Constraint-Based Causal Structure Learning with Consistent Separating Sets

Honghao Li, Hervé Isambert
Institut Curie, Paris

Abstract

We consider constraint-based methods for causal structure learning, such as the PC algorithm or any PC-derived algorithms, whose first step consists in pruning a complete graph to obtain an undirected graph skeleton, which is subsequently oriented. During the first step, an edge between two variables is removed whenever the two variables are marginally or conditionally (given separating set) independent. Yet, the iterative removal of edges is not robust against sampling noise and is prone to spurious conditional independences in finite datasets. In particular, there is no guarantee that the separating sets identified during the iterative pruning step remain consistent with the final graph. In this paper, we propose a simple modification of PC and PC-derived algorithms so as to ensure that all separating sets of conditional independences are consistent with the final graph, whose explainability is thus enhanced. It is achieved by repeating the constraint-based causal structure learning scheme, iteratively, while searching for separating sets that are consistent with the graph obtained at the previous iteration. Ensuring the consistency of separating sets can be done at a limited complexity cost, through the use of block-cut tree decomposition of graph skeletons, and is found to increase their validity in terms of actual d-separation. It also improves the sensitivity of constraint-based methods while retaining good overall structure learning performance. Finally and foremost, ensuring separating set consistency improves the interpretability of constraint-based models for real-life applications. This article is an abridged version of Li et al. 2019.

Introduction

While the oracle versions of constraint-based methods have been demonstrated to be sound and complete (Zhang 2008; Spirtes, Glymour, and Scheines 2000; Pearl 2009), a major limitation of these methods is their lack of robustness with respect to sampling noise for finite datasets. This has largely limited their use to analyze real-life data so far, although important advances have been made lately, in particular, to limit the order-dependency of constraint-based methods (Colombo and Maathuis 2014) or to improve their robustness to sampling noise by recasting them within a maximum likelihood framework (Affeldt and Isambert 2015; Affeldt, Verny, and Isambert 2016).

However, it remains that constraint-based methods still lack graph consistency, in practice, as they do not guarantee that the learnt structures belong to their presumed class of graphical models, such as a completed partially directed acyclic graph (CPDAG) model for the PC (Spirtes and Glymour 1991; Kalisch and Bühlmann 2008; Kalisch et al. 2012) or IC (Pearl and Verma 1991) algorithms, or a partial ancestral graph (PAG) for FCI or related constraint-based algorithms allowing for unobserved latent variables (Spirtes, Meek, and Richardson 1999; Richardson and Spirtes 2002; Colombo et al. 2012; Verny et al. 2017; Sella et al. 2018). By contrast, search-and-score structure learning methods (Koller and Friedman 2009) inherently enforce graph consistency by searching structures within the assumed class of graphs, *e.g.*, within the class of directed acyclic graphs (DAG). Similarly, hybrid methods such as MMHC (Tsamardinos, Brown, and Aliferis 2006) can also ensure graph class consistency by maximizing the likelihood of edge orientation within the class of DAGs.

This paper concerns, more specifically, the inconsistency of separating sets used to remove dispensable edges, iteratively, based on conditional independence tests. This inconsistency arises as some separating sets might no longer be compatible with the final graph, if they were not already incompatible with the current skeleton, when testing for conditional independence during the pruning process. It occurs, for instance, when a node in a separating set is not on any indirect path linking the extremities of a removed edge, as noted in (Spirtes, Glymour, and Scheines 2000). Such inconsistencies can be seen as a major shortcoming of constraint-based methods, as the primary motivation to learn and visualize graphical models is arguably to be able to read off conditional independences directly from the graph structure (Spirtes, Glymour, and Scheines 2000; Pearl 2009).

In the following, we propose a simple modification of PC or PC-derived algorithms so as to ensure that all conditional independences identified and used to remove dispensable edges are consistent with the final graph. It is achieved by repeating the constraint-based causal structure learning scheme, iteratively, while searching for separating sets that are consistent with the graph obtained at the previous iteration, until a limit cycle of successive graphs is reached. The union of the graphs over this limit cycle is then guaranteed to be consistent with the separating sets

and corresponding conditional independences used to remove all dispensable edges from the initial complete graph. Enforcing sepset consistency of constraint-based methods is found to limit their tendency to uncover spurious conditional independences early on in the pruning process when the combinatorial space of possible separating sets is still large. As a result, enforcing sepset consistency reduces the large number of false negative edges usually predicted by constraint-based methods (Colombo and Maathuis 2014) and, thereby, achieve a better balance between their sensitivity and precision. Ensuring the consistency of separating sets is also found to increase their validity in terms of actual d-separation and, therefore, to improve the interpretability of constraint-based models for real-life applications. Moreover, ensuring the consistency of separating sets can be done at a limited complexity cost, through the use of block-cut tree decomposition of graph skeletons, which enables to learn causal structures with consistent separating sets for a few hundred nodes. By contrast, earlier methods aiming at reducing the number of d-separation conflicts or other structural inconsistencies through SAT-based approaches, *e.g.* (Hyttinen et al. 2013), have a much larger complexity burden, which limits their applications to very small networks in practice.

