Appendix A. Causal Discovery Algorithms

A.1. PC and FCI

PC algorithm is a constraint-based algorithm. That is, it learns a set of causal graphs that satisfy the conditional independencies embedded in the data at hand. There are two main steps. The first (lines 1-1 in Algorithm 1) takes as input the data at hand along with a significance level α and outputs a skeleton graph witch contains only undirected edges. The second (lines 1-1 in Algorithm 1) consists of orienting the undirected edges of the skeleton graph to form an equivalence class of DAGs. Note that the first step contributes to most of the computational costs. The PC-algorithm is proved to be efficient for sparse graphs. The main reason for that is that the neighbors of a particular node are dynamically updated (line 1 in Algorithm 1) once and edge is deleted Le et al. (2016).

The FCI algorithm Spirtes et al. (1999)(Algorithm 2) is also a constraint-based algorithm and is considered as a generalization of the PC algorithm. The main difference between PC and FCI is that the latter takes into account the presence of common hidden confounders between observed variables. Consequently, instead of producing a PDAG, the output of FCI is a partial ancestral graph (PAG) with possibly five types of edges: $\longrightarrow, \longleftrightarrow, \circ\!\!-\!\!, \circ\!\!-\!\!\circ, \circ\!\!\rightarrow$. The "$\circ$" mark represents undetermined edge mark. In other words, "$\circ$" can be either a tail "–" or a head "$\rightarrow$". $\longleftrightarrow$ shows that there are hidden confounders between the two variables on either side of the arrow. $X \circ\!\!\rightarrow Y$ implies that either X causes Y or there are hidden confounders between both variables. $X \circ\!\!-\!\!\circ Y$ might be: X causes Y, Y causes X, there are common hidden confounders between both variables, X causes Y and there are hidden confounders between both variables, or Y causes X and there are hidden confounders between both variables. As in the first step of the PC algorithm, FCI relies on statistical independence tests to infer the skeleton of the graph. It is in the second step that FCI deviates from the PC algorithm.

A.2. GES algorithm

GES (Algorithm 3) consists of searching over an abstract search space of states and transitions. Each state is a CPDAG that corresponds to a Markov equivalence class of DAGs, all of which happen to have the same BIC score Haughton (1988). The search objective is the state that maximizes BIC score, hence, the output of GES is not a single DAG, but an equivalence class of DAGs represented as a CPDAG.

The transitions of the search space are given by the following rule: a transition from a state to another exists if and only if there are two DAGs, one on each equivalence class, that differ only in the addition or removal of exactly one edge. Hence, there are two types of transitions: forward (adding one edge) and backward (removing one edge). The neighboring states for the state $\mathcal{P}$ are represented with the variable neighbors. The explicit computation of the neighboring states is illustrated and discussed in Chickering (2002); Dor and Tarsi (1992); Gamella (2021). The change in BIC score after following a transition can be computed using a simple rule instead of fitting the whole global model on both states because the BIC score can be decomposed as the sum of the local BIC scores of each of its directed and undirected parents. This optimization corresponds to $\Delta\mathsf{BIC}(\mathcal{P}, \mathcal{P}', \mathcal{D})$ in Algorithm 3. The

Algorithm 1: PC algorithm.

Input: Dataset $\mathcal{D}$, and significance level α.
Output: PDAG $\mathcal{P}$.

$G \leftarrow$ totally connected (undirected) skeleton
$d \leftarrow 0$
while $|adj_G(X) \setminus Y| \geq d$ *for every pair od adjacent vertices* $X - Y$ *in* G **do**
  for *each adjacent pair* $X - Y$ *in* G **do**
    if $(|adj_G(X) \setminus Y| \geq d)$ **then**
      for *each* $Z \subseteq adj_G(X) \setminus Y$ **do**
        if $|Z| = d$ *and* $I(X, Y|\boldsymbol{Z}) \geq \alpha$ **then**
          Remove edge $X - Y$ in G
          Save $\mathbf{Z}$ as the separating set of $X - Y$
          break
        end
      end
    end
  end
  $d \leftarrow d + 1$
end
$\mathcal{P} \leftarrow G$
for *each triple of vertices* (X, C, Y) *such that* $C \in adj_{\mathcal{P}}(X)$ *and* $Y \notin adj_{\mathcal{P}}(X)$ **do**
  if $C \notin \boldsymbol{Z}$ *(separating set of* $X - Y$*)* **then**
    orient $X - C - Y$ as $X \rightarrow C \leftarrow Y$ in $\mathcal{P}$
  end
end
while *unoriented edges exist* **do**
  for *each* (X, C, Y) *with* $X \rightarrow C - Y$ *and* $Y \notin adj_{\mathcal{P}}(X)$ **do**
    orient $C - Y$ as $C \rightarrow Y$ in $\mathcal{P}$; // Rule 1
  end
  for *each chain* $X \rightarrow C \rightarrow Y$ **do**
    orient $X - Y$ as $X \rightarrow Y$ in $\mathcal{P}$; // Rule 2
  end
  for *each pair of chains* $X \rightarrow C_1 \rightarrow Y$ *and* $X \rightarrow C_2 \rightarrow Y$ *such that* $C_2 \notin adj_{\mathcal{P}}(C_1)$ **do**
    orient $X - Y$ as $X \rightarrow Y$ in $\mathcal{P}$; // Rule 3
  end
end
return $\mathcal{P}$

