Partitioned hybrid learning of Bayesian network structures

Jireh Huang[1] · Qing Zhou[1]

Abstract

We develop a novel hybrid method for Bayesian network structure learning called partitioned hybrid greedy search (pHGS), composed of three distinct yet compatible new algorithms: Partitioned PC (pPC) accelerates skeleton learning via a divide-and-conquer strategy, p-value adjacency thresholding (PATH) effectively accomplishes parameter tuning with a single execution, and hybrid greedy initialization (HGI) maximally utilizes constraint-based information to obtain a high-scoring and well-performing initial graph for greedy search. We establish structure learning consistency of our algorithms in the large-sample limit, and empirically validate our methods individually and collectively through extensive numerical comparisons. The combined merits of pPC and PATH achieve significant computational reductions compared to the PC algorithm without sacrificing the accuracy of estimated structures, and our generally applicable HGI strategy reliably improves the estimation structural accuracy of popular hybrid algorithms with negligible additional computational expense. Our empirical results demonstrate the competitive empirical performance of pHGS against many state-of-the-art structure learning algorithms.

Keywords Bayesian networks · Directed acyclic graphs · Structure learning · Greedy search · PC algorithm · Divide-and-conquer

1 Introduction

Bayesian networks are compact yet powerful graphical models that efficiently encode in their graphical structures probabilistic relationships amongst a large number of variables (Neapolitan 2004). Despite their utility for probabilistic inference, the problem of recovering from data the structure of the true underlying Bayesian network that governs a domain of variables is notoriously challenging (Chickering et al. 2004). The space of Bayesian network structures grows super-exponentially with the number of variables,

severely limiting exhaustive evaluation of all structures and motivating decades of work in developing efficient algorithms for structure learning (Robinson 1977; Spirtes et al. 2000).

Generally, Bayesian network structure learning algorithms can be classified as one of the following three classes of algorithms. Constraint-based methods strategically test conditional independence relationships between pairs of variables, first determining the existence of edges before inferring orientations (Spirtes and Glymour 1991; Meek 1995). In the score-based approach, heuristics are designed to optimize some scoring criterion that evaluates the goodness-of-fit of a proposed structure to the available data (Heckerman et al. 1995; Chickering 2002b; Russell and Norvig 2009). Finally, hybrid methods combine the two strategies, optimizing a score over a reduced space of structures restricted through a constraint-based approach (Tsamardinos et al. 2006a; Gasse et al. 2014).

The PC algorithm, named after its authors, is often considered state-of-the-art amongst constraint-based methods for Bayesian network structure learning because of its polynomial complexity for sparse graphs and attractive theoretical properties (Spirtes and Glymour 1991; Kalisch and Bühlmann 2007). Even with its favorable scaling, PC can quickly become unwieldy for large networks, motivating various developments to structure learning speed. Several works have contributed to accelerating its execution with various parallelization strategies, resulting in speed-ups ranging from up to ten times to over three orders of magnitude (Kalisch et al. 2012; Le et al. 2016; Madsen et al. 2017; Scutari 2017; Zarebavani et al. 2020). However, these improvements are primarily feats of distributed processing implementation and are limited by the availability of required hardware. Gu and Zhou (2020) proposed a hybrid framework for partitioned estimation of Bayesian networks called partition, estimation, and fusion (PEF) in the interest of distributing learning by adopting a divide-and-conquer strategy. Unfortunately, its application to the PC algorithm does not in general retain the completeness of the PC algorithm and is limited in its capacity for parallel processing. Finally, none of these contributions tackle the practical problem that the performance of constraint-based algorithms can vary substantially with certain tuning parameters, potentially requiring multiple algorithm executions.

Prominent hybrid methods leverage the efficiency of constraint-based strategies to considerably reduce the space of Bayesian network models but sacrifice the asymptotic guarantees of constraint-based edge orientation for the generally superior empirical structural accuracy of restricted greedy search (Tsamardinos et al. 2006a). This is characteristic of members of what we call the generalized sparse candidate (GSC) framework, named after the sparse candidate algorithm (Friedman et al. 1999), in which a greedy search in the DAG space is executed from an empty graph (EG) restricted to a sparse set of candidate edges obtained through a constraint-based strategy. Hybrid algorithms belonging to GSC include max–min hill-climbing (MMHC) and hybrid hybrid parents and children (H2PC), which, despite being popular and widely regarded as well-performing, are well-known to be lacking in asymptotic guarantees (Tsamardinos et al. 2006a; Gasse et al. 2014). While the adaptively restricted greedy equivalence search (ARGES) stands out as a hybrid framework with established consistency (Nandy et al. 2018), our simulations suggest that ARGES can likewise empirically benefit from the developments in our work. In particular, both GSC and ARGES initialize their respective greedy searches with an EG and, to our knowledge, no principled and well-performing initialization strategy without assuming expert knowledge has been proposed.

We propose an answer to these challenges by the development of the partitioned hybrid greedy search (pHGS) algorithm, a hybrid structure learning algorithm that can be considered the composition of three independent contributions to the computational efficiency,

theoretical guarantees, and empirical performance of Bayesian network structure learning. In particular, pHGS accomplishes the following:

1. Restricts the search space with our proposed partitioned PC (pPC) algorithm that improves on the efficiency of the PC algorithm while retaining its soundness and completeness and capacity for parallel processing;
2. Mitigates the need for parameter tuning by automatically selecting the sparsity-controlling threshold of conditional independence tests with our p-value adjacency thresholding (PATH) algorithm that extends the accessibility of constraint-based consistency;
3. Initializes the restricted greedy search with our hybrid greedy initialization (HGI) algorithm that elevates the asymptotic guarantees of existing hybrid algorithms such as members of the GSC framework to that of sound and complete constraint-based methods while improving empirical performance.

After reviewing relevant preliminaries in Sect. 2, the novel components of pHGS are organized in the remainder of this paper as follows. In Sect. 3, we develop the pPC algorithm which employs a partitioned estimation strategy to reduce the number of statistical tests required for the exhaustive conditional independence investigation in PC-like CPDAG learning. We additionally detail the PATH thresholding algorithm, which efficiently generates and selects from a set of CPDAG estimates with varying sparsity from a single execution of pPC (or PC) and extends the accessibility of classical asymptotic consistency results to more flexible parameter specification. We begin Sect. 4 with a brief review of score-based structure learning before developing HGI, a greedy initialization strategy which endears constraint-based edge orientation to the empirical setting with desirable theoretical guarantees.

We empirically validate pPC, PATH, and HGI in Sect. 5, first independently and then collectively in the form of pHGS through an extensive simulation study. We show that pPC generally requires significantly fewer statistical calls as compared to PC, and that PATH effectively accomplishes the task of parameter tuning from a single algorithm execution with practically negligible additional computational expense. Compared to repeated executions of PC, the combined effect of pPC and PATH consistently achieves significant computational reductions without sacrificing (and indeed often improving on) estimation accuracy. We demonstrate the effectiveness of HGI on several instantiations of the GSC hybrid framework, and validate the holistic merits of pHGS against several popular structure learning algorithms. Though the focus of our paper is on the discrete case, we include succinct comments and results for our methods on high-dimensional Gaussian data in the discussion in Sect. 6 and the Supplementary Information.

2 Background

A *graph* $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a structure composed of a set of nodes $\mathbf{V} = \{1, \dots, p\}$ and a set of edges $\mathbf{E}$. For a pair of distinct nodes $i, j \in \mathbf{V}$, we encode an undirected edge between i and j in $\mathcal{G}$ by an unordered connected pair $i - j \in \mathbf{E}$, and a directed edge from i to j in $\mathcal{G}$ by an ordered pair $i \rightarrow j \in \mathbf{E}$. A *directed acyclic graph* (DAG) has only directed edges and is oriented such that there are no directed cycles in $\mathcal{G}$. A DAG $\mathcal{G}$ defines the structure of a *Bayesian network* of a joint probability distribution P of variables $\mathbf{X}$ corresponding to $\mathbf{V}$ if P factorizes according to the structure of $\mathcal{G}$:

$$P(\mathbf{X}) = \prod_{i=1}^{p} P(X_i \mid \mathbf{\Pi}_i^{\mathcal{G}}), \tag{1}$$

where $\mathbf{\Pi}_i^{\mathcal{G}} = \{X_j : j \to i \in \mathbf{E}\}$ denotes the parents of X_i according to $\mathcal{G}$. In this paper, we may refer to a node $i \in \mathbf{V}$ and its corresponding variable $X_i \in \mathbf{X}$ interchangeably. In a causal DAG, $i \to j$ asserts that i is a direct cause of j, whereas more generally, a DAG encodes in its structure a set of conditional independence statements between distinct variables according to the above factorization. For ease of notation, we let $\mathbf{X_k} = \{X_k \in \mathbf{X} : k \in \mathbf{k}\}$ for $\mathbf{k} \subseteq \mathbf{V}$.

This paper focuses on the setting in which P is a discrete probability distribution, although the presented strategies are not limited to such a domain. Each variable X_i probabilistically attains one of $r_i \geq 2$ states depending on the attained states of its parents $\mathbf{\Pi}_i^{\mathcal{G}}$. The conditional probability distributions of the variables given each of their parent configurations are multinomial distributions.

Let $(X_i \perp X_j | \mathbf{X_k})_P$ denote that X_i and X_j are independent given conditioning set $\mathbf{X_k} \subseteq \mathbf{X} \setminus \{X_i, X_j\}$ in P, and $(X_i \perp X_j | \mathbf{X_k})_{\mathcal{G}}$ that X_i and X_j are d-separated by $\mathbf{X_k}$ in $\mathcal{G}$. The factorization in Eq. (1) implies that $\mathcal{G}$ and P satisfy the *(global) Markov condition*: for disjoint sets of variables $\mathbf{A}, \mathbf{B}, \mathbf{C} \subseteq \mathbf{X}$,

$$(\mathbf{A} \perp \mathbf{B} | \mathbf{C})_{\mathcal{G}} \Rightarrow (\mathbf{A} \perp \mathbf{B} | \mathbf{C})_P. \tag{2}$$

2.1 Markov equivalence

Multiple DAGs may encode the same set of d-separation statements and thus redundantly entail the same conditional independence statements. Such DAGs are said to be *Markov equivalent*. Formally, two DAGs $\mathcal{G}$ and $\mathcal{G}'$ are Markov equivalent if $(\mathbf{A} \perp \mathbf{B} | \mathbf{C})_{\mathcal{G}} \Leftrightarrow (\mathbf{A} \perp \mathbf{B} | \mathbf{C})_{\mathcal{G}'}$ for all mutually disjoint subsets $\mathbf{A}, \mathbf{B}, \mathbf{C} \subseteq \mathbf{X}$. We refer to Markov equivalent DAGs as simply *equivalent* and belonging to the same *equivalence class*. Given our distributional assumptions on P, equivalent DAGs are indistinguishable without background information or experimental data. As our interest lies in structure learning from observational data, the objective amounts to recovering the equivalence class of the underlying DAG.

The *skeleton* of a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is the undirected graph obtained from replacing every adjacent (that is, connected) node pair in $\mathcal{G}$ with an undirected edge. A *v-structure* is a triplet $i, j, k \in \mathbf{V}$ oriented $i \to k \leftarrow j$ in $\mathcal{G}$ with i and j not adjacent. Let the *pattern* of $\mathcal{G}$ be the *partially DAG* (PDAG) obtained by orienting all and only the v-structures of $\mathcal{G}$ in its skeleton, leaving all remaining edges undirected. The following theorem was adapted from Verma and Pearl (1991) to characterize equivalent DAGs.

Theorem 1 (Meek (1995)) *Two DAGs are equivalent if and only if they have the same patterns.*

Implied by Theorem 1 is the existence of *compelled* and *reversible* edges. An edge $i \to j$ in a DAG $\mathcal{G}$ is compelled if it exists oriented as stated in every DAG in the equivalence class of $\mathcal{G}$, whereas it is reversible if it is directed $j \to i$ in at least one DAG in the equivalence class of $\mathcal{G}$. Meek (1995) detailed a set of sound and complete rules known as *Meek's rules* (R1, R2, R3, and R4) that deterministically extend the pattern of a graph $\mathcal{G}$ to its *completed*

PDAG (CPDAG), a PDAG featuring a directed edge for every compelled edge and an undirected edge for every reversible edge (Chickering 2002a). As the unique representation of its equivalence class, the CPDAG is the structure of interest for structure learning methods in the observational setting.

2.2 Faithfulness

The global Markov property, as stated in Eq. (2), defines an avenue for inference regarding the conditional independence relationships in P according to information encoded in its Bayesian network structure $\mathcal{G}$. As our interest is in recovering $\mathcal{G}$ from data generated from and thus sample estimates of probability distribution P, we require the assumption of faithfulness to infer the structure of $\mathcal{G}$ from P.

Definition 1 (Faithfulness) A distribution P and a DAG $\mathcal{G}$ are said to be *faithful* to each other if all and only the conditional independence relations true in P are entailed by the d-separation statements in $\mathcal{G}$, i.e.

$$(\mathbf{A} \perp \mathbf{B} \,|\, \mathbf{C})_{\mathcal{G}} \Leftrightarrow (\mathbf{A} \perp \mathbf{B} \,|\, \mathbf{C})_{P}.$$

Under faithfulness, we may say in such a case that $\mathbf{A}$ and $\mathbf{B}$ are separated by $\mathbf{C}$, regardless of whether we are referring to d-separation or conditional independence. If P is faithful to $\mathcal{G}$, then the existence of an edge between any distinct pair of nodes i and j can be necessarily and sufficiently determined by the nonexistence of a separation set of variables that render i and j conditionally independent in P. In particular,

$$i, j \in \mathbf{V} \text{ are not connected in } \mathcal{G} \Leftrightarrow \exists \mathbf{k} \subseteq \mathbf{V} \setminus \{i,j\} \text{ such that } (X_i \perp X_j \,|\, \mathbf{X_k})_P. \quad (3)$$

Throughout the development of our methodology, in what we call the population versions of procedures, we assume possession of all conditional independence information in P denoted $\{\perp_P\}$, thus having conditional independence oracles perfectly corresponding to d-separation. For inferring conditional independence from finite samples of discrete data $\mathcal{D}$ in the sample counterparts, we use the popular G^2 log-likelihood ratio test of independence for empirical estimation of conditional independence in P with some significance level threshold α, denoting the G^2 test statistic for testing $(X_i \perp X_j \,|\, \mathbf{X_k})_P$ as $G^2_{ij|\mathbf{k}}$ (Spirtes et al. 2000). We briefly discuss the basic notation for and evaluation of the G^2 test in Appendix C, referring details and examples to (Neapolitan 2004, Sect. 10.3.1)

2.3 The PC algorithm

The well-known PC algorithm (Spirtes and Glymour 1991), named after its authors, is widely considered the gold standard constraint-based structure learning method. The PC algorithm first efficiently estimates a skeleton, reducing the criterion stated in Eq. (3) by leveraging sparsity. Let $\mathbf{N}_i^{\mathcal{G}} = \{X_j \in \mathbf{X} : i \text{ and } j \text{ are connected in } \mathcal{G}\}$ be the *neighbors*, or adjacencies, of node i in a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$. If $\mathcal{G}$ is a DAG, the following is evident from the Markov condition:

$$i, j \in \mathbf{V} \text{ are not connected in } \mathcal{G} \Leftrightarrow \exists \mathbf{X_k} \subseteq \mathbf{N}_i^{\mathcal{G}} \setminus \{X_j\} \text{ or } \exists \mathbf{X_k} \subseteq \mathbf{N}_j^{\mathcal{G}} \setminus \{X_i\}$$
$$\text{such that } (X_i \perp X_j \,|\, \mathbf{X_k})_P. \quad (4)$$

Algorithm 1 `PC-skeleton({⊥⊥_P})` (PC-stable; population version)

---

input: conditional independence information $\{\perp\!\!\!\perp_P\}$
output: undirected graph $\mathcal{G}$
1: form the complete undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ over nodes $\mathbf{V} = \{k : X_k \in \mathbf{X}\}$
2: initialize $\mathbf{S} = \emptyset$
3: initialize $l = 0$
4: **repeat**
5:      set $\mathcal{G}' = \mathcal{G}$ to fix adjacencies
6:      **for all** unordered node pairs $i - j$ adjacent in $\mathcal{G}'$ with $|\mathbf{N}_i^{\mathcal{G}'}|-1$ or $|\mathbf{N}_j^{\mathcal{G}'}|-1 \geq l$
    do
7:          **for all** unique subsets $\mathbf{X_k} \subseteq \mathbf{N}_i^{\mathcal{G}'} \setminus \{X_j\}$ and $\mathbf{X_k} \subseteq \mathbf{N}_j^{\mathcal{G}'} \setminus \{X_i\}$ of size
    $|\mathbf{k}| = l$ **do**
8:              **if** $(X_i \perp\!\!\!\perp X_j \,|\, \mathbf{X_k})_P$ **then**
9:                  store separation set $\mathbf{S}(i,j) = \mathbf{S}(j,i) = \mathbf{k}$
10:                 disconnect i and j in $\mathcal{G}$ and continue to the next pair of nodes
11:             **end if**
12:         **end for**
13:     **end for**
14:     $l = l + 1$
15: **until** there is no node i with $|\mathbf{N}_i^{\mathcal{G}}| - 1 \geq l$, or $l > m$ for some user-specified m

---

For easy reference in our algorithm description, we detail an implementation of the skeleton estimation step of the PC algorithm known as PC-stable in Algorithm 1 (Colombo and Maathuis 2014). Note that as discussed, in the population versions of procedures we assume possession of conditional independence oracles. For finite-sample execution, we replace $\{\perp\!\!\!\perp_P\}$ with data samples $\mathcal{D}$ from which conditional independence relationships are inferred using a consistent test and some threshold α. The key difference that distinguishes the PC-stable implementation from the original PC algorithm is that in line 5, the adjacencies are fixed in $\mathcal{G}'$ such that the considerations of adjacent node pairs within the outermost loop (lines 4–15) become order-independent and thus executable in parallel. We further discuss parallel execution of the PC algorithm in Sect. 3.1.2. Note that for every node i, $\mathbf{N}_i^{\mathcal{G}} \subseteq \mathbf{N}_i^{\mathcal{G}'}$ for $\mathbf{N}_i^{\mathcal{G}'}$ in any stage in Algorithm 1, preserving the general design of the original PC skeleton learning method by ensuring the exhaustive investigation of Eq. (4) and thus retaining its theoretical properties. Hereafter, when we discuss the PC algorithm, we refer to the PC-stable implementation.