Results

Background

The PC algorithm (Spirtes and Glymour 1991) is the archetype of constraint-based structure learning methods (Spirtes, Glymour, and Scheines 2000; Pearl 2009), as illustrated in fig. 1. Given a dataset over a set of variables (vertices), it starts from a complete graph $\mathcal{G}_c$. By a series of statistical tests on each pair of variables, all dispensable edges $X - Y$ are removed if a (conditional) independence and separating set Sep_{XY} can be found, *i.e.* $(X \perp\!\!\!\perp Y \mid \text{Sep}_{XY})$ (step 1). The resulting undirected graph is called the **skeleton**. V-structures are then identified, $X \rightarrow Z \leftarrow Y$, if $(X \perp\!\!\!\perp Y \mid \text{Sep}_{XY})$ and $Z \notin \text{Sep}_{XY}$ (step 2). Additional assumptions (e.g., acyclicity) allow for the propagation of v-structure orientations to some of the remaining undirected edges (Zhang 2008) (step 3). While the oracle version of

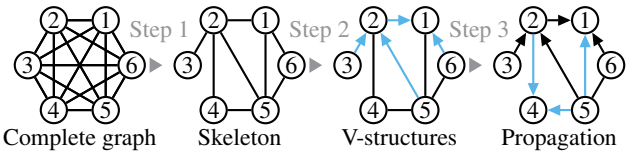

Figure 1: General procedure of constraint-based methods.

the PC-algorithm has been shown to be sound and complete, its application is known to be sensitive to the finite size of real life datasets. In particular, the PC-algorithm in its original implementation (Spirtes, Glymour, and Scheines 2000) is known to be order-dependent, in the sense that the output depends on the lexicographic order of the variables. This issue can be circumvented with a simple modification given in algorithm 1 and is referred to as step 1 of PC-stable algorithm (Colombo and Maathuis 2014).

Algorithm 1 Find skeleton and separating sets (step 1 of PC-stable algorithm)

Require: Conditional independence assessment with significance level α for all pairs of variables
$\mathcal{G} \leftarrow \mathcal{G}_c(\boldsymbol{V}, \boldsymbol{E})$
$\ell \leftarrow -1$
repeat
  $\ell \leftarrow \ell + 1$
  for all vertices $X_i \in \boldsymbol{V}$ **do**
    $a(X_i) \leftarrow \text{adj}(\mathcal{G}, X_i)$
  repeat
    Select a new pair $(X_i, X_j) \in \boldsymbol{E}$ such that $|a(X_i) \setminus \{ X_j \}| \geq \ell$
    $\text{Sep}_{X_iX_j} \leftarrow \emptyset$
    repeat
      Select a new set $\boldsymbol{C} \subseteq a(X_i) \setminus \{ X_j \}$ such that $|\boldsymbol{C}| = \ell$
      if $(X_i \perp\!\!\!\perp X_j \mid \boldsymbol{C})_\alpha$ **then**
        ▷ *Remove edge from* $\mathcal{G}$ $(X_i \nsim X_j)$ ◁
        $\boldsymbol{E} \leftarrow \boldsymbol{E} \setminus (X_i, X_j)$
        $\text{Sep}_{X_iX_j} \leftarrow \text{Sep}_{X_iX_j} \bigcup \boldsymbol{C}$
    until $X_i \nsim X_j$ or all sets $\boldsymbol{C}$ have been considered
  until all pairs $(X_i, X_j) \in \boldsymbol{E}$ have been considered
until all pairs $(X_i, X_j) \in \boldsymbol{E}$ satisfy $|a(X_i) \setminus \{ X_j \}| \leq \ell$
return $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$, $\{ \text{Sep}_{X_iX_j} \}$ for all pairs $(X_i, X_j) \in \boldsymbol{E}$

Lack of Robustness and Consistency of Constraint-Based Methods

Beyond the order-dependence of the PC Algorithm, the general lack of robustness of constraint-based methods stems from their tendency to uncover spurious conditional independences (false negatives) between variables. This trend originates from the fact that conditioning on other variables amounts to "slicing" the available data into smaller and smaller subsets, corresponding to different combinations of categories or discrete values of the conditioning variables, over which independence tests are essentially "averaged" to assess conditional independence.