Algorithm 2: FCI algorithm.

Input: Dataset $\mathcal{D}$, and significance level α.
Output: Partial ancestral graph (PAG).

$G \leftarrow$ skeleton found by PC
orient each edge in G as $\circ\!\!-\!\!\circ$
for *each unshielded triple* (X, C, Y) **do**
  if $C \notin \boldsymbol{Z}$ *(separating set of* (X, Y)*)* **then**
    orient the edges $X \ast\!\!-\!\!\ast C \ast\!\!-\!\!\ast Y$ as $X \ast\!\!\rightarrow C \leftarrow\!\!\ast Y$
  end
end
repeat
  if $X \ast\!\!\rightarrow C \circ\!\!-\!\!\ast Y$*, and* $Y \notin adj_G(X)$ **then**
    orient the triple as $X \ast\!\!\rightarrow C \rightarrow Y$; // Rule 1
  end
  if $X \rightarrow C \ast\!\!\rightarrow Y$ *or* $X \ast\!\!\rightarrow C \rightarrow Y$*, and* $X \ast\!\!-\!\!\circ C$ **then**
    orient $X \ast\!\!-\!\!\circ C$ as $X \ast\!\!\rightarrow C$; // Rule 2
  end
  if $X \ast\!\!\rightarrow C\ast \leftarrow\!\!\ast Y$*,* $X \ast\!\!-\!\!\circ D \circ\!\!-\!\!\ast Y$*,* $Y \notin adj_G(X)$*, and* $D \ast\!\!-\!\!\circ C$ **then**
    orient $D \ast\!\!-\!\!\circ C$ as $D \ast\!\!\rightarrow C$; // Rule 3
  end
  if $\pi = \langle D, ..., X, C, Y \rangle$ *is a discriminating path between* D *and* Y *for* C*, and* $C \ast\!\!-\!\!\circ Y$ **then**
    if $C \notin Sepset(D, Y)$ **then**
      orient $C \circ\!\!-\!\!\ast Y$ as $C \rightarrow Y$; // Rule 4
    else
      orient the triple $\langle X, C, Y \rangle$ as $X \leftrightarrow C \leftrightarrow Y$
    end
  end
until *none of the above rules applies* ;

greedy strategy of GES consists of repeatedly following the best forward transition at each state that it encounters until reaching a local maximum, i.e. until the next state reduces the BIC score, (this is the forward phase in Algorithm 3) and then, analogously (backward phase), repeatedly following the best backward transition until a local maximum is reached. The distinctive essential feature of GES is that its greedy technique, which prunes the search

Algorithm 3: GES algorithm.
Input: Dataset $\mathcal{D}$ of $|V|$ variables.
Output: CPDAG $\mathcal{P}$ that maximizes BIC score.
$\mathcal{P} \leftarrow$ disconnected CPDAG of $|V|$ nodes
score $\leftarrow 0$
for phase $\in$ [forward, backward] **do**
  while True **do**
    neighbors $\leftarrow \{\mathcal{P}' : \mathcal{P} \rightarrow \mathcal{P}'$ is a phase-transition$\}$
    if |neighbors|$= 0$ **then**
      break
    end
    $\mathcal{P}' \leftarrow \underset{\mathcal{P}' \in \text{neighbors}}{\arg\max} \Delta\text{BIC}(\mathcal{P}, \mathcal{P}', \mathcal{D})$
    Δscore $\leftarrow \Delta\text{BIC}(\mathcal{P}, \mathcal{P}', \mathcal{D})$
    if Δscore < 0 **then**
      break
    end
    $\mathcal{P} \leftarrow \mathcal{P}'$
    Add Δscore to score
  end
end
return $\mathcal{P}$, score