After determining the skeleton of a DAG $\mathcal{G}$, knowledge about the conditional independence relationships between variables (namely, the accrued separation sets $\mathbf{S}$) can be used to detect the existence of v-structures and orient the skeleton to the pattern of $\mathcal{G}$. Recovery of the CPDAG of $\mathcal{G}$ can then be achieved by repeated application of Meek's rules (Meek 1995). This process, which we refer to as `skel-to-cpdag` (Algorithm 6 in Appendix B), is guaranteed to orient the skeleton of a DAG $\mathcal{G}$ to its CPDAG given accurate conditional independence information entailed by $\mathcal{G}$. For details regarding constraint-based edge orientation, see Appendix B.

The complete PC(-stable) algorithm consists of skeleton estimation according to Algorithm 1 followed by edge orientation according to Algorithm 6, and is well-known to be sound and complete for CPDAG estimation (Kalisch and Bühlmann 2007; Colombo and Maathuis 2014).

3　The pPC and PATH algorithms

In constraint-based methods, the computational expense of edge orientation has been noted to be generally insignificant compared to that of skeleton estimation (Chickering 2002a; Madsen et al. 2017). As such, we develop the pPC algorithm to reduce the computational expense of skeleton estimation by imposing a partitioned ordering to the conditional independence tests. Similarly, we propose the PATH algorithm that effectively accomplishes the task of parameter tuning by efficiently generating a solution path of estimates from a single execution of pPC or PC.

3.1　The partitioned PC algorithm

The pPC algorithm improves on the already desirable efficiency of the PC algorithm while retaining its attractive theoretical properties and empirical structure learning accuracy. The structure follows similarly to the PEF strategy applied to the PC algorithm in Gu and Zhou (2020). We develop improvements and computational exploits to further increase performance, formulate pPC to retain soundness and completeness, and propose adaptations to address the challenges of learning the structure of discrete Bayesian networks.

The intuition motivating a partitioned strategy is that any structure learning algorithm that scales worse than linearly with p will be able to estimate $\kappa > 1$ subgraphs for node clusters that partition the p nodes faster than a single graph on all nodes. If the p nodes can be reliably partitioned such that the connectivity between clusters is weak relative to within clusters, then we can expect that there will not be many false positive edges (as a result of causal insufficiency) within subgraphs. Consequently, the adjacencies are expected to be relatively well-estimated, providing a selective candidate set of neighbors to screen the edges between subgraphs. Coupled with the assumed weak connectivity between clusters, the process of determining the existence of edges amongst clusters is expected to be efficient.

The pPC algorithm estimation process proceeds as follows. We partition the p nodes into κ clusters using a version of the modified hierarchical clustering algorithm proposed in Gu and Zhou (2020) applied with a normalized discrete distance metric, additionally blacklisting marginally independent node pairs. We then apply the PC algorithm to estimate edges within clusters, and filter and refine edges between nodes in different clusters. Finally, we achieve completeness by applying a reduced PC algorithm before orienting the edges.

3.1.1　Clustering

As previously motivated, the task of obtaining an effective partition of the nodes is crucial for the success of the skeleton learning. A partition with many clusters κ is desirable for greatest computational benefit in subgraph estimation, but each cluster must be substantive so as to minimally violate causal sufficiency. To accomplish this, the distances between nodes are measured by a normalized mutual information distance metric, and the target number of clusters and initial clusters are chosen adaptively.

Mutual information, denoted $I(X_i, X_j)$, serves as a similarity measure between discrete random variables X_i and X_j and may be interpreted as the Kullback–Leibler divergence between the joint probability distribution and the product of their marginals. We obtain a distance measure by inverting the pairwise mutual information after normalizing using the joint entropy $H(X_i, X_j)$. In particular, we formulate the distance between each pair of variables X_i and X_j as

$$d_{ij} = 1 - \frac{I(X_i, X_j)}{H(X_i, X_j)} \in [0, 1]. \tag{5}$$

The proposed distance d_{ij} is a metric in the strict sense as shown by Kraskov et al. (2005), meaning it is symmetric, non-negative, bounded, and satisfies the triangle inequality. In practice, we compute the empirical quantities of the mutual information and joint entropy ($\hat{I}$ and $\hat{H}$, respectively) between discrete variables (see Appendix C).

Given our distance matrix $D = (d_{ij})_{p \times p}$, we apply Algorithm 1 in Gu and Zhou (2020) with average linkage to determine a cut l for the agglomerative hierarchical clustering of the p nodes. Succinctly described, we choose the highest cut such that the resulting cluster consists of the greatest number of large clusters, defined as node clusters of at least size $0.05p$ according to a loose suggestion by Hartigan (1981). We then merge clusters of size less than $0.05p$ with other small clusters or into large clusters sequentially, ordered by average linkage, until every cluster is a large cluster. For further details regarding the algorithm, we refer to the original paper. The clustering step partitions the p nodes into κ clusters, returning the cluster labels $\mathbf{c} = \{c_1, \dots, c_p\}$, with $c_i \in \{1, \dots, \kappa\}$ denoting the cluster label of node i.

While the pairwise computation of both the mutual information and the joint entropy may seem expensive for the purpose of obtaining a partition, we take advantage of two exploits to accomplish this economically. Observing that $I(X_i, X_j) = H(X_i) + H(X_j) - H(X_i, X_j)$, we need only compute the marginal entropies $H(X_i) = I(X_i, X_i)$ to derive the joint entropies from the pairwise mutual information, a reduction from $p(p-1)$ computations to $p(p+1)/2$. Further noting that the discrete unconditional G^2 test statistic for investigating the marginal independence between X_i and X_j is computed as $G_{ij}^2 = 2n \cdot \hat{I}(X_i, X_j)$ [see Eq. (23) in Appendix C], an initial edge screening can easily be obtained through the evaluation

$$\Pr(\chi_f^2 > 2n \cdot \hat{I}(X_i, X_j)) > \alpha \; \Rightarrow \; (X_i \perp X_j)_P \\ \Rightarrow \text{ blacklist the edge } i - j, \tag{6}$$

where f is the degrees of freedom corresponding to the test of independence between i and j conditioned on $\mathbf{k} = \emptyset$ (see Appendix C). This effectively accomplishes the empty conditioning set ($l = 0$) testing step of the PC algorithm by separating all marginally independent pairs of variables (Algorithm 2 line 2).

Note that for continuous data, we recommend using correlation as a similarity measure, which Gu and Zhou (2020) found to lead to reasonable partitions of nodes for divide-and-conquer strategies for learning Gaussian Bayesian networks. This design may also take advantage of Eq. (6) by testing for zero correlation. After the clustering step, the pPC algorithm and other methods developed in this paper can be generalized to continuous cases with straightforward substitutions of conditional independence tests and score functions.

3.1.2 Partitioned skeleton estimation

We now apply the PC algorithm skeleton learning phase to estimate κ disconnected undirected subgraphs according to the partition obtained in the clustering step. Practically, independently applying the PC algorithm to each node cluster benefits from at most κ processors if distributed as such for parallel processing. Furthermore, the speed-up is limited

by the longest estimation runtime, usually corresponding to the largest node cluster. In contrast, the design of the PC-stable implementation (Algorithm 1) by Colombo and Maathuis (2014) allows for parallel investigation of adjacent node pairs in lines 6–13, provided that updating the graph estimate is deferred to a synchronization step between iterations of l. Several contributions and implementations exist for this approach, referred to as vertical parallelization, which addresses the case where the number of variables p is large (Kalisch et al. 2012; Le et al. 2016; Scutari 2017; Zarebavani et al. 2020). Alternatively, a horizontal parallelization approach parallelizes across data observations and is preferred when the sample size n is large (Madsen et al. 2017). In these parallelization paradigms, due to the large number of distributed tasks, the number of utilizable computing processors is not practically limited, and the computational load is reasonably expected to be evenly distributed. To take advantage of these developments in parallelizing the PC algorithm, we estimate subgraphs within the node clusters by executing Algorithm 1 with the following modifications: (i) form the initial complete undirected graph by only connecting nodes within clusters, (ii) delete edges according to Eq. (6), and (iii) begin investigating candidate conditioning sets of size $l = 1$. The result is an undirected graph $\mathcal{G}$ on $\mathbf{V}$ consisting of κ disconnected subgraphs.

At this stage, given a good partition, we expect the node adjacencies to be relatively well-estimated, with the exception of extraneous edge connections within clusters due to the violation of causal sufficiency and missing edge connections between clusters that are disconnected by the partition. Recall from Sect. 3.1.1 that many pairs, including those between clusters, were removed from consideration via the initial marginal independence filtering according to Eq. (6). We further refine the pairs between clusters through a two-step screening process, our strategy being similar to Algorithm 2 designed by Gu and Zhou (2020), with modifications made to accommodate discrete structure learning. Note that this is where we anticipate to derive the most computational advantage over the PC algorithm. Assuming a block structure was successfully detected, we aim to circumvent the lower order conditional independence tests in our investigation of Eq. (4) by separating as many between cluster pairs as possible with the currently estimated adjacencies.

The proposed between cluster screening process is summarized in the following operations to the currently estimated skeleton $\mathcal{G} = (\mathbf{V}, \mathbf{E})$:

$$\mathbf{E} \leftarrow \mathbf{E} \cup \left\{ i - j : c_i \neq c_j, (X_i \not\perp X_j)_P, \text{ and } (X_i \not\perp X_j | \mathbf{N}_i^{\mathcal{G}} \cup \mathbf{N}_j^{\mathcal{G}})_P \right\}, \tag{7}$$

$$\mathbf{E} \leftarrow \mathbf{E} \setminus \left\{ i - j \in \mathbf{E} : c_i \neq c_j, \text{ and } (X_i \perp X_j | \mathbf{N}_i^{\mathcal{G}} \setminus \{X_j\})_P \right.$$
$$\left. \text{or } (X_i \perp X_j | \mathbf{N}_j^{\mathcal{G}} \setminus \{X_i\})_P \right\}, \tag{8}$$

where c_i is the cluster label of X_i. The first screen in Eq. (7) constructively connects the between cluster edges that are dependent marginally, as assessed according to Eq. (6), as well as conditioned on the union of the neighbor sets. With the addition of edges between clusters, the second screen in Eq. (8) disconnects pairs that are separated by the newly updated adjacencies. As in Algorithm 1, we fix adjacencies to retain the capacity for parallel investigation of the considered node pairs. Note that every between-cluster edge that is present in the underlying DAG will be connected by Eq. (7), and Eq. (8) can only prune false positives. Thus, after this step every node pair will have been considered and, in the population setting, every truly connected edge in the underlying DAG will be connected in $\mathcal{G}$.

Algorithm 2 pPC($\{\perp\!\!\!\perp_P\}$) (population version)

---

input: conditional independence information $\{\perp\!\!\!\perp_P\}$
output: CPDAG estimate $\mathcal{G}$
    partitioning
1: assign cluster labels **c** by partitioning all nodes into κ clusters as in Sect. 3.1.1
    within cluster edge estimation
2: execute $\mathcal{G} \leftarrow$ PC-skeleton($\{\perp\!\!\!\perp_P\}$) (Algorithm 1), with the following modifications:
        line 1: form undirected graph by only connecting node pairs $i, j \in \mathbf{V}$ with $c_i = c_j$
        line 3: delete marginally independent pairs from $\mathcal{G}$ according to Eq. (6) and initialize $l = 1$
    between cluster edge screening
3: initialize $\mathcal{G}' = \mathcal{G}$ to fix adjacencies
4: **for all** unordered node pairs i, j such that $c_i \neq c_j$ and $(X_i \not\!\perp\!\!\!\perp X_j)_P$ **do**
5:     **if** $(X_i \not\!\perp\!\!\!\perp X_j | \mathbf{N}_i^{\mathcal{G}'} \cup \mathbf{N}_j^{\mathcal{G}'})_P$ **then**
6:         connect i and j in $\mathcal{G}$
7:     **else**
8:         store separation set in $\mathbf{S}(i,j) = \mathbf{S}(j,i)$
9:     **end if**
10: **end for**
11: set $\mathcal{G}' = \mathcal{G}$ to update fixed adjacencies
12: **for all** unordered node pairs i, j adjacent in $\mathcal{G}$ with $c_i \neq c_j$ **do**
13:     **if** $(X_i \perp\!\!\!\perp X_j | \mathbf{N}_i^{\mathcal{G}'} \setminus \{X_j\})_P$ or $(X_i \perp\!\!\!\perp X_j | \mathbf{N}_j^{\mathcal{G}'} \setminus \{X_i\})_P$ **then**
14:         store separation set in $\mathbf{S}(i,j) = \mathbf{S}(j,i)$
15:         disconnect i and j in $\mathcal{G}$
16:     **end if**
17: **end for**
    complete edge estimation
18: execute $\mathcal{G} \leftarrow$ PC-skeleton($\{\perp\!\!\!\perp_P\}$) (Algorithm 1), continuing from $\mathcal{G}$ and restricting to Eq. (9) with the following replacements:
        line 1: initialize undirected graph with current estimate $\mathcal{G}$
        line 2: omit this line
        line 3: initialize $l = 1$
        line 7: **for all** unique subsets $\mathbf{X_k} \subseteq \mathbf{N}_i^{\mathcal{G}'} \setminus \{X_j\}$ and $\mathbf{X_k} \subseteq \mathbf{N}_j^{\mathcal{G}'} \setminus \{X_i\}$ satisfying $|\mathbf{k}| = l$ and criteria Eq. (9) **do**
    edge orientation
19: execute $\mathcal{G} \leftarrow$ skel-to-cpdag($\mathcal{G}, \mathbf{S}$) (Algorithm 6) to orient $\mathcal{G}$

