i$. There exists a bi-directed edge $V_i \leftrightarrow V_j \in \mathcal{E}$ in $G$ if $V_i$ and $V_j$ are independent given their parents.
vertices \( X \) represent the nodes that are either parent, children of \( V \), or share a directed edge with \( V \). For a variable set \( V \), \( Pa(V) = \{ Pa(V_i) \}_{V_i \in V} \setminus V \). \( Ch(V) \) also follows the same. However, \( An(V) \), \( Do(V) \) and \( Nbr(V) \) has set \( V \) included. We let \( G_S \) to denote an induced subgraph of \( G \) over any subset \( S \) of node \( V \), \( G_S \) be the graph obtained by removing the incoming edges to \( S \) from \( G_S \), and \( G_S \) be \( G_S \) with all outgoing edges of \( S \) removed. We define an intervention as \( do(x) \) where \( do(x) \) 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 \( P_x(\mathbf{V}) \) and the causal graph as \( G_{\mathbf{z}} \). Let \( (X, Y, Z) \) 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 \( Z \{X, Y \notin Z \} \) if \( i \) every non-collider on \( p \) is not in \( Z \) and \( ii \) every collider on \( p \) is an ancestor of some \( Z \in Z \). If there is no d-connecting path between \( X \) and \( Y \) relative to \( Z \), we say \( X \) and \( Y \) are d-separated relative to \( Z \), denoted as \( (X \perp \perp Y | Z)_G \).

**Definition 2.3 (Causal Effect Identifiability [25]).** Let \( X, Y, Z \) be disjoint sets. The causal effect of an action \( do(x) \) on a set of variables \( Y \) in a given context \( z \) is said to be identifiable from \( P \) in \( G \) if \( P_x(y | z) \) 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 \( X \) and a target variable \( Y \), distribution shift can be categorized into sub-groups (131) based on assumptions about the training domain and test domain. For example: \( i \) \( P(Y) \) changes while \( P(Y | X) \) stays fixed (target shift), \( ii \) \( P(Y | X) \) changes while \( P(Y) \) stays fixed (conditional shift), \( iii \) only \( P(X) \) 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 \( S \) (133) pointing to \( H \). Suppose 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 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 \( Q, W \subseteq V \), if \( (Y \perp \perp S \mid W)_{G_{\mathbf{z}}} \) and \( P(Y | do(Q), W) \) is identifiable in \( G \) and \( P(Y | do(Q), W) \neq P(Y) \), then \( P(Y | do(Q), W) \) 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, \( P(Y | do(X), H, T) \) is a graph surgery estimator because \( P(Y | do(X), H, T) \) is identifiable and \( (Y \perp \perp S \mid H)_{G_{\mathbf{z}}} \). Note that \( P(Y | do(X), H, T) \neq P(Y) \), i.e., the intervention \( do(X) \) 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 \( P(N | do(H), A, I, X) \) is a graph surgery estimator since the query is identifiable and \( (N \perp \perp S \mid A, I, X)_{G_{\mathbf{z}}} \). 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 \( Y \) that has no children. Then \( G \) is called a \( Y \)-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, \( G_{Z,R,W} \) is 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 \( X, Y \) be sets of variables in \( G \). A path \( p \) between \( X \) and \( Y \) is called an inducing path if every non-endpoint vertex is a collider on the path and an ancestor of either \( X \) or \( Y \).

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

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

3 FINDING GRAPH SURGERY ESTIMATORS IN POLYNOMIAL TIME

In this section, we describe the details of our approach of finding graph surgery estimators. We first 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 A.

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

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

We can use Figure 2 to illustrate Definition 3.1. Let \( X = \{Q, T\}, W = \{R, W\}, Y = \{Y, Z\} \). By rule 2 of do-calculus, \( P(Y,Z|do(Q,T), R, W) = P(Y,Z|do(Q,T, W), R) \). 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 \( P(Y,Z|do(Q,T), R, W) \). The following theorem describes the relationship between a hedge and causal identifiability.

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

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

In Figure 2, for variable \( Z, K = An(Z)_{G_K} = C(Z)_{G_K} = \{R, Z\} \) is an ACS for \( Z \) while \( K' = \{Q, R, Z\} \) is the largest set satisfying the same ACS condition. Thus, \( T_Z = K' = \{Q, R, Z\} \) 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 \( K \) in \( G \) as \( T_K \). 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 \( Y \) of nodes in \( G \) is an ancestral confounded component (AC-component) if \( Y \) is a singleton e.g. \( Y = \{Y\} \) or \( Y \) is a union of two distinct AC-components \( Y_1, Y_2 \) 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 \( Y = \{Y\} \). The output of Find-MACS-on-set \( G, Y \) 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 \( H, J \subseteq V \), we have \( Y \nabla S, J \) or \( P(Y|do(H), J) \) 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 MACs. 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 $K = H$ (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 $G$, $H := \{ H : H \in Ch(Y), Pa(T_H) \neq S \}$ and $T_J := \bigcup_{H \in R} T_H$ for any $K \subseteq H$, where $T_H$ is the MACS with respect to the variable $H$. Let $D = Pa(T_Y \cup T_J)$. If $S$ is not a parent of $T_Y$, then $P(Y|do(D), K, W)$ is identifiable in $G$ and $Y \perp S|W, K \cap G$ for any $W \subseteq (T_Y \cup T_J) \setminus (Y \cup K)$.