Figure 2: Two scenarios of inconsistent conditional independence: $2 \perp\!\!\!\perp 6 \mid 3$ regarding the skeleton, where a path between 3 and 6 that does not go through 2 is expected but missing. And $3 \perp\!\!\!\perp 6 \mid 1$ regarding the partially directed graph, where 1 as the common descendant of 3 and 6 is not expected in the separating set.

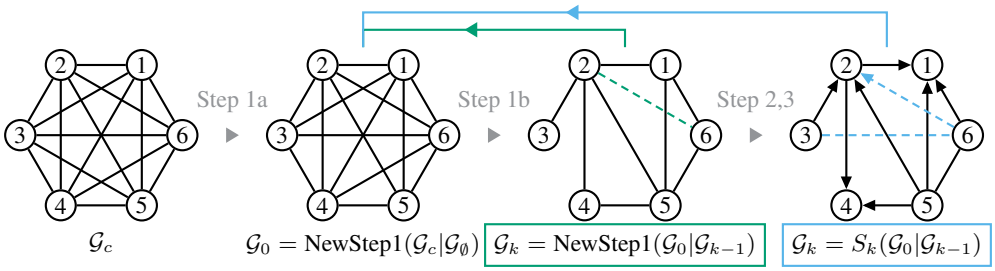

Figure 3: Illustration of the iterative algorithms to learn graphical models with either orientation-consistent or skeleton-consistent separating sets. In the skeleton-consistent scheme, the conditional independence $2 \perp\!\!\!\perp 6 \mid 3$ is not consistent in the skeleton of $\mathcal{G}_{k-1}$, thus the edge $2 - 6$ is retained in the skeleton of $\mathcal{G}_k$ (dashed green). In the orientation-consistent scheme, the conditional independence $3 \perp\!\!\!\perp 6 \mid 1$ is not consistent in $\mathcal{G}_{k-1}$, thus the edge $3 - 6$ is retained in $\mathcal{G}_k$ (dashed blue, together with $2 - 6$ which is now oriented).

Hence, by making sure that all separating sets are actually consistent with the final graph, one expects to reduce the number of false negative edges due to spurious conditional independences inferred during the edge pruning process and, thereby, to improve the sensitivity (or recall) of the PC or PC-stable algorithms.

The inconsistency of separating sets can be of different forms, regarding either the skeleton (type I) or the final (partially) oriented graph (type II), as illustrated in fig. 2.

A type I inconsistency corresponds to a conditional independence relation such as $(2 \perp\!\!\!\perp 6 \mid 3)$ in fig. 2, for which there is no path between vertex 2 and 6 that passes through 3. This type of inconsistency often involves edges evaluated early on in the pruning process when few edges have been removed, and thus the combinatorial space of possible separating sets is still large. In particular, edge $3 - 6$, which is eventually removed in the final graph, may still exist when the edge $2 - 6$ is under consideration.

A type II inconsistency is a different kind of incompatibility originating from the orientation of the skeleton. It occurs, in particular, when a conditional independence relation is conditioned on at least one common descendant of the pair of interest in the final graph, e.g., $(3 \perp\!\!\!\perp 6 \mid 1)$ in fig. 2. Since it stems from the orientation of edges (steps 2&3), the origin of type II inconsistencies is generally more complex and results from a cascade of errors in both conditional independence tests and orientation.

These two types of inconsistency help define the following consistent set for candidate nodes of separating sets in absence of latent variables:

Definition 1 (Consistent set)**.** Given a graph $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$ and a set of variables $\{ X, Y, Z \} \subseteq \boldsymbol{V}$,

$$\begin{aligned}\text{Consist}(X, Y \mid \mathcal{G}) = \{ Z \in \text{adj}(X) \setminus \{ Y \} \mid \\ 1.\ \text{at least one path } \gamma^Z_{XY} \text{ exists in } \mathcal{G}; \\ 2.\ Z \text{ is not a child of } X \text{ in } \mathcal{G} \}\end{aligned}$$

where γ^Z_{XY} is a path from X to Y passing through Z. Note that for an undirected graph, the second condition is always satisfied.

Consistent PC Pseudocodes

Definition 2. $\text{NewStep1}(\mathcal{G}_1|\mathcal{G}_2)$ is a modified version of PC-stable step 1 (algorithm 1) where we replace $\mathcal{G}_c$ by $\mathcal{G}_1$, and $a(X_i) \setminus \{X_j\}$ by $a(X_i) \setminus \{X_j\} \cap \text{Consist}(X_i, X_j \mid \mathcal{G}_2)$.