Remark 1 The formulation of the discrete G^2 tests of independence require enumerating and counting across all conditioning variable configurations. An unavoidable consequence is that the computational complexity and memory requirement increase dramatically with increasing conditioning variables, especially when they have a large number of discrete levels. As such, it is of practical interest to restrict the size of conditioning sets to some user-specified $m > 0$. Thus, for the evaluations in Eqs. (7) and (8) (Algorithm 2 lines 5 and 13), if for example $|\mathbf{N}_i^{\mathcal{G}} \cup \mathbf{N}_j^{\mathcal{G}}| > m$, we instead investigate the unique sets $\mathbf{X_k} \subseteq \mathbf{N}_i^{\mathcal{G}} \cup \mathbf{N}_j^{\mathcal{G}}$ such that $|\mathbf{k}| = m$. Furthermore, in the case that the considered neighborhood is large due
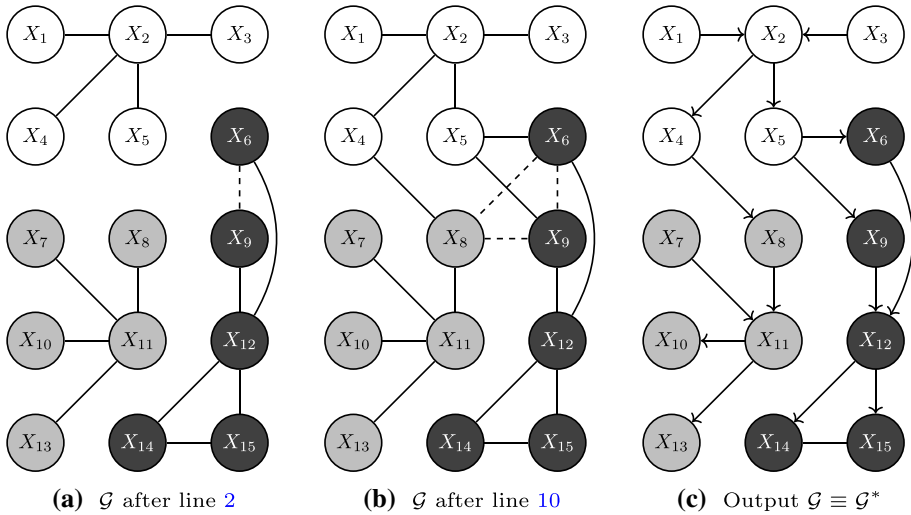
Fig. 1 Example of various stages of pPC (Algorithm 2). Node shading denotes clusters, lines represent estimated edges, with dashed lines representing false positive edges. The estimated graph after line 2 and after line 10 are shown in **a** and **b**, respectively. The final output is equivalent to the CPDAG of the underlying DAG $\mathcal{G}^*$ in **c**

to an unfavorable partition, the investigated conditioning sets may be limited to those contained within only a subset of the neighborhood, provided the subsequent step is adjusted accordingly [(b) in Eq. (9)].

However, it is important to note that the constraint-based criterion for edge existence expressed in Eq. (4) has not yet been fully investigated for all node pairs. The tests for edges within clusters only considered conditioning sets consisting of nodes within the same clusters, and tests for edges between clusters were limited to the empty set, $\mathbf{N}_i^{\mathcal{G}} \cup \mathbf{N}_j^{\mathcal{G}}$, $\mathbf{N}_i^{\mathcal{G}} \setminus \{X_j\}$, and $\mathbf{N}_j^{\mathcal{G}} \setminus \{X_i\}$. In particular, for each remaining adjacent node pair $i - j \in \mathbf{E}$, the possible separation sets that have not been evaluated are limited to either of the following cases, if any:

$$
\begin{aligned}
&\text{(a) } c_i = c_j \text{ (within clusters), sets } \mathbf{k} \text{ where } \exists k \in \mathbf{k} \text{ such that } c_k \neq c_i; \\
&\text{(b) } c_i \neq c_j \text{ (between clusters), sets } \mathbf{k} \neq \emptyset \text{ not defined in Eq. (7) or Eq. (8).}
\end{aligned}
\tag{9}
$$

The most straightforward continuation to achieve completeness in our partitioned skeleton learning process would be to exhaustively evaluate the dependence of the remaining connected node pairs in $\mathcal{G}$ conditioned on the remaining conditioning sets. We accomplish this by restarting the PC algorithm on the current skeleton, evaluating independence conditioned on sets restricted to criteria Eq. (9), before finally orienting the resulting skeleton to a CPDAG with `skel-to-cpdag` (Algorithm 6) to complete structure learning.

The resulting pPC algorithm is detailed in Algorithm 2, and an example of its execution is illustrated in Fig. 1, which provides intuition for its theoretical properties expressed in Theorem 2. The pPC algorithm first estimates the edges within the clusters to obtain $\kappa = 3$ disconnected subgraphs (Fig. 1a). False positives may be present in these subgraphs in the case that conditioning sets that separate truly disconnected variables contain variables

belonging to different clusters, such as in the case of $X_6 - X_9$ for which the common parent X_5 is in a different cluster. Similarly, two false positive edges are constructively connected along with all true positives according to Eq. (7) (Fig. 1b), but these are quickly pruned by the second screening of edges between clusters with Eq. (8) (not shown). $X_6 - X_9$ remains to be separated, requiring $(X_6 \perp X_9 | X_5)_P$ to be investigated in line 18 to recover the underlying skeleton before orientation to the true CPDAG $\mathcal{G}^*$ in line 19 (Fig. 1c).

The pPC algorithm (Algorithm 2) is sound and complete, formalized in Theorem 2, which we prove in Appendix A.

Theorem 2 *Suppose that probability distribution P and DAG $\mathcal{G}^*$ are faithful to each other. Then given conditional independence oracles, the output of the pPC algorithm (Algorithm 2) is the CPDAG that represents the equivalence class of $\mathcal{G}^*$.*

Its implication is the asymptotic consistency of pPC for fixed p as $n \to \infty$, given a consistent conditional independence test such as the G^2 test (Cressie and Read 1989). Note that while the computational savings may depend heavily on the quality of the partitioning, Theorem 2 holds regardless of the obtained clusters.

3.2 *p*-value adjacency thresholding

Despite the attractive theoretical properties of algorithms such as pPC and PC, it is well-known that in practice, constraint-based algorithms suffer from the multiple testing problem, a challenge exacerbated when p is large (Spirtes 2010). The effect of individual errors can compound throughout the conditional independence testing process, leading to erroneous inferences regarding both the existence and orientation of edges (Koller and Friedman 2009; Spirtes 2010). In addition to deteriorated quality of structures estimated, mistakes in conditional independence inferences can result in invalid PDAG estimates that do not admit a consistent extension (see Remark 2). In practical applications, the choice of conditional independence test threshold α can significantly control the sparsity and quality of the resulting estimate. To the best of our knowledge, proposed theoretical thresholds depend on unknown quantities and are not practically informative, such as in Kalisch and Bühlmann (2007). Empirically, the optimal choice of α varies depending on factors such as the sample size and the structure and parameters of the underlying Bayesian network, and no universally well-performing value is known.

We propose the PATH algorithm to generate and select from a CPDAG solution path across various values of α from a single execution of the pPC algorithm, or indeed from any constraint-based structure learning algorithm that is able to obtain the following. Define the maximum p-values $\Phi = (\Phi_{ij})$ such that Φ_{ij} is the maximum p-value obtained by the conditional independence test between i and j across all conditioning sets $\mathbf{k} \in \mathcal{K}_{ij}$ visited in the algorithm, as well as the corresponding separation sets, thus extending the definition of $\mathbf{S}$. That is, for all distinct node pairs i, j,

$$
\begin{aligned}
\Phi_{ij} = \Phi_{ji} &:= \max_{\mathbf{k} \in \mathcal{K}_{ij}} \Pr\left(\chi_f^2 > G_{ij|\mathbf{k}}^2 \right), \\
\mathbf{S}(i,j) = \mathbf{S}(j,i) &:= \operatorname{argmax}_{\mathbf{k} \in \mathcal{K}_{ij}} \Pr\left(\chi_f^2 > G_{ij|\mathbf{k}}^2 \right).
\end{aligned}
\tag{10}
$$

For a connected node pair $i - j$, $\mathbf{S}(i,j)$ may be considered the conditioning set closest to separating i and j, and Φ_{ij} measures how close.

The process itself is straightforward: for a sequence of significance levels $\{\alpha^{(t)}\}$, we obtain updated PDAG estimates $\mathcal{G}^{(t)}$ by thresholding the maximum achieved p-values Φ to obtain skeleton estimates with edge sets $\mathbf{E}^{(t)} = \{i - j : \Phi_{ij} \leq \alpha^{(t)}\}$ and then orienting them to CPDAGs according skel-to-cpdag (Algorithm 6) with the corresponding separation information $\mathbf{S}^{(t)} = \{\mathbf{S}(i, j) : \Phi_{ij} > \alpha^{(t)}\}$. The quality of the estimates are then evaluated by a scoring criterion and the highest-scoring network is returned. In what follows, we develop the choice of the threshold values and present the strategy for estimate generation and selection.

We begin with a graph estimate obtained by executing the pPC algorithm with some maximal threshold value α. The goal is to start with the densest graph so that the elements of Φ and $\mathbf{S}$ in Eq. (10) are maximized over a larger number of visited conditioning sets $|\mathcal{K}|$. We generate a sequence of τ values decreasing from $\alpha^{(1)} := \alpha$ to some minimum threshold value $\alpha^{(\tau)}$. This sequence may be incremented according to some linear or log-linear scale, but we choose to achieve maximal difference in sparsity amongst estimates by utilizing the information in Φ. Given each $\alpha^{(t)}$ corresponding to estimate $\mathcal{G}^{(t)} = (\mathbf{V}, \mathbf{E}^{(t)})$, we choose $\alpha^{(t+1)}$ such that

$$|\mathbf{E}^{(t)}| - |\mathbf{E}^{(t+1)}| \approx \frac{|\mathbf{E}^{(1)}| - |\mathbf{E}^{(\tau)}|}{\tau - 1}. \tag{11}$$

Noting that $|\mathbf{E}^{(t)}| = \sum_{i<j} \mathbb{1}\{\Phi_{ij} \leq \alpha^{(t)}\}$ for indicator function $\mathbb{1}\{\cdot\}$, it is easy to see that the sequence $\alpha^{(1)}, \ldots, \alpha^{(\tau)}$ can be straightforwardly obtained using the order statistics of the elements of Φ.

Once a solution path of CPDAG estimates $\{\mathcal{G}^{(t)} : t \in \{1, \ldots, \tau\}\}$ is obtained, we select the highest quality estimate by means of score-based selection. The *Bayesian information criterion* (BIC) is a penalized log-likelihood score derived from the asymptotic behavior of Bayesian network models, with established consistency (Schwarz 1978). The formulation of the BIC score makes clear its *score decomposability* [see Eq. (24) in Appendix C]; that is, the score $\phi(\mathcal{G}, \mathcal{D})$ can be computed as the sum of the scores of the individual variables with respect to their parents in $\mathcal{G}$: $\phi(\mathcal{G}, \mathcal{D}) = \sum_{i=1}^{p} \phi(X_i, \mathbf{\Pi}_i^{\mathcal{G}})$. The BIC score is additionally *score equivalent*, evaluating all Markov equivalent DAGs as of identical quality, with a higher value indicating a better fit to the data.

Due to score equivalence, it is sufficient to evaluate each CPDAG $\mathcal{G}^{(t)}$ with any arbitrary DAG $\tilde{\mathcal{G}}^{(t)}$ in its equivalence class, called a *consistent extension* of $\mathcal{G}^{(t)}$. Dor and Tarsi (1992) proposed a simple algorithm, which we refer to as pdag-to-dag, that obtains such an extension by orienting the undirected (reversible) edges in $\mathcal{G}^{(t)}$ without inducing directed cycles or introducing new v-structures, and is guaranteed to succeed if a consistent extension exists (further discussed in Sect. 4.2). After obtaining these DAG extensions, in practice, score decomposability can be leveraged to avoid scoring p nodes for τ estimates by setting $\Delta^{(1)} = 0$ and computing the score differences between estimates $\Delta^{(t)} = \phi(\tilde{\mathcal{G}}^{(t)}, \mathcal{D}) - \phi(\tilde{\mathcal{G}}^{(t-1)}, \mathcal{D})$ for $t = 2, \ldots, \tau$, caching computed node score to avoid redundant computations. The best solution can then be straightforwardly obtained according to

$$t^* = \text{argmax}_{t \in \{1, \ldots, \tau\}} \phi(\mathcal{G}^{(t)}, \mathcal{D}) = \text{argmax}_{t \in \{1, \ldots, \tau\}} \sum_{r=1}^{t} \Delta^{(r)}.$$

We detail the PATH solution path strategy in Algorithm 3.

Algorithm 3 PATH(Φ, **S**, $\mathcal{D}$, τ, $\alpha^{(\tau)}$) (sample version)

---

input: maximum p-values Φ, separation sets **S**, data $\mathcal{D}$, number of estimates τ,
    minimum threshold $\alpha^{(\tau)}$

output: CPDAG $\mathcal{G}^{(t^*)}$, CPDAG solution path $\{\mathcal{G}^{(t)} : t \in \{1, \ldots, \tau\}\}$

1: generate a decreasing sequence of τ values $\{\alpha^{(t)}\}$ from $\alpha^{(1)} := \max_{i,j} \Phi_{ij}$ to $\alpha^{(\tau)}$
    by Eq. (11)

2: **for** $t = 1, 2, \ldots, \tau$ **do**

3:     obtain skeleton $\mathcal{G}^{(t)} = (\mathbf{V}, \mathbf{E}^{(t)})$ by thresholding: $\mathbf{E}^{(t)} = \{i - j : \Phi_{ij} \leq \alpha^{(t)}\}$

4:     obtain separation sets $\mathbf{S}^{(t)} = \{\mathbf{S}(i, j) : \Phi_{ij} > \alpha^{(t)}\}$

5:     execute $\mathcal{G}^{(t)} \leftarrow$ `skel-to-cpdag`$(\mathcal{G}^{(t)}, \mathbf{S}^{(t)})$ (Algorithm 6 in Appendix B)

6:     obtain DAG extension $\tilde{\mathcal{G}}^{(t)} \leftarrow$ `pdag-to-dag`$(\mathcal{G}^{(t)})$ (Dor & Tarsi, 1992)

7:     **if** $t = 1$ **then**

8:         $\Delta^{(t)} = 0$

9:     **else**

10:        compute score difference $\Delta^{(t)} = \phi(\tilde{\mathcal{G}}^{(t)}, \mathcal{D}) - \phi(\tilde{\mathcal{G}}^{(t-1)}, \mathcal{D})$

11:     **end if**

12: **end for**

13: select the best estimate $t^* = \text{argmax}_{t \in \{1, \ldots, \tau\}} \sum_{r=1}^{t} \Delta^{(r)}$

14: return $\mathcal{G}^{(t^*)}$ and $\{\mathcal{G}^{(t)} : t \in \{1, \ldots, \tau\}\}$

Remark 2 While `pdag-to-dag` is guaranteed to extend a PDAG $\mathcal{G}$ to a DAG if any consistent extension exists, in the presence of finite-sample error, Algorithm 6 may obtain a PDAG estimate $\mathcal{G}$ for which no such extension exists. In such a case, we say that $\mathcal{G}$ does not admit a consistent extension, and refer to it as an invalid CPDAG. Such PDAGs contain undirected edges that cannot be oriented without inducing cycles or constructing additional v-structures, and do not encode any probabilistic model of *P*. To account for these, we restrict the candidate graphs considered in line 13 to valid CPDAGs. In the case that no valid CPDAG is obtained, we obtain semi-arbitrary DAG extensions $\tilde{\mathcal{G}}^{(t)}$ by first applying the algorithm by Dor and Tarsi (1992) and randomly directing as many remaining undirected edges as possible without introducing any cycles, finally removing edges that cannot be oriented. The resulting DAGs are used to score the PDAGs, and the original PDAGs are returned in the output as these structures are nonetheless interpretable even as incomplete dependency structures.