Corollary 3.9. Let $T_Y$ be the MACS of $Y$ in $G$ and $D = Pa(T_Y) \setminus Y$. If $S$ is not a parent of $T_Y$, then $P(Y|do(D), W)$ is identifiable in $G$ and $Y \perp S|W, K \cap G$ for any $W \subseteq T_Y \setminus Y$.

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

$$T_C := \bigcup_{H \in Ch(Y)} T_H$$

$$Z := \{ Z : Z \in (C(Y) \cap Nbr(Y)) \setminus (T_Y \cup T_C) \}$$

$$T_B := \bigcup_{Z \in M} T_Y \cup Z$$

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

### 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 3 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. This indicates that there is no graph surgery estimator in $G$.

At lines 7 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 $Z$ and $R$ change. Conditioning on the variable set $Z$ allows us to find more invariant predictors. $R$ 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, $H$ 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 $Z$ is non-empty, for each variable $H \in Z$, we call the sub-routine Find-MACS-on-set for $R \cup H$ (line 6). This
1: **Input:** Selection variable $S$, target $Y$, causal graph $G = (V, E)$
2: **Output:** An invariant predictor $P^*$ or FAIL to indicate that there is no graph surgery estimator in $G$.
3: $P_{set} = \emptyset; L_{set} = \emptyset$
4: $T_Y = \text{Find-MACS-on-set}(G, Y)$
5: if $S \in Pa(T_Y)$ then
6: Return FAIL
7: $P_{set}, L_{set}, T_Y = \text{addBestIP}(S, Y, T_Y, P_{set}, L_{set}, \emptyset, \emptyset)$
8: $P_{set}, L_{set}, T_C = \text{addBestIP}(S, Y, T_Y, P_{set}, L_{set}, \text{Ch}(Y), \emptyset)$
9: $T = T_Y \cup T_C$
10: $P_{set}, L_{set}, T_C = \text{addBestIP}(S, Y, T_Y, P_{set}, L_{set}, (C(Y) \cap \text{Nbr}(Y)) \setminus T, \{Y\})$
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 $R$ 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 all 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 subroutine 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 $Z = \text{Ch}(Y)$ as a parameter. At Algorithm $2$ lines $5, 10$ we iterate over $Z$ 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 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 $3$, $Ch(Y) = \{Z\}$ and $T_C = \{A, C\}$, $T_Z = \{E, Y, Z\}$. We can not utilize child $C$ for invariant predictors since $S \notin Pa(T_C)$. 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 T. Finally, at line $10$ we only consider specific bi-directed neighbors $N$ of $Y$ that are not in $T$, such that $N = C(Y) \cap \text{Nbr}(Y) \setminus T$. 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 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

![Causal Graph Example](image_url)
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 $P(A_{W}|do(X))$ where $A_{W}$ 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) [22] Let $|Ch(S)| = C$, $M = Ch(S)$, $Q = V \setminus (M \cup Y)$. Given a causal graph $G = (V, E)$ and disjoint variables $X, Y \subseteq V$, the time complexity of Graph Surgery Estimator (GSE) (Algorithm 5 in Section B.4) is $O(2^{|V| - C} - 1 \times B)$, 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.

**Theorem 4.2.** [23] Find-MACS-on-set $(G, Y)$ outputs the MACS of $Y$ in polynomial time in the size of graph.

**Theorem 4.3.** (ID4IP Complexity) Given a causal graph $G = (V, E)$ and disjoint variables $X, Y \subseteq V$, the complexity of ID4IP (Algorithm 3) is $O(\left| (\mathcal{C}(Y) \cap \text{Nbr}(Y)) \setminus (T_Y \cup T_C) \right| + |Ch(Y)| + 1)K + (|T_Y| - 1 + |T_J| + |T_J| - |\mathcal{H}| - |\mathcal{H}||B)$, 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 $T_C := \bigcup_{H \in \mathcal{C}(Y)} T_H$.

## 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](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
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 BN Learner 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,
We set the target to be the variable Raf. 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 $\text{PIP}3, \text{PIP}2, \text{Plcg}$ 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 \textit{DIFFICULTY} to be the selection variable and \textit{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 \textit{HISTORY}, \textit{LVEDVOLUME}, \textit{STROKEVOLUME}, \textit{CVP}, and \textit{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.

### 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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Sachs (11 nodes)</th>
<th>Alarm (37 nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSE</td>
<td>0.80</td>
<td>0.57</td>
</tr>
<tr>
<td>ID4IP</td>
<td>0.80</td>
<td>0.83</td>
</tr>
<tr>
<td>LR</td>
<td>0.53</td>
<td>0.52</td>
</tr>
</tbody>
</table>

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

![Figure 6: Modified Sachs causal graph](image)

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.

### 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 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