Algorithm 4: Direct LiNGAM.
Input: Dataset with data columns $\bar{X}, \bar{Y}, ..., \bar{Z}$ representing k variables $X, Y, ..., Z$, and threshold $\alpha > 0$
Output: DAG with weights matrix $W_{X \rightarrow Y}$
$S \leftarrow [\,]$ Empty causal order list
while $|S| < k$ **do**
  $X = \underset{X \notin S}{\arg\min} \sum_{Y \notin S \cup \mathcal{X}} I(X; r_{\bar{X} \rightarrow \bar{Y}})$
  Push X to the end of S
  for $Y \notin S \cup \mathcal{X}$ **do**
    $\bar{Y} \leftarrow r_{\bar{X} \rightarrow \bar{Y}}$; // Remove the effect of X on Y
  end
end
$W_{X \rightarrow Y} \leftarrow 0$ for all $X, Y \in S$
for $Y \in S$ **do**
  pa $\leftarrow \mathcal{X} : \mathcal{X}$ precedes $\mathcal{Y}$ in $\mathcal{S}$
  $W_{\text{pa} \rightarrow Y} \leftarrow$ linear coefficients $(Y = f(\text{pa}))$
  Set small values in $W_{\text{pa} \rightarrow Y}$ to 0 (if abs.$< \alpha$)
end
return W

space dramatically, is guaranteed to find the optimal state of the whole space, provided that the data matches the statistical model.

A.3. LiNGAM algorithm

LiNGAM is an algorithm based on causal asymmetries that, unlike the previously discussed algorithms, yields a unique directed graph (DAG) and corresponding parameters. However, the stronger causal discovery power comes at the expense of more assumptions that have to be satisfied.

LiNGAM requires linearity and non-gaussianity of the variables to recover causal directions and learn functional relationships Shimizu et al. (2006). If the assumptions are satisfied, causal direction between two variables can be determined by fitting linear regression and measuring the independence between the cause variable X , and the residuals $r_{X \rightarrow Y}$ of the effect variable Y when predicted using X. Mutual information is usually used as a metric for independence Hyvärinen and Smith (2013), although other metrics have been proposed Shimizu (2014).

The DirectLiNGAM implementation (Algorithm 4) learns the causal graph in two steps. First, it finds the causal order of the variables: an ordered list, where the first is the exogenous variable (has no parents in the graph), the second is the child of the exogenous variable, that has the most descendants etc. Next, the causal order is used to compute the adjacency

matrix that specifies the strength of the connections. Specifically, starting from the end of the list, each variable is regressed on all the others that comes before it in the causal order (potential parents).

A.4. SBCN Algorithm

A Suppes-Bayes Causal Network (SBCN) Bonchi et al. (2017) is a different type of causal graph that is used specifically for fairness assessment purposes. SBCN deviates from the causal graphs used above in three aspects. First, vertices in an SBCN correspond to Bernoulli variables with binary values. For example, $\langle$ Gender = female $\rangle$ and $\langle$ Gender = male $\rangle$ correspond to two different vertices. Second, causal relations between vertices follow the Suppes's definition of causality Hitchcock (2002); Suppes (1973) (different from the typical definition of causality Pearl (2009)) which requires temporal priority and probability raising. For example, a node a is a cause of a node y ($a \rightarrow y$) if and only if, a occurs before y (temporal priority) and the cause a raises the probability of the effect y, that is, $\mathbb{P}(y|a) > \mathbb{P}(y|\neg a)$ (probability raising). Third, every edge (causal relation) is assigned a weight corresponding to the confidence score. The weight is simply the extent of the probability raising ($W(a, y) = \mathbb{P}(y|a) - \mathbb{P}(y|\neg a)$). Discovering the SBCN structure from the data is a hybrid approach using constraint-based as well as score-based ideas.

Algorithm 5: SBCN
Input: Dataset $\mathcal{D}$ with a set of Bernoulli variables $\mathbf{V}$, and a partial order r of V.
Output: SBCN $= (V, E^*, W)$.
for *all pairs* $(x, y) \in V$ **do**
  if $r(y) \leq r(x)$ ***and*** $\mathbb{P}(x \mid y) > \mathbb{P}(x \mid \neg y)$ **then**
    add the edge (x, y) to SBCN
  end
end
Consider $G(V, E^*, W)_{fit} = \emptyset$
while ! $StoppingCriterion()$ **do**
  let $G(V, E^*, W)_{neighbors}$ be the neighbor solutions of $G(V, E^*, W)_{fit}$
  remove from $G(V, E^*, W)_{neighbors}$ any solution whose edges are not included in SBCN
  consider a random solution $G_{current}$ in $G(V, E^*, W)_{neighbors}$
  if $score_{BIC}(D, G_{current}) > score_{BIC}(D, G_{fit})$ **then**
    $G_{fit} = G_{current}$
    $\forall edge(x, y)$ of G_{fit}, $W(x, y) = \mathbb{P}(x \mid y) - \mathbb{P}(x \mid \neg y)$
  end
end
SBCN $= G_{fit}$
return SBCN