The computational expense of executing Algorithm 3 can reasonably be expected to be insignificant compared to any constraint-based algorithm to which it is attached, supported by our empirical results in Sect. 5.2. Our exploitation of score decomposability reduces the order of score computations far below the worst case of $O(\tau p)$, which is already much more efficient than any favorable order of conditional independence tests such as polynomial with *p*. As for the τ executions of `skel-to-cpdag` and `pdag-to-dag`, Chickering (2002a) found the computational cost of applying the edge orientation heuristics to be insignificant, as did Madsen et al. (2017) for `skel-to-cpdag` in comparison to skeleton learning. As such, the computational cost for score-based selection from the solution path can be expected to be essentially inconsequential, an assertion further validated in our experiments.

In the large-sample limit, the correctness of Eq. (4) and `skel-to-cpdag` implies the asymptotic consistency of PC and pPC under certain conditions without any solution path (see Lemma 2 in Appendix A). This property can be achieved with a consistent conditional independence test by controlling type I error with $\alpha_n \to 0$ as $n \to \infty$, thus rendering the test Chernoff-consistent (Cressie and Read 1989; Shao 2003, Definition 2.13). As α_n is not practically informative due to its implicit dependence on n, Algorithm 3 contributes an element of accessibility to this asymptotic result in the form of the following theorem, a proof for which can be found in Appendix A.

Theorem 3 *Suppose the distribution P is fixed and faithful to a DAG with CPDAG $\mathcal{G}^*$, $\mathcal{D}$ is data containing n i.i.d. samples from P, and ϕ is a consistent score. Let $\Phi_n = (\Phi_{n,ij})$ and $\mathbf{S}_n$ be the maximum p-values and corresponding separation sets recorded for any exhaustive investigation of Eq. (4), executed with a consistent test applied with threshold α_n. Let $\hat{\mathcal{G}}_n^{(t^*)}(\alpha_n)$ be the selected estimate of Algorithm 3 with parameters $\tau = 1 + \sum_{i<j} \mathbb{1}\{\Phi_{n,ij} \leq \alpha_n\}$ and $\alpha^{(\tau)} = 0$ applied to Φ_n and $\mathbf{S}_n$. Then there exists $a_n \to 0$ as $n \to \infty$ such that if $\alpha_n \geq a_n$ when n is large,*

$$\lim_{n \to \infty} \Pr\left[\hat{\mathcal{G}}_n^{(t^*)}(\alpha_n) = \mathcal{G}^*\right] = 1. \tag{12}$$

In particular, Eq. (12) holds if α_n is fixed to some constant $\alpha \in (0, 1)$ for all n.

In the finite-sample setting, the thresholded estimate $\mathcal{G}^{(t)}$ corresponding to $\alpha^{(t)} < \alpha^{(1)}$ typically does not correspond exactly to the estimate obtained by directly executing PC or pPC with significance level $\alpha^{(t)}$. Adjacencies that would be disconnected in the earlier stages of the learning process with threshold $\alpha^{(t)}$ may survive to the later stages with a more lenient threshold $\alpha^{(1)}$. The enlarged neighborhoods may persist as additional (potentially false positive) connections in the output structure, but may also cause more conditioning sets to be considered and thus lead to the deletion of edges that would not have been deleted in the execution with $\alpha^{(t)}$. Thus, while the estimated structure from executing with $\alpha^{(1)}$ may reasonably be expected to be denser than that of $\alpha^{(t)}$, it may not be a supergraph of the latter, and the p-values for each node pair [Eq. (10)] are optimized over different conditioning sets, so the thresholded estimate with PATH is likely to differ. Nonetheless, our empirical results demonstrate the potential of the solution path generated by thresholding, showing that PATH applied to pPC and PC is able to produce estimates of competitive quality to the best of those obtained by multiple executions with various α.

To the best of our knowledge, there is currently no easy way to choose the optimal threshold α_n for PC (or pPC), and thus repeated executions are often needed for parameter tuning. The application of PATH pragmatically allows for a single execution of PC or pPC with fixed threshold α while returning estimates with both theoretical guarantees (Theorem 3) and empirical well-performance (Sect. 5.2).

4 Consistent hybrid structure learning

Despite the accessibility provided by PATH (Algorithm 3), the asymptotic guarantees of constraint-based learning strategies do not necessarily translate to well-performance in practice. For this reason, based on their experiments with which our simulation results are in agreement, Tsamardinos et al. (2006a) preferred greedy search in the DAG space over

constraint-based orientation in their development of their algorithm, even though the former is lacking in comparable theoretical guarantees.

In the interest of elevating the class of algorithms that share such an compromise, we develop the HGI strategy to preserve the asymptotic guarantees of sound and complete constraint-based structure learning while improving on the empirical well-performance of the current standard hybrid framework. We motivate and develop HGI in this section, first reviewing relevant standard score-based and hybrid structure learning before describing the HGI strategy in detail.

4.1 Score-based and hybrid structure learning

Chickering (2002a) distills score-based Bayesian network learning into two general problems: the evaluation problem and the identification problem. In this section, we begin by introducing the relevant tenets of score-based structure learning under these categories.

We have briefly interacted with the evaluation problem in our discussion of the BIC score in Sect. 3.2, which more broadly involves the development of scoring criteria to evaluate the goodness-of-fit of a Bayesian network to data. The existence of equivalence classes (Sect. 2.1) motivates the design of metrics that evaluate all structures within an equivalence class as of equal quality, satisfying the score equivalence property. The BIC score that we utilize satisfies this property, and is equivalent to the (negative) minimum description length (MDL) in Rissanen (1978). Other scores that are score equivalent include log-likelihood, Akaike's information criterion (AIC), and Bayesian Dirichlet equivalence score (BDeu) (Akaike 1974; Buntine 1991; Heckerman et al. 1995). Notwithstanding, prominent scores that are not score equivalent exist as well, such as the K2 score (Cooper and Herskovits 1991) and, more recently, ℓ_1-regularized likelihood scores (Fu and Zhou 2013; Gu et al. 2019). In their investigation, Liu et al. (2012) found BIC to have favorable model selection performance relative to a number of other scores.

The consistency of the BIC score guarantees that in the large-sample limit, $\mathcal{G}^* = \text{argmax}_{\mathcal{G}} \phi(\mathcal{G}, \mathcal{D})$ is in the equivalence class of the underlying DAG. BIC additionally retains the property of *local consistency* (Chickering 2002b), meaning for any DAG $\mathcal{G}$ and another DAG $\mathcal{G}'$ resulting from adding the edge $X_i \rightarrow X_j$ to $\mathcal{G}$, the following two properties hold asymptotically:

$$(X_j \not\perp X_i \mid \mathbf{\Pi}_j^{\mathcal{G}})_P \Rightarrow \phi(\mathbf{X} \mid \mathcal{G}', \mathcal{D}) > \phi(\mathbf{X} \mid \mathcal{G}, \mathcal{D}), \text{ and} \tag{13}$$

$$(X_j \perp X_i \mid \mathbf{\Pi}_j^{\mathcal{G}})_P \Rightarrow \phi(\mathbf{X} \mid \mathcal{G}', \mathcal{D}) < \phi(\mathbf{X} \mid \mathcal{G}, \mathcal{D}). \tag{14}$$

We have discussed the BIC score as having desirable qualities for evaluating Bayesian network structures, being decomposable, score equivalent, consistent, locally consistent, and empirically well-performing. However, finding the global optimum $\mathcal{G}^*$ is highly non-trivial, reminding us of the problem of identification.

Relevant to our work is the general *greedy search* algorithm which repeatedly moves from the current state to the neighboring state that maximally improves the optimization criterion (in our application, BIC) until no improvement can be thusly achieved (Russell and Norvig 2009). That is, the algorithm is guaranteed to terminate in a locally optimal state, where locality is determined by the chosen definitions of a state and its neighborhood. The popular hill-climbing (HC) algorithm is a greedy search in the state space of DAGs, with neighboring states defined as DAGs obtainable by a single directed edge

addition, deletion, or reversal applied to the current DAG (Heckerman et al. 1995; Russell and Norvig 2009). The greedy equivalence search (GES) algorithm is a sound and complete variation of greedy search in which the state space is CPDAGs representing equivalence classes, with a forward-stepping edge addition phase followed by an edge deletion phase (Meek 1997; Chickering 2002a, b).

While widely applied and regarded as efficient and well-performing, the locality of the HC search in the DAG space unavoidably risks the common problem of accepting locally optimal yet globally suboptimal solutions. Gámez et al. (2011) showed that under certain conditions, HC returns a minimal independence map of the probability distribution P, but it does not guarantee a globally optimal result. HC can be augmented to more thoroughly search the DAG space with one or both of *tabu list* and *random restarts*, respectively governed by parameters (t_0, t_1) and (r_0, r_1). In what is known as the *tabu search*, a solution is obtained through HC while a tabu list stores the last t_1 DAG structures visited during the search. Then, the HC procedure is continued for up to t_0 iterations while allowing for minimal score decreases, with a local neighborhood restricted by the tabu list to avoid previously visited structures. In HC with random restarts, the HC procedure is repeated r_0 times after the initial execution by perturbing the current solution with r_1 random local changes. In our work, we prefer augmenting HC with a tabu list rather than random restarts due to its generally superior efficiency and its reliable and deterministic well-performance.

As mentioned in Sect. 1, prominent hybrid structure learning algorithms are instantiations of what we call the GSC framework, in which HC is executed from an EG restricted to a sparse set of candidate edge connections. That is, for a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ estimated using a constraint-based approach, define $\mathbf{A} = \{(i, j) : i \text{ and } j \text{ are connected in } \mathcal{G}\}$ as the set of candidates: the unordered node pairs that have not been determined to be conditionally independent. HC is then executed from an EG on $\mathbf{V}$, considering adding an edge $i \rightarrow j$ only if $(i, j) \in \mathbf{A}$. MMPC and H2PC are two well-known examples, obtaining $\mathbf{A}$ according to sound skeleton estimation algorithms max–min parents and children (MMPC) and hybrid parents and children (HPC), respectively (Tsamardinos et al. 2006a; Gasse et al. 2014). The GSC strategy guarantees estimation of a valid DAG restricted to $\mathbf{A}$, but often accepts locally optimal solutions that are structurally inaccurate due to the connectivity of the DAG space induced by the HC neighborhood. As will be seen in Sect. 5.3, this problem persists even when the search space is well-restricted and a tabu list is utilized, leaving much to be desired. In this work, we primarily focus on improving upon the theoretical properties and empirical performance of the GSC framework, though our contributions may be applied to another hybrid approach that uses greedy search in the space of equivalence classes instead of DAGs, which we discuss briefly in Sect. 6 and the Supplementary Information.

4.2 Hybrid greedy initialization

We now develop our proposed HGI strategy to overcome the aforementioned difficulties for hybrid algorithms belonging to the GSC framework. Our method is designed to retain the soundness and completeness of constraint-based structure learning while empirically improving structural estimation accuracy and achieving higher-scoring structures as compared to those obtained by the GSC framework. The primary novel contribution is the introduction of a score-based ordering to the application of orientation heuristics to obtain a favorable initialization for HC. Given the skeleton output of a constraint-based algorithm, we sequentially add v-structures that most improve the score, scored with respect to directed edges. We then make greedy determinations for the remaining undirected edges
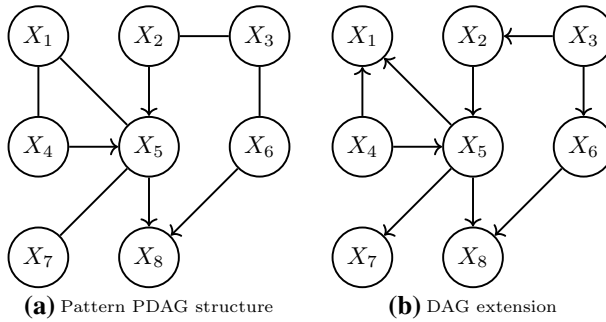
(a) Pattern PDAG structure   **(b)** DAG extension

Fig. 2 Example of `pdag-to-dag` (Dor and Tarsi 1992) applied to a PDAG pattern structure

according to efficient criteria from `pdag-to-dag`, assisted by Meek's rules R1–4 (Dor and Tarsi 1992; Meek 1995). Finally, we execute HC initialized by the resulting DAG. From a score-based learning perspective, the formulation of HGI may be understood as a principled strategy for obtaining a good starting point for greedy search. In what follows, we further detail and discuss the HGI algorithm.

Recall that `pdag-to-dag` (introduced in Sect. 3.2) is guaranteed to obtain a consistent extension of a PDAG if one exists, and thus implicitly includes Meek's rules R1–4 when the given PDAG is a valid pattern that admits a consistent extension. Let $\mathcal{G}_0$ and $\mathcal{G}$ be identical copies of a PDAG to be oriented. The algorithm repeatedly searches for a node j satisfying the following conditions in a PDAG $\mathcal{G}_0$:

(a) j is a *sink*: that is, j has no edges directed outwards in $\mathcal{G}_0$;
(b)   For every vertex k connected to j by an undirected edge in $\mathcal{G}_0$, k is adjacent to all the other vertices which are adjacent to j in $\mathcal{G}_0$.

If such a node j can be found, all undirected edges adjacent to j are oriented into j in $\mathcal{G}$ and $\mathcal{G}_0$. Node j is then a *complete sink*, a node satisfying (a) with no undirected edges incident to it, and is removed from $\mathcal{G}_0$ along with all edges incident to it in order to uncover subsequent candidate nodes. This process is repeated until $\mathcal{G}$ is fully oriented to a DAG, or until no such node j can be found, in which case the initial PDAG does not admit a consistent extension. Briefly exposited, (a) ensures acyclicity by requiring that all directed paths induced by considered orientations terminate in sinks, and (b) ensures that considered orientations do not create new v-structures if applied.

Consider an example of `pdag-to-dag` applied to a PDAG in Fig. 2. The algorithm proceeds as follows. Starting from the PDAG structure in Fig. 2a, nodes X_1, X_7, and X_8 satisfy conditions (a) and (b), with X_8 additionally a complete sink. Nodes X_2, X_4, and X_6 violate condition (a), and X_3 and X_5 violate condition (b). Since X_1, X_7, and X_8 are not adjacent to each other, they may be selected by the algorithm in an arbitrary order without affecting the particular outcome of the DAG extension, resulting in orientations $X_4 \rightarrow X_1$, $X_5 \rightarrow X_1$, and $X_5 \rightarrow X_7$. Once these nodes are removed from consideration, X_5 and X_6 are likewise removed as complete sinks, and the remaining undirected edge $X_2 - X_3$ may be oriented in either direction as both X_2 and X_3 satisfy (a) and (b).

Since the implementation of `pdag-to-dag`, as proposed, does not straightforwardly lend itself to greedy application, we accomplish this by developing a decomposed version of

`pdag-to-dag`. Let $\mathcal{G}_0$ be a PDAG with only v-structures oriented, and let $\mathcal{G}$ be a DAG consisting of only the directed edges in $\mathcal{G}_0$. We prioritize checking for and removing all complete sinks from consideration by deleting all edges incident to such nodes in $\mathcal{G}_0$, and we greedily consider orienting $i \rightarrow j$ in $\mathcal{G}_0$ and $\mathcal{G}$ if $i - j$ is an undirected edge in $\mathcal{G}_0$ and conditions (a) and (b) are satisfied for node j. For example, in a single greedy step applied to Fig. 2a, we would first remove X_8 from $\mathcal{G}_0$ as a complete sink, resulting in nodes X_1, X_6, and X_7 satisfying (a) and (b). We then greedily consider the individual edge orientations $X_4 \rightarrow X_1$, $X_5 \rightarrow X_1$, $X_5 \rightarrow X_7$, and $X_3 \rightarrow X_6$, applying the orientation that most improves the score computed with respect to the structure of $\mathcal{G}$ (i.e., all edges that have determined orientations). This design essentially decomposes the node-centric operations in `pdag-to-dag` into single edge operations (e.g., $X_4 \rightarrow X_1$ and $X_5 \rightarrow X_1$ are considered as individual orientations instead of both being considered with node X_1), and its result is a DAG in the same equivalence class as the output of `pdag-to-dag` given a valid PDAG. In practice, as with the sequential v-structure application, the greedy ordering filters edges and selects between ambiguous orientations. In the case that undirected edges still exist in $\mathcal{G}_0$ and no node satisfying (a) and (b) can be found, we likewise greedily consider transformations compelled by Meek's rules R1–4 applied to $\mathcal{G}_0$. We detail the HGI strategy in Algorithm 4.