Note that algorithm $\text{NewStep1}(\mathcal{G}_c|\mathcal{G}_c)$ corresponds to the unmodified step 1 of original PC-stable algorithm 1. By constrast, algorithm $\text{NewStep1}(\mathcal{G}_c|\mathcal{G}_\emptyset)$ removes all edges corresponding to independence without conditioning, as no separating set is involved. This unconditional independence search will be noted **step 1a**, while the subsequent conditional independence search will be referred to as **step 1b**, thereafter.

Definition 3. $S(\mathcal{G}_1|\mathcal{G}_2)$ is a modified version of the PC-stable algorithm, where step 1 is replaced by $\text{NewStep1}(\mathcal{G}_1|\mathcal{G}_2)$ from definition 2.

Then, definition 3 allows to define algorithm 2, which ensures a consistent constraint-based algorithm through an iterative call of S algorithms, $(S_k)_{k\in\mathbb{N}^*}$, following an initial **step 1a**, $\text{NewStep1}(\mathcal{G}_c|\mathcal{G}_\emptyset)$. As illustrated on Figure 3 and proved below, algorithm 2 achieves separating set consistency by repeating **step 1b**, **2** and **3**, iteratively, while searching for separating sets that are consistent with the graph obtained at the previous iteration, until a limit cycle of successive graphs is reached.

Algorithm 2 Sepset consistent PC algorithm (1st version, orientation consistency)

Require: $\boldsymbol{V}, \mathcal{D}(\boldsymbol{V})$, significance level α
Ensure: $\mathcal{G}$ with consistent separating sets
  $\mathcal{G}_0 \leftarrow \text{NewStep1}(\mathcal{G}_c|\mathcal{G}_\emptyset)$
  $k \leftarrow 0$
  repeat
    $k \leftarrow k + 1$
    $\mathcal{G}_k \leftarrow S_k(\mathcal{G}_0|\mathcal{G}_{k-1})$
  until loop detected, *i.e.*, $\exists n > 0, \mathcal{G}_{k-n} = \mathcal{G}_k$
  $\mathcal{G} \leftarrow \bigcup (\mathcal{G}_j)_{j=k-n}^{k}$, discarding conflicting orientations
  return $\mathcal{G}$ and consistent separating sets

Alternatively, one may require a separating set consistency at the level of the skeleton only, *i.e.*, before the orienta-

Algorithm 3 Sepset consistent PC algorithm (2nd version, skeleton consistency)

Require: $\boldsymbol{V}, \mathcal{D}(\boldsymbol{V})$, significance level α
Ensure: $\mathcal{G}$ with consistent separating sets
  $\mathcal{G}_0 \leftarrow$ NewStep1$(\mathcal{G}_c|\mathcal{G}_\emptyset)$
  $k \leftarrow 0$
  repeat
    $k \leftarrow k+1$
    $\mathcal{G}_k \leftarrow$ NewStep1$(\mathcal{G}_0|\mathcal{G}_{k-1})$
  until loop detected, *i.e.*, $\exists n > 0, \mathcal{G}_{k-n} = \mathcal{G}_k$
  $\mathcal{G} \leftarrow \bigcup (\mathcal{G}_j)_{j=k-n}^{k}$ and consistent separating sets with respect to the graph skeleton $\mathcal{G}$
  Step 2 (orientation of v-structures in $\mathcal{G}$)
  Step 3 (propagation of orientations in $\mathcal{G}$)
  for all removed edges (X, Y) in $\mathcal{G}$ **do**
    Sepset$(X, Y \mid \mathcal{G}) \leftarrow$ Sepset$(X, Y \mid \mathcal{G}_k)$
    if Sepset$(X, Y \mid \mathcal{G}) \notin$ Consist$(X, Y \mid \mathcal{G})$ and Sepset$(X, Y \mid \mathcal{G}) \notin$ Consist$(Y, X \mid \mathcal{G})$ **then**
      Add undirected edge (X, Y) to $\mathcal{G}$
  return $\mathcal{G}$ and consistent separating sets

tion steps, which corresponds to algorithm 3, below. Indeed, early sepset inconsistencies at the level of the skeleton might cause orientation errors, which in turn can lead to the rejection of valid consistent separating sets in algorithm 2. As outlined in Figure 3, the modification of algorithm 3 only concerns step 1b, which is called iteratively until a limit cycle is reached. Then, the orientation steps 2&3 are performed as for classical PC or PC-derived algorithms, but using consistent separating sets with respect to the union of skeletons returned by the iterative call of step 1b in algorithm 3. However, as the orientation steps 2&3 might induce additional type II inconsistencies, algorithm 3 requires a final consistency check for all separating sets with respect to the final graph $\mathcal{G}$.