Algorithm 4 `HGI`($\mathcal{G}_0$, $\mathcal{D}$, **U**) (sample version)

---

input: undirected graph $\mathcal{G}_0$, data $\mathcal{D}$, and v-structures **U**
output: DAG $\mathcal{G}$

1: initialize $\mathcal{G}$ as the empty graph on **V**
2: **repeat**
3:    orient $i \rightarrow k$ and $j \rightarrow k$ in $\mathcal{G}_0$ and $\mathcal{G}$ for $(i, k, j) \in \mathbf{U}$ that most improves $\phi(\mathcal{G}, \mathcal{D})$ and does not induce any cycles or conflict with any v-structures in $\mathcal{G}$
4: **until** no such improvement possible
5: **repeat**
6:    delete all edges incident to complete sinks in $\mathcal{G}_0$
7:    apply the first applicable of the following operations for $i - j$ in $\mathcal{G}_0$:
      (i) apply $i \rightarrow j$ to $\mathcal{G}_0$ and $\mathcal{G}$ satisfying (a) and (b) that most improves $\phi(\mathcal{G}, \mathcal{D})$
      (ii) delete $i - j$ from $\mathcal{G}_0$ satisfying (a) and (b) that if applied, would most deteriorate $\phi(\mathcal{G}, \mathcal{D})$
      (iii) apply $i \rightarrow j$ to $\mathcal{G}_0$ and $\mathcal{G}$ compelled by R1-4 that most improves $\phi(\mathcal{G}, \mathcal{D})$
      (iv) delete $i - j$ from $\mathcal{G}_0$ with $i \rightarrow j$ compelled by R1-4 such that if applied, would most deteriorate $\phi(\mathcal{G}, \mathcal{D})$
8: **until** no such operation possible

Important to note is that in the population setting, v-structure detection and orientation is order-independent, and while the particular DAG obtained by (our decomposed) `pdag-to-dag` is order-dependent, it will always recover a DAG in the same equivalence class if successful (i.e., a consistent extension of the input PDAG exists). In such a case, whatever ordering imposed on both or either of the heuristics has no meaningful effect on the result. Furthermore, given a greedy criterion, a locally consistent score will asymptotically accept proposed additions of truly connected edges due to Eq. (13), thus preserving guaranteed

identification of the equivalence class of the underlying DAG. Indeed, in such a setting, a lenient score that prefers denser graphs is sufficient as only property Eq. (13) is required.

In the finite-sample setting, incorrectly inferred conditional independence information can result in the determination of incomplete or extraneous and even conflicting v-structures, and could result in PDAGs that do not admit a consistent extension (Remark 2). The outcome of a naive non-greedy application of v-structures and (our decomposed) pdag-to-dag empirically varies in quality depending on the order by which the operations are applied due to conflicting operations and obstacles induced by the acyclicity constraint, providing the primary incentive for greedy decisions regarding proposed constraint-based orientations. From a constraint-based learning perspective, greedy forward stepping imposes a greedy ordering on the application of v-structures and other potentially conflicting or ambiguous edge orientations, while additionally providing an element of selectivity by disregarding operations that deteriorate the score.

As already discussed, Algorithm 4 asymptotically preserves sound and complete orientation of the skeleton of the underlying DAG $\mathcal{G}$ to a DAG in its equivalence class, straightforwardly evident from our discussion thus far.

Lemma 1 *Suppose that probability distribution P is fixed and faithful to a DAG $\mathcal{G}^*$, $\mathcal{D}_n$ is data containing n i.i.d. samples from P, and ϕ is a score satisfying local consistency. Let $\hat{\mathcal{G}}_n$ be the output of Algorithm 4. If $\mathcal{G}_0$ is the skeleton of $\mathcal{G}^*$ and **U** contains the v-structures of $\mathcal{G}^*$, then $\hat{\mathcal{G}}_n$ is in the same equivalence class as $\mathcal{G}^*$ with probability approaching one as $n \to \infty$.*

Note that while we state Lemma 1 assuming possession of all v-structures **U** that are present in the underlying DAG, these may be correctly obtained asymptotically depending on what information is available from the skeleton estimation method, which we discuss in Appendix B.

Indeed, neither operations (ii)–(iv) in line 7 nor any subsequent score-based search is necessary for Lemma 1 to hold, but rather serve in a corrective capacity in the finite-sample setting. The process of completing and deleting sinks to uncover subsequent sinks in $\mathcal{G}_0$ requires decisions for each undirected edge participating in a node satisfying (a) and (b) in order to continually progress in the algorithm. Operation (ii) discards each proposed edge addition $i \to j$ that deteriorates the score when no improvement according to (i) is possible so that j can be completed and removed. In the case that no node satisfying both (a) and (b) can be found, we apply the same greedy criterion to all edges compelled by Meek's rules R1–4 in operations (iii) and (iv). These rules are not subject to a leaf-to-root construction and often help resume applications of (i) and (ii), for example by deleting an undirected edge $i - j$ participating in an unshielded triple from $\mathcal{G}_0$ so that j can satisfy (b).

Note that in finite-sample applications, repeated application of (i)–(iv) does not guarantee orientation or deletion of all undirected edges in $\mathcal{G}_0$ [e.g. consider an undirected square where no vertex satisfies (a) and (b) and no edge is compelled by R1–4], though we empirically find it to typically address most if not all edges. Furthermore, while the adjacency criterion (b) exists to prevent the creation of additional v-structures, it is still possible for new v-structures to be created by deletion. Consider an undirected triangle in $\mathcal{G}_0$ where all three vertices i, j, and k satisfy (a) and (b). The greedy ordering may orient $i \to k$ and $j \to k$, remove node k once it is a complete sink, and eventually delete $i - j$, leaving $i \to k \leftarrow j$ as a new v-structure in $\mathcal{G}$.

While essentially equivalent in the large sample setting, directly executing a greedy decomposed `pdag-to-dag` poses a number of pragmatic advantages over first greedily applying Meek's rules in the presence of finite-sample error. The sink criterion (a) effectively accomplishes acyclicity checks for each proposed edge orientation, which grow increasingly computationally burdensome for larger networks. It additionally induces a leaf-to-root construction with operations that minimally conflict with subsequent operations, with $i \rightarrow j$ only denying i from satisfying (a) until j is removed. This further strengthens the effect of the greedy ordering in minimizing ambiguity in the initial DAG construction process. Indeed, considering Theorem 1, the order of greedy v-structure application is unambiguous given a score equivalent metric. In contrast, HC from an EG restricted to sparse candidates $\mathbf{A}$ begins with $O(|\mathbf{A}|^2)$ ambiguous edge additions where, for any distinct node pair $(i, j) \in \mathbf{A}$, adding the edge $i \rightarrow j$ or $j \rightarrow i$ results in the same score improvement, again evident from Theorem 1. HC may encounter many such non-unique edge additions which are typically decided according to a node ordering that is often arbitrary, and their compounding effect can result in conflicts that, together with the acyclicity constraint, entrap HC in local solutions.

Remark 3 Relevant to our work is the PEF framework by Gu and Zhou (2020), a hybrid strategy consisting of a final fusion step that is conceptually analogous to a non-greedy form of sparse candidate HC initialized with the directed edges of an estimated PDAG rather than an EG. The algorithm removes all undirected edges from a PDAG input and performs local edge additions, reversals, and deletions to the resulting DAG that improve the overall score by repeatedly iterating through the surviving node pairs in a semi-arbitrary order, simultaneously testing for conditional independence. While they empirically demonstrated this process to correct many of the errors in the estimated structure, we find that the order with which the edges are visited can result in varying degrees of improvement, and the testing strategy performs redundant conditional independence tests. Furthermore, naively initializing a score-based search with the PDAG output of a constraint-based algorithm may prove volatile given the sensitivity of `skel-to-cpdag` to erroneous conditional independence inferences as well as its order-dependence. Lastly, even if initialized by the DAG consisting of the compelled edges of the underlying DAG and perfectly restricted to true connections, neither PEF nor HC with a consistent score guarantees asymptotic orientation to a DAG in the equivalence class of $\mathcal{G}$.

Algorithm 5 pHGS($\mathcal{D}$, α, τ, $\alpha^{(\tau)}$) (sample version)

---

input: data $\mathcal{D}$, threshold α, number of estimates τ, minimum threshold $\alpha^{(\tau)}$
output: DAG $\mathcal{G}$
  1: execute pPC($\mathcal{D}, \alpha$) (the sample version of Algorithm 2), omitting edge orientation in line 19, to obtain Φ and $\mathbf{S}$ as in Eq. (10)
  2: execute $\mathcal{G}^{(t^*)} \leftarrow$ PATH($\Phi, \mathbf{S}, \mathcal{D}, \tau, \alpha^{(\tau)}$) (Algorithm 3) with the following modifications:
      line 5: detect v-structures $\mathbf{U}^{(t)}$ with $\mathbf{S}^{(t)}$ and execute $\mathcal{G}^{(t)} \leftarrow$ HGI($\mathcal{G}^{(t)}, \mathcal{D}, \mathbf{U}^{(t)}$) (Algorithm 4)
      line 6: copy $\tilde{\mathcal{G}}^{(t)} \leftarrow \mathcal{G}^{(t)}$
  3: execute hill-climbing (or otherwise greedy search) to obtain $\mathcal{G}$, initialized with the selected estimate $\mathcal{G}^{(t^*)}$ and restricted to $\mathbf{A} = \{(i,j) : \Phi_{ij} \leq \alpha^{(1)}\}$ where $\alpha^{(1)} = \alpha$ is the maximum p-value (see Sect. 3.2)

Finally, we detail in Algorithm 5 the partitioned hybrid greedy search (pHGS) algorithm, a composition of pPC, PATH, and HGI. The pHGS algorithm efficiently restricts the search space with the pPC algorithm (Algorithm 2), obtaining Φ and $\mathbf{S}$ as in Eq. (10) for use in PATH. Instead of generating τ CPDAG estimates with skel-to-cpdag, PATH instead obtains τ DAG estimates by detecting v-structures $\mathbf{U}^{(t)}$ in each thresholded skeleton $\mathcal{G}^{(t)}$ and executing HGI (Algorithm 4). See Appendix B for details on v-structure detection. The highest-scoring of the τ estimates $\mathcal{G}^{(t^*)}$ is selected to initialize HC (or an alternate score-based search algorithm) restricted to the active set $\mathbf{A} = \{(i,j) : \Phi_{ij} \leq \alpha^{(1)}\}$. We choose the maximum threshold $\alpha^{(1)}$ (see Sect. 3.2) for the restriction instead of $\alpha^{(t^*)}$ corresponding to the highest-scoring estimate to reduce false negatives that excessively restrict the score-based exploration in the finite-sample setting.

In the large-sample limit, under the same conditions and parameter specifications as Theorem 3 and Lemma 1, the output of pHGS (Algorithm 5) is a DAG that is Markov equivalent to the underlying DAG. Indeed, this result is already achieved by the modified PATH in line 2, and in such a case the subsequent greedy search exists only to verify its optimality.

5 Numerical results

We conducted extensive simulations to demonstrate the merits of pPC, PATH, HGI, and pHGS alongside a number of other popular structure learning algorithms. In addition to considering data simulated from various reference Bayesian network configurations, which we introduce in Sect. 5.1, we analyze real data in the form of a well-known flow cytometry dataset in Sect. 5.4.

5.1 Simulation set-up

The performance of our methods were primarily evaluated in comparison to several structure learning algorithms on numerous reference Bayesian networks obtained from the

Table 1 Networks for data generation consisting of κ connected sub-networks with p nodes, $|\mathbf{E}|$ edges, average number of neighbors $|\mathbf{N}^{\mathcal{G}}|$, maximum in-degree $\max_i |\mathbf{\Pi}_i^{\mathcal{G}}|$, and $|\mathbf{\Theta}|$ number of parameters

	Network	κ	p	$|\mathbf{E}|$	$\overline{|\mathbf{N}^{\mathcal{G}}|}$	$\max_i |\mathbf{\Pi}_i^{\mathcal{G}}|$	$|\mathbf{\Theta}|$
1	EARTHQUAKE	200	1000	1103	2.206	2	2380
2	CANCER	200	1000	1123	2.246	2	2399
3	ASIA	125	1000	1243	2.486	2	2544
4	SURVEY	167	1002	1347	2.689	2	4444
5	ANDES	5	1115	2245	4.027	6	7082
6	WIN95PTS	14	1064	2184	4.105	7	9525
7	CHILD	50	1000	1309	2.618	2	11,649
8	ALARM	28	1036	1689	3.261	4	16,574
9	MIX	14	1011	1877	3.713	7	17,319
10	SACHS	91	1001	1822	3.640	3	18,746
11	PIGS	3	1323	2182	3.299	2	20,102
12	HEPAR2	15	1050	2077	3.956	6	23,007
13	INSURANCE	38	1026	2120	4.133	3	40,918
14	HAILFINDER	18	1008	1541	3.058	4	54,322
15	WATER	39	1014	2303	4.542	5	72,316
16	MUNIN1	7	1064	1776	3.338	3	83,208
17	PATHFINDER	10	1090	1968	3.611	5	96,037
18	DIABETES	3	1239	2035	3.285	2	302,008
19	MILDEW	29	1015	1913	3.769	3	345,575
20	BARLEY	21	1008	2101	4.169	4	1,771,800

Bayesian network repository compiled alongside the R package bnlearn (Scutari 2010, 2017). Most available discrete networks were considered, with the MUNIN networks represented by MUNIN1 and the LINK network omitted because certain minuscule marginal probabilities required much larger sample sizes to generate complete sets of valid variables. The following preprocessing procedures were applied to each network. For each random variable X_i, non-informative states x_i with $\Pr(X_i = x_i) = 0$ were removed, and non-informative variables X_i with $|r_i| = 1$ were likewise removed. Furthermore, each variable X_i was restricted to $|r_i| \leq 8$, with the extraneous discrete states of excessively granular variables removed by randomly merging states. The conditional probability distributions imposed by merged states were averaged, weighted according to their marginal probabilities.

In order to demonstrate the effectiveness of our methods for learning large discrete networks, we generated larger versions of each network with a house implementation of the tiling method proposed by Tsamardinos et al. (2006b), modified to approximately preserve the average in-degree amongst non-minimal nodes. In particular, let $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ be the structure consisting of κ disconnected subgraphs to be connected by tiling. For a minimal node k (that is, k has no parents), instead of probabilistically choosing the number of added interconnecting edges, denoted by e_k, according to $e_k \sim \text{Unif}\{0, d := \max_{i \in \mathbf{V}} |\mathbf{\Pi}_i^{\mathcal{G}}|\}$, we let $\Pr(e_k = a) = \sum_{i \in \mathbf{V}} \mathbb{1}\{|\mathbf{\Pi}_i^{\mathcal{G}}| = a\}/|\mathbf{V}|$ for $a = 0, \ldots, \min\{d, 4\}$. Note that in this process we did not enforce any block structure on the tiled structures.

The considered networks, along with select descriptive characteristics, are presented in Table 1, ordered by increasing complexity. The MIX network consists of the 14

networks from Table 1 with the least complexity, tiled in random order. For each network configuration, we generated $N = 100$ datasets with $n = 25,000$ data samples each, for a total of 2000 datasets. The p columns of each dataset were randomly permuted so as to obfuscate any information regarding the causal ordering. Note that while only one sample size was considered for all the networks of similar order in p, the networks vary significantly with respect to sparsity, structure, and complexity, thus representing a wide variety of conditions.

Algorithm implementations of competing algorithms MMPC, HPC, HITON, IAMB, MMHC, and H2PC, which we briefly introduce in their respective featuring sections, were obtained from the R package `bnlearn`, which is written in R with computationally intensive operations delegated to C (R Core Team 2021; Scutari 2010, 2017). We also compared against GES and FGES from `rcausal`, the R package wrapper for the Tetrad Library (Wongchokprasitti 2019). Our pPC, PATH, HGI, and pHGS implementations were built in R and Rcpp using tools from the `bnlearn` package, and the results for PC were obtained by executing pPC restricted to $\kappa = 1$ for fair comparison. We have made our methods publicly available in the form of an R package at https://github.com/jirehhuang/phsl.

We evaluate the quality of a graph estimate $\hat{\mathcal{G}} = (\mathbf{V}, \hat{\mathbf{E}})$ with respect to the underlying DAG $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ by considering the Jaccard index (JI) of the CPDAG of $\hat{\mathcal{G}}$ in comparison to the CPDAG of $\mathcal{G}$. JI is a normalized measure of accuracy (higher is better) that is monotonically associated with the F1 score, computed as $\mathrm{JI} = \mathrm{TP}/ (|\mathbf{E}| + |\hat{\mathbf{E}}| - \mathrm{TP})$ where TP is the number of true positive edges: the number of edges in the CPDAG of $\hat{\mathcal{G}}$ that coincide exactly with the CPDAG of $\mathcal{G}$ (both existence and orientation).

We use the JI as our primary accuracy metric over the popular structural Hamming distance (SHD) as we find it to be consistent with SHD (higher JI almost always indicates lower SHD) and for its convenience as a normalized metric. The choice of evaluating the CPDAG estimates rather than DAG estimates is motivated foremost by the fact that given that our estimates are inferred from observational data, the orientation of reversible edges in DAGs provide no meaningful interpretation (see Sect. 2.1). Additionally, the aforementioned metrics allow for evaluation of the quality of estimated PDAGs that do not admit a consistent extension (see Remark 2).

Regarding efficiency, execution time is confounded by factors such as hardware, software platform, and implementation quality. Even if the aforementioned variables are accounted for, performing all simulations on the same device cannot guarantee consistent performance over all simulations and instead severely constrains the feasible scope of study. Thus, we evaluate the estimation speed of structure learning algorithms by the number of statistical calls to conditional independence tests or local score differences, with fewer calls indicating greater efficiency. For pPC, we additionally include mutual information and entropy evaluations to account for the expense of clustering (see Sect. 3.1.1).

Remark 4 In general, we find the relative execution time comparisons to be approximately equivalent to the relative number of statistical calls, within comparable implementations. For example, the relative estimation speed of pPC as compared to PC is approximately equivalently characterized by their relative execution time and relative number of statistical calls. The implementation of HC, however, does not scale as well as that of pPC, resulting in disproportionately greater relative execution times compared to relative number of statistical calls.

5.2 pPC and PATH

As the pPC algorithm can be considered an augmentation of the PC algorithm by imposing an ordering to the conditional independence tests by partitioning, we highlight its performance against the PC algorithm. We additionally apply the PATH augmentation to pPC and PC.

Note that our proposed HGI strategy motivates the design of high-performing constraint-based algorithms that not only efficiently restrict the search space, but also demonstrate potential for good score-based search initialization with HGI by producing structurally accurate estimates. As such, we further validate the performance of pPC and PATH against four other established constraint-based structure learning algorithms, all local discovery methods, modified with a symmetry correction for skeleton learning (Aliferis et al. 2010). Max-min parents and children (MMPC) uses a heuristic that selects variables that maximize a minimum association measure before removing false positives by testing for conditional independence (Tsamardinos et al. 2003b, 2006a). The fast version of the incremental association Markov blanket algorithm (Fast-IAMB; IAMB in this paper) is a two-phase algorithm that consists of a speculative stepwise forward variable selection phase designed to reduce the number of conditional independence tests as compared to single forward variable selection, followed by a backward variable pruning phase by testing for conditional independence (Tsamardinos et al. 2003a; Yaramakala and Margaritis 2005). The semi-interleaved implementation of HITON[1] parents and children (SI-HITON-PC; HITON in this paper) iteratively selects variables based on maximum pairwise marginal association while attempting to eliminate selected variables by testing for conditional independence (Aliferis et al. 2003, 2010). Finally, HPC is comprised of several subroutines designed to efficiently control the false discovery rate while reducing false negatives by increasing the reliability of the tests (Gasse et al. 2014). For each of these methods, following skeleton estimation, we orient edges by detecting and orienting v-structures according to Eq. (22) (in Appendix B) and applying Meek's rules R1–4.

The maximum size of considered conditioning sets m was chosen empirically for a balance between efficiency and well-performance: $m = 3$ for pPC and PC, $m = 4$ for HPC, and $m = 5$ for the remaining methods. We note that HPC insignificantly varies in efficiency with m and produces the most accurate estimates in our simulations with $m = 4$, and the remaining competing methods are more efficient but significantly less accurate with $m < 5$. Additionally for pPC, we restricted the maximum set size to 3 for the evaluations in Eqs. (7) and (8), and the maximum considered neighborhood size to 5 (see Remark 1). We executed each algorithm with each of the following ten choices of significance level thresholds:

$$\alpha \in \mathcal{A} := \{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005,$$
$$0.0001, 0.00005, 0.00001, 0.000005\}. \tag{15}$$

For each execution of pPC and PC, estimates for $\tau = 10$ thresholding values were automatically generated with PATH (Algorithm 3) according to Eq. (11), restricted to a minimum value of $\alpha^{(\tau)} = 10^{-5}$.

The comparison results for pPC and PATH are reported in Table 2. We first compare pPC against PC in terms of computational efficiency and estimation accuracy.

---

[1] From the Greek word "Χιτον", pronounced "*hee-tón*", meaning "cover", "cloak", or "blanket".

Table 2 Accuracy (JI) and efficiency (Normalized Calls) comparison between pPC and PC, without and with PATH (indicated by None and $\tau = 10$, respectively)

	JI						Normalized Calls[a]		
	pPC		PC		pPC*	PC*	PC	pPC*	PC*
	$\alpha = 0.1$		$\alpha = 0.1$		$\alpha \in \mathcal{A}$	$\alpha \in \mathcal{A}$	$\alpha = 0.1$	$\alpha \in \mathcal{A}$	$\alpha \in \mathcal{A}$
	None	$\tau = 10$	None	$\tau = 10$	None	None	$\tau = 10$	None	None
1	0.447	**0.725**	0.390	0.697	0.712	0.707	3.261	8.516	11.384
2	0.243	**0.333**	0.259	0.332	0.292	0.292	7.905	7.264	15.643
3	0.218	**0.284**	0.211	0.275	0.261	0.261	1.435	4.580	5.122
4	0.322	**0.362**	0.309	**0.362**	0.360	0.360	2.821	7.914	10.321
5	0.497	0.560	0.487	**0.561**	0.515	0.505	4.384	7.668	12.707
6	0.403	**0.411**	0.382	0.404	0.408	0.398	2.500	8.358	10.889
7	0.564	**0.579**	0.536	0.562	0.569	0.548	1.586	8.446	9.355
8	0.474	0.478	0.469	**0.480**	0.477	0.473	1.812	8.150	9.845
9	0.517	**0.546**	0.504	0.536	0.519	0.506	1.297	7.456	8.068
10	0.542	**0.551**	0.506	0.522	0.543	0.508	1.896	8.204	9.908
11	0.851	0.852	0.867	**0.871**	0.851	0.867	1.562	8.910	10.641
12	0.204	**0.233**	0.197	0.224	0.204	0.199	2.236	7.675	9.815
13	0.404	**0.424**	0.401	0.419	0.408	0.404	1.900	8.123	9.886
14	0.337	0.360	0.341	**0.361**	0.340	0.346	1.466	8.582	9.591
15	**0.305**	0.298	0.291	0.272	**0.305**	0.291	1.875	8.501	10.223
16	0.086	0.086	0.087	0.087	0.088	**0.090**	2.009	8.573	13.332
17	0.053	0.053	0.053	0.052	**0.054**	0.053	1.266	8.805	9.860
18	0.219	0.219	0.262	0.262	0.239	**0.263**	1.765	8.705	11.573
19	0.324	0.297	0.312	0.274	**0.326**	0.313	1.665	8.676	10.256
20	0.158	0.157	0.149	0.149	**0.168**	0.160	1.684	8.239	9.662

Columns pPC* and PC* provide the highest JI and total statistical calls of executions for all 10 $\alpha \in \mathcal{A}$. Rows correspond to the networks in Table 1, and best values are provided in boldface

[a]Normalized by pPC-PATH($\alpha = 0.1, \tau = 10$)

Unsurprisingly, pPC demonstrates the greatest computational benefit over PC for large α, typically halving the number of conditional independence tests for $\alpha = 0.1$, as seen from the normalized calls of PC with $\alpha = 0.1$ and PATH ($\tau = 10$) in the table. Note that our partitioning strategy is solely responsible for this computational improvement as parallelization has no bearing on the efficiency metric, and by design pPC can, like PC, further benefit from parallel execution. The reduction suffers from diminishing returns with decreasing α, with an average speed-up of about 22% across the ten α. Notwithstanding, we found pPC and PC, even without PATH, to generally prefer larger α. In particular, estimates with thresholds $\alpha = 0.1$ and 0.05 respectively produced the best estimates (highest JI) for over 50% and 17% of datasets, resulted in the highest average JI scores for 13 and 3 network configurations, and achieved the highest two JI scores averaged across all datasets. As such, algorithm executions with large significance level thresholds are not unreasonable in practice, which coincides with the general strategy of PATH. We note that while pPC appears to be typically slightly more accurate than PC, we do not find the improvement to be substantial and primarily assert that pPC performs comparably to PC.

Additionally, we see from Table 2 that PATH applied to pPC and PC is able to obtain, from a single execution with $\alpha = 0.1$ and $\tau = 10$, estimates of similar and often superior quality compared to the best estimates without PATH (pPC* and PC*) obtained from 10 executions with the various $\alpha \in \mathcal{A}$. Important to note is that the solution path automatically selects an estimate based on a BIC selection strategy restricted to valid CPDAG estimates, if any (see Remark 2), whereas for the multiple executions the maximum JI (as computed with respect to the CPDAG of the underlying DAG) for each dataset was chosen. The BIC selection strategy appears less effective for a couple of networks (e.g., 15 and 19), where on average the original estimates without PATH were more structurally accurate than those chosen from a solution path. One explanation for the worse performance could be the presence of invalid CPDAG estimates. PATH prefers valid estimates, and may prefer lower-scoring valid estimates over more structurally accurate invalid estimates. In the case that all estimates are invalid, the semi-arbitrary DAG extension process can be volatile, resulting in structurally inaccurate estimates being selected. As anticipated in Sect. 3.2, the computational expense required to execute Algorithm 3 is practically negligible in comparison to skeleton estimation. The statistical calls for pPC and PC with PATH ($\alpha = 0.1$ and $\tau = 10$) include the scores evaluated for BIC selection from the generated solutions by PATH, and are practically indistinguishable from those without PATH, with the score evaluations typically consisting of less than 0.1% of the statistical calls.

In Table 3, we compare pPC with PATH against other constraint-based structure learning algorithms. We exclude PC as its comparison with pPC is thoroughly demonstrated in Table 2. Again, competing methods report optimal results and total statistical calls for the executions across the 10 significance levels $\alpha \in \mathcal{A}$. In terms of structural accuracy, the only algorithm that can compete against pPC is HPC, which outperforms pPC in 12 of the network configurations, sometimes by a substantial margin. However, when it comes to efficiency, there is no contest against the pPC algorithm, in most cases even if the number of calls were averaged across the 10 executions instead of summed. Additionally, pPC most often produced valid CPDAG estimates, succeeding with 44% of the datasets with only $\alpha = 0.1$ and $\tau = 10$ in contrast to from 10% by HPC to up to 41% by IAMB with $\alpha \in \mathcal{A}$.

In the Supplementary Information, we provide additional results for the comparisons of pPC against three constraint-based methods, PC, MMPC, and HPC, thus far discussed in Tables 2 and 3. We include detailed tables that report additional metrics as well as standard deviations, and a figure that visualizes the comparisons and variability of the accuracy and efficiency results across executions by including the results for individual datasets for each network (Fig. S6). Furthermore, we provide extended simulation results investigating the effect of the clustering quality of pPC on its subsequent structure learning, as well as evaluating the performance of PATH for varying sample sizes.

5.3 HGI and pHGS

Having discussed the theoretical merits of HGI in Sect. 4.2, we demonstrate the empirical performance of HGI and pHGS in this section, beginning with the comparison and application to the GSC framework. We refer to unrestricted HC as simply HC and perfectly restricted HC as GSC*. In general, for a restriction of the search space with constraint-based algorithm Alg, we refer to the GSC version as Alg-HC, and the version with HGI as Alg-HGI–HC. However, we refer to the versions of established algorithms MMHC and H2PC that are augmented with HGI as MMHC–HGI and H2PC–HGI.

Table 3 Accuracy (JI) and efficiency (Normalized Calls) comparison amongst constraint-based methods

	JI					Normalized calls[a]			
	pPC[b]	MMPC	HPC	IAMB	HITON	MMPC	HPC	IAMB	HITON
1	0.725	0.759	**0.760**	0.749	0.759	33.933	86.837	53.532	17.189
2	0.333	0.757	0.805	**0.870**	0.701	30.377	157.542	39.762	16.831
3	0.284	**0.306**	0.245	0.260	0.299	15.271	47.117	21.783	7.758
4	0.362	0.773	0.818	**0.835**	0.734	33.130	97.750	40.456	20.167
5	0.560	0.563	**0.692**	0.532	0.606	34.973	168.566	33.262	61.176
6	**0.411**	0.295	0.379	0.251	0.243	30.886	101.774	44.803	17.194
7	0.579	0.087	**0.717**	0.317	0.077	29.551	67.678	49.607	15.100
8	**0.478**	0.200	0.434	0.314	0.158	29.016	78.442	59.514	14.939
9	0.546	0.423	**0.633**	0.360	0.316	12.150	53.621	14.422	7.762
10	0.551	0.259	**0.724**	0.352	0.358	27.139	100.016	30.732	14.306
11	0.852	0.536	**0.974**	0.385	0.276	12.454	69.107	12.761	6.200
12	0.233	0.259	**0.399**	0.289	0.262	21.322	82.737	31.368	12.408
13	**0.424**	0.189	0.310	0.252	0.114	24.916	95.332	38.883	12.967
14	**0.360**	0.246	0.325	0.287	0.220	32.241	59.202	40.034	16.646
15	**0.298**	0.283	0.287	0.291	0.232	32.291	88.191	48.761	16.602
16	**0.086**	0.004	0.010	0.032	0.004	11.962	72.271	12.746	6.757
17	0.053	0.067	**0.077**	0.050	0.062	7.282	19.078	10.687	3.737
18	0.219	0.067	**0.221**	0.073	0.061	28.936	74.800	26.148	14.965
19	**0.297**	0.066	0.252	0.232	0.064	32.707	82.580	36.943	16.629
20	0.157	0.055	**0.244**	0.093	0.055	24.945	96.625	22.784	12.777

Rows correspond to the networks in Table 1, and the best values are provided in boldface

[a]Normalized with respect to pPC-PATH($\alpha = 0.1, \tau = 10$)

[b]pPC-PATH($\alpha = 0.1, \tau = 10$), while all other methods report the highest JI from and the (normalized) total statistical calls for the executions across the 10 $\alpha \in \mathcal{A}$

The HC phase of each algorithm was augmented with a tabu list to avoid $t_1 = 100$ previously visited DAG structures for $t_0 = 100$ suboptimal iterations (see Sect. 4.1). All score-based methods evaluated structures with the BIC score. We executed pHGS with significance level $\alpha = 0.01$, and generated and selected from $\tau = 10$ HGI estimates in PATH by thresholding to a minimum of $\alpha^{(\tau)} = 10^{-5}$.

The accuracy results for HGI and pHGS are summarized in Table 4. All methods other than HC, GSC*, and pHGS report the highest JI for the executions across the ten $\alpha \in \mathcal{A}$. In the first column, the performance of (unrestricted) HC leaves much to be desired with its generally lackluster structural accuracy in comparison with the hybrid methods. Exceptions exist, as anticipated by Table 3 in which constraint-based methods struggle to produce good estimates for some higher complexity networks, often inferring excessive false negatives that erroneously reduce the search space. In such cases, hybrid approaches are limited by their constraint-based component and thus perform worse than HC.