Theorem 4. *The separating sets returned by algorithms 2 and 3 are consistent with respect to the final graph $\mathcal{G}$.*

Proof. Firstly, the limit cycles in algorithms 2 and 3 are warranted to be finite by the deterministic nature of these algorithms and the finite set of graphs $\mathcal{G}_j$.

In algorithm 2, as the union of graphs $\bigcup (\mathcal{G}_j)_{j=k-n}^{k}$ does not remove any edge from the last graph $\mathcal{G}_k$ and discards all conflicting orientations with previous graphs $\mathcal{G}_j$, $j \in \{ k-n, k-1 \}$, taking the union of graphs does not create any new conditional independence relation, nor any inconsistency regarding the final separating sets. More precisely, all removed edges in $\mathcal{G}_k$ have separating sets consistent with respect to at least one graph in the union ($\mathcal{G}_{k-1}$), which is thus also consistent with respect to the union of graphs $\mathcal{G}$.

In algorithm 3, the consistency of separating sets is guaranteed by similar arguments, but only with respect to the skeleton. As the orientation and propagation steps 2&3 might induce additional type II inconsistencies, algorithm 3 requires a final consistency check for all separating sets. Adding back edges with inconsistent separating sets in the final graph $\mathcal{G}$ then guarantees that all the separating sets are consistent with respect to definition 1. □

Tests of Consistency

A unitary operation of algorithms 2 and 3 is to test, for a vertex $Z \in \text{adj}(X) \setminus \{ Y \}$ in $\mathcal{G}$, if $Z \in \text{Consist}(X, Y \mid \mathcal{G})$, which requires that (1) at least one path from X to Y passing through Z (*i.e.* γ_{XY}^{Z}) exists in $\mathcal{G}$ and (2) Z is not a child of X in $\mathcal{G}$ (definition 1).

To test the first condition, it is conceptually simple to first get all paths between X and Y, then check if Z lies in at least one of them, This is however unfeasible as the complexity of getting all paths between two vertices can be large, depending on the edge density of the graph. Fortunately, it is possible to get directly the set of all Z for which at least on path γ_{XY}^{Z} exists. This can be done very efficiently with the help of biconnected component analysis based on block-cut tree decomposition.

To begin with, we provide the terminology necessary in the following discussion. A **connected graph** $\mathcal{G}$ is such that there is a path between each pair of vertices of $\mathcal{G}$. A **connected component** of a graph is a maximal connected subgraph. An **articulation point** (or **cut point**) is a vertex in a connected graph whose removal would disconnect the graph and thus increase its number of connected components. A **biconnected graph** is a connected graph without articulation point. A **biconnected component** (or **block**) is a maximal biconnected subgraph.

For a pair (X, Y) in a graph $\mathcal{G}$, one of the necessary conditions for its separating set to be consistent, as stated in definition 1, is that for each vertex Z in the separating set, Z lies on a path γ_{XY}^{Z} between X and Y in the skeleton of $\mathcal{G}$. For one pair of vertices, checking the existence of a path for all Z can already be time consuming if the degrees of the vertices are large. In addition, the complexity will be further multiplied by the number of pairs to be considered. Fortunately, it is possible to avoid this high complexity with the help of the biconnected component analysis based on block-cut tree decomposition, and thus to limit the search of consistent separating vertices within those that are consistent with respect to the skeleton.

Definition 5 (Block-cut tree)**.** The block-cut tree decomposition of a connected graph $\mathcal{G}$ is denoted by $\mathcal{T}(\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{Br})$ where $\boldsymbol{B} = \{ b_i \}_{i=1}^{m}$ is the set of biconnected components (or blocks) of $\mathcal{G}$, $\boldsymbol{C} = \{ c_j \}_{j=1}^{n}$ is the set of articulation points (or cut points) and $\boldsymbol{Br} = \{ (b_i, c_j) \mid b_i \in \boldsymbol{B}, c_j \in \boldsymbol{C}, b_i \text{ and } c_j \text{ are adjacent in } \mathcal{T} \}$ is the set of connections between $\boldsymbol{B}$ and $\boldsymbol{C}$.

In the following we establish a relation between biconnected components and the path existence problem.

Lemma 6 (Menger's theorem for biconnected graph)**.** *Let $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$ be a biconnected graph, $\{ X, Y \} \subseteq \boldsymbol{V}$ a pair of vertices. There is a cycle in $\mathcal{G}$ that contains X and Y.*

Theorem 7. *Let $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$ be an undirected graph, $\mathcal{H}(\boldsymbol{V}_\mathcal{H}, \boldsymbol{E}_\mathcal{H}) \subseteq \mathcal{G}$ a biconnected component of $\mathcal{G}$, $\{ X, Y \} \subseteq \boldsymbol{V}_\mathcal{H}$ a pair of vertices, and $Z \in \boldsymbol{V}_\mathcal{G}$ a third vertex. There is a path γ_{XY}^{Z} if and only if $Z \in \boldsymbol{V}_\mathcal{H}$.*

Proof. If there is a path γ_{XY}^{Z}, suppose that $Z \notin \boldsymbol{V}_{\mathcal{H}}$, then the subgraph $\mathcal{H}'$ of $\mathcal{G}$ over $\boldsymbol{V}_{\mathcal{H}} \cup \{ Z \}$ is biconnected thanks to γ_{XY}^{Z}, and $\mathcal{H} \subset \mathcal{H}'$ is not a biconnected component of $\mathcal{G}$ as it is not maximal. Therefore we must have $Z \in \boldsymbol{V}_{\mathcal{H}}$.

If $\{ X, Y, Z \} \subseteq \boldsymbol{V}_{\mathcal{H}}$, then lemma 6 guarantees a cycle that contains Z and Y. Since $\boldsymbol{V}_{\mathcal{H}}$ contains at least three vertices, such a cycle contains $n \geq 1$ vertices other than Z and Y, and can be represented by two edge-distinct paths between Z and Y:

$$\gamma_{ZY}^{(1)} = ZU_1U_2 \cdots U_kY, \quad \gamma_{ZY}^{(2)} = ZU_{k+1}U_{k+2} \cdots U_nY$$

where $k \in \mathbb{Z}^{\geq 0}$ (with $k = 0$ indicating a direct edge between Z and Y), $n \in \mathbb{Z}^{+}, k < n$ and $\{ U_i \}_{i=1}^{n}$ are distinct vertices. Since Y is not an articulation point, there is a path γ_{XZ} that does not contain Y:

$$\gamma_{XZ} = XD_1D_2 \cdots D_mZ$$

where $m \in \mathbb{Z}^{\geq 0}$ and $\{ D_j \}_{j=1}^{m}$ are distinct vertices. If $\{ U_i \}_{i=1}^{n} \cap \{ D_j \}_{j=1}^{m} = \emptyset$, then there is a path

$$\gamma_{XY}^{Z} = \gamma_{XZ}\gamma_{ZY}^{(i)}, i \in \{ 1, 2 \}.$$

Otherwise, suppose $\{ U_i \}_{i=1}^{n} \cap \{ D_j \}_{j=1}^{m} = \{ D_{p_1}, D_{p_2}, \ldots, D_{p_t} \}$ where $t \in \mathbb{Z}^{+}$ and $p_1 < p_2 < \cdots < p_t$, and suppose $D_{p_1} = U_l$. If $l \leq k$, then there is a path

$$\gamma_{XY}^{Z} = XD_1D_2 \cdots D_{p_1}(U_l)U_{l-1} \cdots U_1\gamma_{ZY}^{(2)},$$

if $l > k$, then there is a path

$$\gamma_{XY}^{Z} = XD_1D_2 \cdots D_{p_1}(U_l)U_{l-1} \cdots U_{k+1}\gamma_{ZY}^{(1)}.$$

As a result, if $\{ X, Y, Z \} \subseteq \boldsymbol{V}_{\mathcal{H}}$, then there is always a path γ_{XY}^{Z}. □

Corollary 8. *Let $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$ be a connected graph, $\mathcal{T}(\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{Br})$ the block-cut tree decomposition of $\mathcal{G}$, $\{ X, Y \} \subseteq \boldsymbol{V}$ a pair of vertices, n_X, n_Y the corresponding nodes of X and Y in $\mathcal{T}$, and $\boldsymbol{S} = \{ Z \in \boldsymbol{V} \setminus \{ X, Y \} \mid$ at least one path γ_{XY}^{Z} exists. $\}$*

1. *If $n_X = n_Y = b_i \in \boldsymbol{B}$, then $\boldsymbol{S} = \boldsymbol{V}(b_i) \setminus \{ X, Y \}$.*
2. *If $n_X \neq n_Y$, let $\nu_{XY} = w_1w_2 \cdots w_k, w_1 = n_X, w_k = n_Y$ be the path between n_X and n_Y where each w_i belongs to $\boldsymbol{B}$ or $\boldsymbol{C}$, then $\boldsymbol{S} = \bigcup (\boldsymbol{V}(w_i))_{i=1}^{k} \setminus \{ X, Y \}$.*

The first case is a direct result of theorem 7. The second case is not difficult to prove once we notice the fact that ν_{XY} is the unique path between n_X and n_Y in $\mathcal{T}$, and that every γ_{XY} must contain all the cut points in ν_{XY}, and thus can be decomposed into segments of paths between these cut points.