We first highlight the demonstrated improvement of initialization with HGI compared to the EG (as in the GSC framework) for different skeleton restriction methods. A perfect restriction to the true skeleton represents the most optimistic scenario for GSC and HGI, wherein all true positives are accessible and no false positives are considered. HGI

Table 4 Accuracy (JI) comparisons for CPDAGs estimated without and with HGI (indicated by EG for empty graph and HGI, respectively) given restrictions obtained by various skeleton methods

Restrict	None	True Skeleton		pPC		MMPC		HPC	
Initial	EG	EG	HGI	EG	HGI	EG	HGI	EG	HGI
Alias	HC	GSC*			pHGS[a]	MMHC		H2PC	
1	0.341	0.478	0.806	0.501	**0.746**	0.500	0.761	0.548	0.762
2	0.419	0.596	0.954	0.577	**0.729**	0.570	0.874	0.590	0.903
3	**0.323**	0.504	0.802	0.255	0.282	0.254	0.302	0.224	0.255
4	0.649	0.770	0.931	0.751	**0.786**	0.726	0.819	0.760	0.887
5	0.610	0.910	0.970	0.715	**0.746**	0.544	0.598	0.686	0.738
6	0.295	0.564	0.736	0.414	**0.467**	0.254	0.306	0.330	0.403
7	0.450	0.535	0.995	0.528	**0.851**	0.191	0.199	0.532	0.854
8	0.361	0.567	0.889	0.411	**0.515**	0.288	0.329	0.493	0.636
9	0.630	0.823	0.902	0.686	**0.726**	0.388	0.441	0.695	0.738
10	0.320	0.417	1.000	0.413	**0.807**	0.276	0.308	0.418	0.921
11	0.827	1.000	1.000	0.992	**0.991**	0.450	0.535	0.990	0.991
12	0.529	0.634	0.763	0.513	0.552	0.247	0.294	**0.585**	0.622
13	0.368	0.513	0.811	0.417	**0.610**	0.212	0.227	0.404	0.423
14	0.456	0.730	0.920	0.387	**0.449**	0.279	0.315	0.455	0.488
15	0.234	0.361	0.581	0.293	**0.399**	0.268	0.366	0.328	0.419
16	**0.261**	0.576	0.655	0.045	0.056	0.007	0.007	0.096	0.102
17	**0.358**	0.407	0.478	0.070	0.072	0.081	0.081	0.345	0.336
18	0.222	0.618	0.946	0.164	0.197	0.063	0.069	**0.241**	0.304
19	0.375	0.549	0.690	0.276	0.320	0.106	0.114	**0.440**	0.481
20	0.260	0.524	0.607	0.144	0.165	0.068	0.080	**0.289**	0.327

Rows correspond to the networks in Table 1, and best values amongst pHGS, HC, MMHC, and H2PC are provided in boldface

[a] pHGS($\alpha = 0.01$, $\tau = 10$); all other methods, excluding HC and GSC*, report the highest JI for the executions across the ten $\alpha \in \mathcal{A}$

additionally enjoys consideration of all true v-structures when detecting v-structures amongst unshielded triples according to Eq. (22). Unsurprisingly, perfectly restricted GSC uniformly improves on the performance of unrestricted HC. The addition of HGI achieves further improvements to structural accuracy of typically 28% and up to 139% (Table 4, True Skeleton). This same trend persists when comparing GSC without and with HGI for empirical skeleton estimation methods pPC, MMPC, and HPC, demonstrating the effectiveness of HGI. For the various skeleton estimation methods, the addition of HGI achieves estimates that are typically 14% and up to 120% more structurally accurate.

We now compare pHGS with established algorithms HC, MMHC, and H2PC, the best values amongst which are indicated in boldface in Table 4. In most of the network configurations, a single execution of pHGS learns estimates of higher quality than the best of the ten executions with $\alpha \in \mathcal{A}$ of MMHC and H2PC. Note that for the GSC framework, the goal of parameter tuning for α is to obtain a balance between true positives and true negatives. MMHC does not outperform pHGS in any meaningful capacity, and H2PC only substantially outperforms pHGS for higher complexity networks due to a mechanism in HPC to reduce false negative edges (Gasse et al. 2014). While the addition of HGI dramatically

Table 5 Median and 95% precision intervals of percent additional statistical calls by GSC and HGI–HC with respect to skeleton learning

	GSC	HGI–HC		
	HC	Detect	HGI	HC
pPC[a]	1.00 (0.04, 1.88)	0.00 (0, 0)	1.12 (0.05, 3.87)	0.52 (0.03, 1.21)
MMPC[b]	0.13 (0.01, 0.28)	0.90 (0.02, 51.64)	0.02 (0, 0.06)	0.06 (0.01, 0.16)
HPC[b]	0.22 (0.02, 0.41)	0.92 (0, 32.56)	0.04 (0.01, 0.08)	0.12 (0.02, 0.21)

Each data point represents one algorithm execution for each dataset

[a]pPC under HGI–HC represents pHGS executed with $\tau = 10$ and $\alpha = 0.01$

[b]MMPC and HPC include the results from every individual execution across the ten $\alpha \in \mathcal{A}$

improves the general accuracy of MMHC, only H2PC–HGI performs competitively with pHGS, reflective of the results in Table 3 where HPC rivaled pPC in the realm of structural accuracy. However, as we will see from our discussion of Table 5, the speed comparisons in Table 3 generally hold for these hybrid variants as well, with pHGS on average nearly an order of magnitude more efficient than H2PC and around 2.5 times more efficient than MMHC per execution, with or without HGI. HC only outperforms the hybrid methods for a few structures in which the latter overly restrict the search space. Overall, we find pHGS to be most well-performing method, followed by H2PC, MMHC, and HC. We provide detailed results for these methods in the Supplementary Information, with boxplots (Fig. S7) visualizing the accuracy comparisons across datasets for each network as well as tables with additional metrics.

In Table 5, we compare the computational expense of GSC and HGI–HC with respect to skeleton learning. Under HGI–HC, the Detect column refers to the computational expense of detecting v-structures necessary to execute HGI (Algorithm 4) (see Appendix B). While HGI–HC typically comes at greater computational cost compared to the HC step in GSC, the expense of either is largely negligible in comparison to that of skeleton learning, generally (and often significantly) fewer than an additional 2% statistical calls. Rare exceptions exist, in particular extreme cases where MMPC or HPC required a significant number of additional tests to detect v-structures. Here, pPC has a clear computational advantage, having the ability to detect v-structures using separation sets accrued throughout skeleton learning [see Eq. (21) in Appendix B], resulting in fewer than 4.8% additional calls for every dataset to execute HGI $\tau = 10$ times in PATH and perform HC from the chosen initial DAG. Other algorithms must conduct additional conditional independence tests to detect v-structures via Eq. (22), which can quickly add up if the learned skeleton structure has a significant number of unshielded triples $i - k - j$, or if either or both of $|\mathbf{N}_i^{\mathcal{G}}|$ and $|\mathbf{N}_j^{\mathcal{G}}|$ are large. On the topic of efficiency, unrestricted HC typically requires three to five times the number of statistical calls to execute as compared to pHGS, providing further validation for the hybrid approach.

In general, we find the initial DAG obtained by HGI (Algorithm 4) through the greedy application of v-structures and greedy decomposed `pdag-to-dag` to be typically superior in structural accuracy compared to the direct application of `skel-to-cpdag` (Algorithm 6), the standard edge orientation strategy of constraint-based algorithms. HGI exhibits the greatest median improvement of 16.7% over `skel-to-cpdag` when applied to pPC, followed by 9.5% with HPC, 7.3% restricted to the

skeleton of the underlying DAG (True Skeleton), and 1.9% with MMPC. In general, pPC detects the most v-structures as its detection criterion Eq. (21) may be considered less strict than Eq. (22) used by True Skeleton, MMPC, and HPC. Consequently, pPC generally detects a significant number of false positive v-structures, thus benefiting most significantly from the greedy v-structure determinations. The poor skeleton estimation performance of MMPC is likely responsible for its lackluster improvement, with its estimated skeletons generally containing the fewest unshielded triples corresponding to true v-structures in the underlying DAGs in comparison to pPC and HPC.

Next, we compare pHGS against GES, introduced in Sect. 4.1, along with one of its variants, which serve well as points of reference for evaluating pHGS as they share comparable theoretical guarantees in our investigative setting, being sound and complete for Bayesian network structure learning (Meek 1997; Chickering 2002a, b). Ramsey (2015) proposed the fast greedy (equivalence) search algorithm (in this paper, FGES), which featured a number of optimizations and adjustments to scale up GES to handle large networks. In addition to techniques such as parallelization and addressing redundant scoring, perhaps the most prominent improvement was that of blacklisting zero correlation edges when computing initial scores for the addition of edges to the EG. This drastically reduces the number of computations of local score differences. A highly optimized implementation of FGES is available in the R package rcausal, and we roughly recover an efficient implementation of the original GES algorithm by omitting the blacklisting step (Wongchokprasitti 2019).

In the detailed results in Table 6, in addition to the JI and $\log_{10}$ number of statistical calls (Calls), we report the number of predicted edges (P), true positives (TP), edges correctly present but with incorrect orientation (R), false positives (FP), and the SHD, which may be interpreted as the number of edge additions, deletions, and reversals required to traverse from one graph to another.

For most networks, pHGS executes significantly faster than GES, with a few exceptions, most notably in networks with higher complexity. With respect to FGES, we find pHGS to be approximately equivalently efficient in most cases, though considerably slower in some. The initial blacklisting feature in FGES functions similarly to that of Eq. (6) in the pPC algorithm, drastically reducing the number of considered edge connections. Furthermore, (F)GES conducts its greedy search in the reduced space of equivalence classes represented by CPDAGs rather than DAGs.

In terms of structural accuracy, pHGS performs competitively against GES and FGES. Consistent with the observations of Ramsey (2015), we find that the significant computational reductions achieved by FGES generally do not compromise the quality of learned structures compared to GES. The results for FGES are quite similar to that of GES for all metrics apart from number of statistical calls, though FGES does appear to obtain marginally sparser networks. As a hybrid algorithm that estimates the skeleton with a constraint-based approach, pHGS is much more restrictive in determining edge connections, resulting in significantly fewer true positives than (F)GES in some networks. As discussed previously in this section, in these scenarios, the pPC component excessively inferred false negatives that erroneously reduced the set of candidate edges. On the other hand, however, pHGS produced substantially fewer false positives than (F)GES for not only these networks but indeed in every network configuration. In such a way, pHGS appears more appropriate in applications where conservative estimation of edge connections is desired.

Table 6 Results comparing pHGS against GES and FGES

| | pHGS | GES | FGES | pHGS | GES | FGES | pHGS | GES | FGES |
	EARTHQUAKE			CANCER			ASIA		
P	911.3	1099.5	1082.0	1124.1	1203.3	1179.2	876.3	1199.7	1176.4
TP	**860.7**	680.3	686.8	947.3	**949.5**	947.0	466.3	**947.9**	941.1
R	**21.9**	227.5	220.1	**126.5**	135.0	131.3	383.7	177.4	**175.6**
FP	**28.7**	191.7	175.1	**50.2**	118.8	101.0	**26.3**	74.4	59.7
SHD	**271.1**	614.4	591.2	**225.9**	292.4	276.9	803.1	369.6	**361.7**
JI	**0.746**	0.447	0.459	**0.729**	0.690	0.699	0.282	0.634	**0.637**
Calls	**5.724**	6.395	5.735	5.739	6.326	**5.724**	5.745	6.374	5.751
	pHGS	GES	FGES	pHGS	GES	FGES	pHGS	GES	FGES
	SURVEY			ANDES			WIN95PTS		
P	1329.2	1431.0	1418.7	1853.6	2398.4	2341.8	1308.6	2023.0	1973.0
TP	**1177.5**	992.4	987.3	1751.8	**2009.2**	1934.6	1111.9	**1403.3**	1366.5
R	**145.2**	340.5	340.6	**57.8**	83.8	94.3	**158.2**	225.3	224.1
FP	**6.5**	98.2	90.7	**44.0**	305.4	312.9	**38.5**	394.4	382.4
SHD	**175.9**	452.8	450.4	**537.2**	541.1	623.4	**1110.6**	1175.1	1199.9
JI	**0.786**	0.556	0.556	0.746	**0.763**	0.730	0.467	**0.501**	0.490
Calls	5.742	6.237	**5.722**	5.924	6.438	**5.918**	**5.864**	6.327	5.893
	pHGS	GES	FGES	pHGS	GES	FGES	pHGS	GES	FGES
	CHILD			ALARM			MIX		
P	1298.8	1312.0	1304.6	1282.9	1649.4	1631.2	1547.8	1850.7	1829.6
TP	1198.1	**1239.0**	1200.5	1010.1	**1322.2**	1313.2	1440.1	**1592.2**	1567.6
R	100.7	**66.7**	99.4	269.9	203.8	**203.5**	94.2	126.0	130.2
FP	**0.1**	6.3	4.7	**3.0**	123.4	114.5	**13.5**	132.5	131.8
SHD	111.0	**76.3**	113.2	681.9	**490.2**	490.3	450.4	**417.4**	441.3
JI	0.851	**0.897**	0.853	0.515	**0.656**	0.654	0.726	**0.746**	0.733
Calls	5.798	6.277	**5.785**	5.837	6.306	5.862	6.223	6.317	**5.907**
	pHGS	GES	FGES	pHGS	GES	FGES	pHGS	GES	FGES
	SACHS			PIGS			HEPAR2		
P	1797.5	1864.4	1862.3	2170.6	2280.2	2280.6	1424.5	1745.2	1685.5
TP	**1615.7**	1544.3	1537.8	**2166.0**	2124.7	2122.2	1245.9	**1375.9**	1325.1
R	**181.9**	264.3	270.0	**4.6**	49.7	51.2	**158.5**	239.2	243.1
FP	**0.0**	55.8	54.4	**0.0**	105.7	107.2	**20.1**	130.1	117.3
SHD	**206.3**	333.6	338.6	**16.0**	163.0	167.0	851.2	**831.3**	869.2
JI	**0.807**	0.722	0.717	**0.991**	0.909	0.907	0.552	**0.563**	0.544
Calls	**5.847**	6.451	5.855	6.523	6.441	**6.139**	5.978	6.317	**5.822**
	pHGS	GES	FGES	pHGS	GES	FGES	pHGS	GES	FGES
	INSURANCE			HAILFINDER			WATER		
P	1512.0	1899.5	1848.4	986.8	1493.8	1480.8	1321.1	1672.4	1642.3
TP	**1375.2**	1198.9	1178.4	783.3	1191.0	**1192.9**	1032.5	**1048.8**	1033.8
R	**133.8**	479.0	479.5	**117.1**	143.6	135.1	**288.3**	427.3	427.5
FP	**3.0**	221.6	190.4	**86.4**	159.3	152.8	**0.3**	196.3	181.0

Table 6 (continued)

	pHGS	GES	FGES	pHGS	GES	FGES	pHGS	GES	FGES
	INSURANCE			HAILFINDER			WATER		
SHD	**747.8**	1142.8	1131.9	844.1	509.3	**500.9**	**1270.7**	1450.5	1450.2
JI	**0.610**	0.425	0.422	0.449	0.646	**0.653**	**0.399**	0.358	0.355
Calls	5.910	6.368	**5.868**	**5.790**	6.267	5.819	**5.779**	6.250	5.783

	pHGS	GES	FGES	pHGS	GES	FGES	pHGS	GES	FGES
	MUNIN1			PATHFINDER			DIABETES		
P	518.2	1629.6	1614.4	290.6	1458.5	1458.5	1053.3	1988.2	1972.2
TP	120.8	890.5	**891.3**	151.6	**1017.9**	1006.9	508.6	1303.6	**1305.0**
R	364.2	**215.7**	216.0	**135.6**	262.0	274.5	523.0	399.1	**395.2**
FP	**33.3**	523.4	507.1	**3.4**	178.7	177.1	**21.7**	285.5	272.0
SHD	1688.5	1408.8	**1391.8**	1819.8	**1128.8**	1138.2	1548.1	1016.8	**1002.0**
JI	0.056	0.354	**0.357**	0.072	**0.423**	0.417	0.197	0.480	**0.483**
Calls	6.437	6.423	**6.194**	6.531	6.560	**6.328**	**6.040**	6.496	6.043

	pHGS	GES	FGES	pHGS	GES	FGES
	MILDEW			BARLEY		
P	1094.8	1526.1	1502.7	758.6	1481.1	1462.7
TP	729.1	**1241.5**	1232.8	404.6	**1120.8**	1117.6
R	365.4	178.2	**178.0**	332.6	**212.9**	213.6
FP	**0.2**	106.4	91.9	**21.3**	147.4	131.5
SHD	1184.2	777.9	**772.1**	1717.7	1127.7	**1114.9**
JI	0.320	**0.565**	**0.565**	0.165	0.455	**0.457**
Calls	5.772	6.138	**5.752**	5.869	6.171	**5.782**

Calls reports $\log_{10}(\text{calls})$

5.4 Real data application

To evaluate our methods applied to real data, we analyzed a flow cytometry dataset generated by Sachs et al. (2005) through a series of experiments measuring $p = 11$ phosphorylated proteins and phospholipids. In particular, we considered their preprocessed dataset containing $n = 5400$ observations discretized to low, medium or high levels of the phosphorylated components, available in the R package sparsebn (Aragam et al. 2019). The structure of the currently accepted signalling network, which we refer to as the consensus network, is shown in Fig. 3a, and contains 20 directed edges.