Each undirected graph $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$ can be decomposed into a set of single vertices and a set of connected subgraphs, where each subgraph can be represented by a block-cut tree. Based on this decomposition, algorithm 4 gives the consistent candidate vertices for separating set for a pair of vertices as described in definition 1.

The block-cut tree decomposition can be done beforehand within a single depth first search with complexity $\mathcal{O}(|\boldsymbol{V}| + |\boldsymbol{E}|)$. Thus for each pair (X, Y), the complexity of finding all candidate Z depends on the size of the block-cut tree. In the worst case where $\mathcal{G}$ is a forest with only bridges (edges, the removal of each bridge increases the number of connected components of $\mathcal{G}$), the number of nodes and branches in the block-cut tree $\mathcal{T}$ of $\mathcal{G}$ is of the same order of $|\boldsymbol{V}|$ and $|\boldsymbol{E}|$, and for all pair of vertices $\{ X, Y \} \subseteq \boldsymbol{V}$ we need to perform a path search in $\mathcal{T}$ of complexity $\mathcal{O}(|\boldsymbol{V}|+|\boldsymbol{E}|)$ to get $\boldsymbol{S}$. In the best scenario where $\mathcal{G}$ is biconnected, $\boldsymbol{S} = \boldsymbol{V} \setminus \{ X, Y \}$ for all pairs. Then, an operation of set intersection $(\text{Ne}(X) \setminus \text{Child}(X)) \cap \boldsymbol{S}$ with linear complexity $\mathcal{O}(|\text{Ne}(X)| + |\boldsymbol{S}|)$ will give the result.

Algorithm 4 Consistent candidates

Require: (Partially directed) graph $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$, its block-cut tree decomposition for each connected component (with respect to its skeleton) $\{ \mathcal{T}_i(\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{Br}) \}$, two vertices $\{ X, Y \} \subseteq \boldsymbol{V}$
Ensure: Set of all candidate vertices Consist$(X, Y \mid \mathcal{G})$.
  if X and Y do not belong to the same block-cut tree $\mathcal{T}_i$ **then**
    return $\emptyset$
  if X and Y belong to the same block $b_i \in \boldsymbol{B}$ **then**
    return $(\text{Ne}(X) \setminus \text{Child}(X)) \cap (\boldsymbol{V}(b_i) \setminus \{ X, Y \})$
  else
    $\nu_{XY} \leftarrow \text{TreePath}(n_X, n_Y) = w_1w_2 \cdots w_k$
    return $(\text{Ne}(X) \setminus \text{Child}(X)) \cap (\bigcup (\boldsymbol{V}(w_i))_{i=1}^{k} \setminus \{ X, Y \})$

Conclusion

In this paper, we propose simple modifications of the PC algorithm also applicable to any PC-derived constraint-based methods, in order to enforce the consistency of the separating sets of discarded edges with respect to the final graph, which is an actual shortcoming of constraint-based approaches.

The existence of sepset inconsistencies with constraint-based methods originates from their tendency to uncover spurious conditional independences early on in the pruning process when the combinatorial space of possible separating sets is still large, unlike in the final typically sparse skeleton. Such spurious conditional independences are responsible, in particular, for the large number of false negative edges and, therefore, frequently poor sensitivity of constraint-based methods (Colombo and Maathuis 2014). By contrast, enforcing sepset consistency enables to achieve a better balance between sensitivity and precision.

To circumvent this inconsistency issue during the skeleton step, we have shown that one can either use sepset consistency taking into account orientations to help reject inconsistent sepsets (algorithm 2) or use sepset consistency of the skeleton to help determine the orientations (algorithm 3). The later approach tends to yield slightly better performance with the setting of the PC-stable algorithm used here (Li et al. 2019) but this is expected to be dependent on the specific settings used, for conditional independence test, orientation and propagation rules, in different constraint-based

methods.

Indeed, the methods and algorithmic implementations presented here are not primarily meant to outcompete a specific PC or PC-derived algorithm but rather to improve the explainability of constraint-based methods, by ensuring the consistency of all separating sets in the final causal graphs.

The approach is very general and applicable to the large variety of constraint-based methods, starting with a complete graph and discarding dispensable edges iteratively based on conditional independence search. Beyond the formal interest of guaranteeing sepset consistency, this is also especially important, in practice, for the interpretability of constraint-based models for real-life applications.

References

Affeldt, S.; and Isambert, H. 2015. Robust reconstruction of causal graphical models based on conditional 2-point and 3-point information. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015*, 42–51. doi:http://auai.org/uai2015/proceedings.shtml.

Affeldt, S.; Verny, L.; and Isambert, H. 2016. 3off2: A network reconstruction algorithm based on 2-point and 3-point information statistics. *BMC Bioinformatics* 17(S2). doi:10.1186/s12859-015-0856-x. URL http://dx.doi.org/10.1186/s12859-015-0856-x.

Colombo, D.; and Maathuis, M. H. 2014. Order-Independent Constraint-Based Causal Structure Learning. *Journal of Machine Learning Research* 15: 3741–3782. URL http://jmlr.org/papers/v15/colombo14a.html.

Colombo, D.; Maathuis, M. H.; Kalisch, M.; and Richardson, T. S. 2012. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Ann. Statist.* 40(1): 294–321. doi:10.1214/11-aos940. URL http://dx.doi.org/10.1214/11-AOS940.

Hyttinen, A.; Hoyer, P. O.; Eberhardt, F.; and Järvisalo, M. 2013. Discovering Cyclic Causal Models with Latent Variables: A General SAT-based Procedure. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, 301–310. Arlington, Virginia, United States: AUAI Press. URL http://dl.acm.org/citation.cfm?id=3023638.3023669.

Kalisch, M.; and Bühlmann, P. 2008. Robustification of the PC-Algorithm for Directed Acyclic Graphs. *Journal Of Computational And Graphical Statistics* 17(4): 773–789.

Kalisch, M.; Mächler, M.; Colombo, D.; Maathuis, M. H.; and Bühlmann, P. 2012. Causal inference using graphical models with the R package pcalg. *J. Stat. Softw.* 47(11): 1–26. doi:10.18637/jss.v047.i11.

Koller, D.; and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Li, H.; Cabeli, V.; Sella, N.; and Isambert, H. 2019. Constraint-based Causal Structure Learning with Consistent Separating Sets. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. URL http://papers.nips.cc/paper/9573-constraint-based-causal-structure-learning-with-consistent-separating-sets.pdf.

Pearl, J. 2009. *Causality: models, reasoning and inference*. Cambridge University Press, 2nd edition.

Pearl, J.; and Verma, T. 1991. A Theory of Inferred Causation. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, 441–452. Morgan Kaufmann Publishers Inc. ISBN 1-55860-165-1. URL http://dl.acm.org/citation.cfm?id=3087158.3087202.

Richardson, T.; and Spirtes, P. 2002. Ancestral graph Markov models. *Ann. Statist.* 30(4): 962–1030. doi:10.1214/aos/1031689015. URL http://dx.doi.org/10.1214/aos/1031689015.

Sella, N.; Verny, L.; Uguzzoni, G.; Affeldt, S.; and Isambert, H. 2018. MIIC online: a web server to reconstruct causal or non-causal networks from non-perturbative data. *Bioinformatics* 34(13): 2311–2313. doi:10.1093/bioinformatics/btx844. URL http://dx.doi.org/10.1093/bioinformatics/btx844.

Spirtes, P.; and Glymour, C. 1991. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review* 9: 62–72.

Spirtes, P.; Glymour, C.; and Scheines, R. 2000. *Causation, Prediction, and Search*. The MIT Press, Cambridge, Massachusetts, 2nd edition.

Spirtes, P.; Meek, C.; and Richardson, T. 1999. An Algorithm for causal inference in the presence of latent variables and selection bias. In *Computation, Causation, and Discovery*, 211–252. Menlo Park, CA: AAAI Press. doi:10.1.1.42.7098.

Tsamardinos, I.; Brown, L. E.; and Aliferis, C. F. 2006. The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning* 65(1): 31–78.

Verny, L.; Sella, N.; Affeldt, S.; Singh, P. P.; and Isambert, H. 2017. Learning causal networks with latent variables from multivariate information in genomic data. *PLoS Comput. Biol.* 13(10): e1005662.

Zhang, J. 2008. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artif. Intell.* 172(16-17): 1873–1896. URL http://dblp.uni-trier.de/db/journals/ai/ai172.html#Zhang08.