We executed pHGS, MMHC, H2PC, HC, and FGES on this dataset, reporting the structure learning performance with respect to the consensus DAG. Since the dataset contains a mixture of observational and interventional data, we directly evaluate the DAG estimates against the consensus network instead of comparing their respective CPDAGs. For FGES, which outputs a CPDAG representing an equivalence class, we obtained a DAG extension by applying lines 5-8 of Algorithm 4 to evaluate the structure learning accuracy of FGES, which we note is significantly more favorable for FGES than comparing its CPDAG to the CPDAG of the consensus network. The hybrid algorithms pHGS, MMHC, and H2PC were executed with $\alpha \in \mathcal{A}$ in Eq. (15), with other parameters unchanged from Sect. 5.2 with the exception of $\alpha^{(\tau)} = 10^{-6}$ to

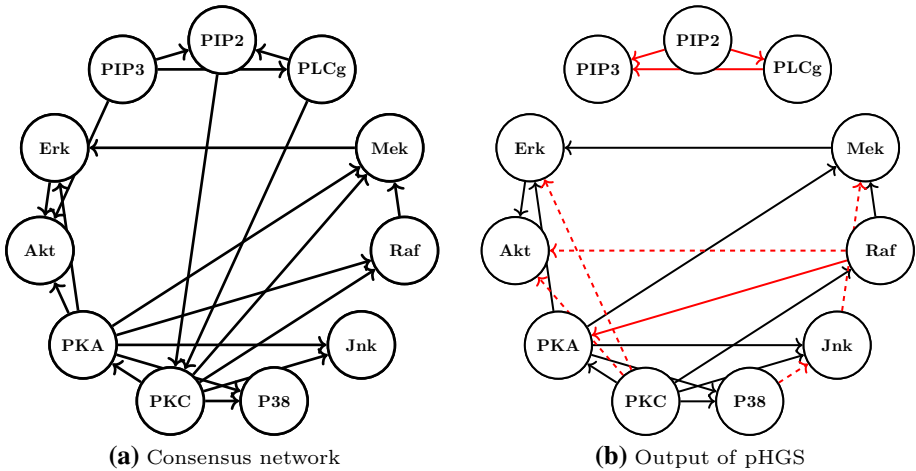**(a)** Consensus network **(b)** Output of pHGS

Fig. 3 Comparison of **a** the consensus network and **b** the DAG estimated by pHGS. In **b**, edges correctly present in the consensus graph are drawn in solid black, edges with incorrect orientation in solid red, and false positive edges in dashed red (Color figure online)

Table 7 Structure learning and difference in test log-likelihood performance on discretized cytometry data

	pHGS	MMHC	H2PC	HC	FGES
P	20	16	20	22	22
TP	**11**	8	8	7	6
R	**4**	**4**	7	7	7
FP	5	**4**	5	8	9
SHD (DAG)	**14**	16	17	21	23
SHD (skeleton)	**10**	12	**10**	14	16
JI	**0.379**	0.286	0.250	0.200	0.167

ensure $\alpha^{(\tau)} < \alpha$ for all executions of pHGS. For these methods, we present the results for the estimates closest in sparsity to the consensus graph (20 edges), which coincide with those with the highest JI. The results are reported in Table 7, and the pHGS estimate is shown in Fig. 3b.

By a substantial margin, pHGS produced the most structurally accurate estimate with respect to the consensus network, recovering 11 out of the 20 well-established causal relationships. The score-based algorithms FGES and HC obtained the densest networks as they do not conduct any restriction of the search space. In contrast, MMHC estimated the sparsest graph and indeed estimated the exact same graph for all 10 $\alpha \in \mathcal{A}$, obtaining different restrictions of the search space via MMPC yet arriving at the same network structure in the restricted HC step.

6 Discussion

In this paper, we proposed three independent yet compatible contributions to the general well-performance of discrete Bayesian network structure learning, culminating in the form of the pHGS algorithm.

First, the pPC algorithm improves on the empirical efficiency of the PC algorithm while retaining its soundness and completeness as well as its empirical performance. Though it is difficult to quantify the expected computational reduction, our simulation results in Table 2 indicate that for the empirically preferred significance level threshold, pPC typically requires less than half the number of conditional independence tests per execution compared to PC. This speed-up is enjoyed at no compromise to structural accuracy, with pPC performing comparably with PC. Second, the PATH algorithm effectively accomplishes the task of parameter tuning for certain constraint-based structure learning algorithms such as pPC and PC, theoretically preserving consistency in the classical setting and empirically achieving highly competitive structural accuracy at negligible computational expense. In the current landscape, the asymptotic result for sound and complete constraint-based structure learning asserts the existence of some uninformative sequence of significance levels $\alpha_n \to 0$ as $n \to \infty$ that recovers the underlying equivalence class in the large-sample limit. We prove that appropriately applied to pPC or PC executed with any fixed threshold $\alpha \in (0, 1)$, PATH asymptotically includes and correctly selects the underlying CPDAG in its generated solution path. We demonstrate an analogous result in the empirical setting, wherein pPC with PATH returns estimates of competitive quality to that of optimistic parameter tuning, achieving significant computational reductions compared to the current standard.

Third, the HGI algorithm provides a principled strategy for initializing score-based search in hybrid methods that asymptotically preserves soundness and completeness of constraint-based structure learning, elevating the GSC framework to consistency in the classical setting while significantly improving its empirical performance. While popular hybrid algorithms MMHC and H2PC forego asymptotic consistency for empirical performance, our HGI strategy makes no such compromise. When applied to GSC with various skeleton estimation strategies (including MMHC and H2PC), HGI significantly improves the estimation accuracy with typically negligible additional computational expense. Notably, a more recent development in hybrid structure learning is ARGES, which adaptively relaxes the restricted search space in order to ensure a search path in the space of equivalence classes (CPDAGs) to the optimal solution (Nandy et al. 2018). Though in this paper we take primary interest in improving upon the GSC framework which operates in the space of DAGs, we present preliminary simulation results in Sect. S1 of the Supplementary Information that indicate significant potential for empirical improvement to ARGES through the initialization provided by HGI.

Altogether, we combined pPC, PATH, and HGI to create the pHGS algorithm, which enjoys the skeleton estimation efficiency of pPC, the parameter tuning by PATH, and the empirical well-performance afforded by HGI. In comparison to MMHC and H2PC, pHGS learns more accurate DAGs in nearly every considered underlying network configuration with significantly reduced computational cost.

While we have empirically demonstrated a significant reduction to the number of conditional independence tests executed by pPC in comparison to PC, we have not established any concrete theoretical complexity results. Indeed, the extent of computational reduction inevitably depends on the quality of the partition with respect to the underlying structure. As such, it is of interest to determine under what conditions pPC is guaranteed to perform fewer tests than PC, and to quantify the reduction. Such an investigation is crucial to establish structure learning consistency of pPC in the sparse high-dimensional setting with multivariate Gaussian distributions. The high-dimensional Gaussian consistency of PC proved in Kalisch and Bühlmann (2007) relies on the number of conditional independence tests performed, determined by the maximum size of conditioning sets reached by PC with no

errors. The same result holds for pPC under the same assumptions if the number of tests investigated by pPC is not greater than that of PC. Note that empirically, we find pPC to always perform fewer tests than PC.

In Sect. 5, we observed superior structure learning performance by HPC and H2PC–HGI against pPC–PATH and pHGS, respectively, though not without significantly greater computational expense. We note that since the skeleton learning phase of HPC involves independently estimating the parents and children of each variable and aggregating the results, there are many obvious opportunities to more efficiently execute a synchronized global version of HPC for skeleton learning that avoids redundant computations. For instance, the first subroutine estimates a supergraph by identifying, for each node $i \in \mathbf{V}$, every node $j \in \mathbf{V} \setminus \{i\}$ that cannot be separated from i by conditioning sets $\{\mathbf{X} \subseteq \mathbf{V} \setminus \{i,j\} : |\mathbf{k}| \leq 1\}$. When considered node-wise, every conditional independence test is executed twice, once in the consideration of i as a neighbor of j and again in the consideration of j as a neighbor of i. A global partition-based strategy, such as the pPC algorithm with the restriction to conditioning sets $|\mathbf{k}| \leq 1$, can be utilized to substantially reduce the computational expense compared to the original formulation.

As formulated, pPC is limited to obtaining $\kappa \leq 20$ clusters as per a loose suggestion by Hartigan (1981). Note that by our design, pPC is not limited to parallel processing utility from at most κ processors, having comparable capacity for parallel execution as PC. Nonetheless, a future direction for further improvement would be the development of an unsupervised criterion to more flexibly determine the target number of clusters that optimizes the efficiency of pPC.

A Proofs

In this section, we prove Theorems 2 and 3.

A.1 Proof of Theorem 2

The proof of Theorem 2 regarding the soundness and completeness of the pPC algorithm (Algorithm 2) follows from the thorough investigation of the criterion expressed in Eq. (4) by the pPC algorithm.

Proof of Theorem 2

Let $\mathbf{c}$ be any clustering labels obtained in the clustering step. Let $\hat{\mathcal{G}}_1$ denote the estimated structure after estimating edges within clusters at line 3, and $\tilde{\mathcal{G}}$ denote the final estimated skeleton at line 18.

We first show that every truly adjacent node pair (that is, adjacent in $\mathcal{G}^*$) is also adjacent in the estimated structure at every stage of Algorithm 2 following line 10. An implication of the Markov condition (Sect. 2) is that no truly adjacent pair of nodes $i, j \in \mathbf{V}$ are independent conditioned on any set of variables not containing X_i or X_j. Thus, every truly adjacent pair of nodes belonging to the same cluster are never disconnected after being connected in the first execution of Algorithm 2 in line 2:

$$\mathbf{N}_i^{\hat{\mathcal{G}}_1} \supseteq \{X_k \in \mathbf{N}_i^{\mathcal{G}^*} : c_k = c_i\} \quad \text{for all } i \in \mathbf{V}. \tag{16}$$

Similarly, every truly adjacent pair of nodes belonging to different clusters will necessarily be connected in the first screening of edges between clusters, and are never disconnected. Thus, $\tilde{\mathcal{G}}$ is equivalent to or a supergraph of the skeleton of $\mathcal{G}^*$, with

$$\mathbf{N}_i^{\tilde{\mathcal{G}}} \supseteq \mathbf{N}_i^{\mathcal{G}^*} \quad \text{for all } i \in \mathbf{V}. \tag{17}$$

Next, we show that no truly nonadjacent node pair (that is, not adjacent in $\mathcal{G}^*$) will be connected in $\tilde{\mathcal{G}}$. Restating Eq. (4), for every truly nonadjacent distinct node pair $i, j \in \mathbf{V}$ there exists at least one conditioning set $\mathbf{X_k} \subseteq \mathbf{N}_i^{\mathcal{G}^*} \setminus X_j$ or $\mathbf{X_k} \subseteq \mathbf{N}_j^{\mathcal{G}^*} \setminus X_i$ such that $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$. Thus, it is sufficient to show that for every truly nonadjacent node pair, $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$ for one such separation set $\mathbf{S}(i,j) := \mathbf{k}$ is evaluated by Algorithm 2. Consider the following for any truly nonadjacent node pair $i, j \in \mathbf{V}$.

Suppose $c_i = c_j$. If there exists a conditioning set $\mathbf{X_k}$ such that $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$ and $c_k = c_j$ for all $k \in \mathbf{k}$, then due to Eq. (16), $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$ is evaluated by the first modified execution of PC in line 2. Otherwise, if $\exists k \in \mathbf{k}$ such that $c_k \neq c_j$, then for any and every conditioning set $\mathbf{X_k}$ such that $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$, $\mathbf{X_k}$ satisfies criteria (a) of Eq. (9). In such a case, because of Eq. (17), $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$ for one such conditioning set $\mathbf{X_k}$ is evaluated by the second modified execution of PC in line 18.

Suppose $c_i \neq c_j$. Ideally, i and j will be separated during the screening of edges between clusters in lines 3-17. If they are not, then for any and every conditioning set $\mathbf{X_k}$ such that $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$, $\mathbf{X_k}$ satisfies criteria (b) of Eq. (9). Since Eq. (17) holds, $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$ for one such $\mathbf{X_k}$ is evaluated by the second modified execution of PC in line 18.

We have shown that Algorithm 2 correctly estimates the skeleton of $\mathcal{G}^*$ and, in particular, for every truly nonadjacent $i, j \in \mathbf{V}$, $(X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P$ for some conditioning set $\mathbf{X_k}$ has been evaluated, with separation set $\mathbf{S}(i,j) := \mathbf{k}$ stored as specified in Algorithm 2. Since these separation sets are guaranteed to be accurate with conditional independence oracles, orientation of $\tilde{\mathcal{G}}$ to the CPDAG of $\mathcal{G}^*$ in line 19 follows directly from the correctness of `skel-to-cpdag` (Algorithm 6) (Spirtes and Glymour 1991; Meek 1995; Kalisch and Bühlmann 2007). □

A.2 Proof of Theorem 3

To prove Theorem 3, we begin by stating and proving a well-known consistency result for constraint-based structure learning in the classical setting.

Lemma 2 *Suppose the distribution P is fixed and faithful to a DAG with CPDAG $\mathcal{G}^*$ and $\mathcal{D}$ is data containing n i.i.d. samples from P. Let $\hat{\mathcal{G}}_n(\alpha_n) = (\mathbf{V}, \hat{\mathbf{E}}_n)$ be the graph output of any exhaustive investigation of Eq. (4) followed by Algorithm 6 executed with a consistent test applied with threshold α_n. Then there exists $\alpha_n \to 0 \ (n \to \infty)$ such that*

$$\lim_{n \to \infty} \Pr[\hat{\mathcal{G}}_n(\alpha_n) = \mathcal{G}^*] = 1.$$

Proof of Lemma 2 Define $p_{n;i,j|\mathbf{k}}$ and $p^*_{n;i,j|\mathbf{k}}$ be the p-values for testing the independence between i and j conditioned on $\mathbf{k}$ with n data samples when i and j are and are not d-separated by $\mathbf{k}$ in $\mathcal{G}^*$, respectively. For example,

$$p_{n;i,j|\mathbf{k}} = \Pr(\chi_f^2 > G_{n;ij|\mathbf{k}}^2 \mid (X_i \perp\!\!\!\perp X_j \mid \mathbf{X_k})_{\mathcal{G}}), \text{ and}$$

$$p_{n;i,j|\mathbf{k}}^* = \Pr(\chi_f^2 > G_{n;ij|\mathbf{k}}^2 \mid (X_i \not\perp\!\!\!\perp X_j \mid \mathbf{X_k})_{\mathcal{G}})$$

for the G^2 test of independence. Given faithfulness, d-separation in $\mathcal{G}^*$ corresponds one-to-one with conditional independence in P. In the case that conditional independence holds (i.e. the null hypothesis is true), $p_{n;i,j|\mathbf{k}} \sim \text{Unif}(0, 1)$, so $\Pr(p_{n;i,j|\mathbf{k}} \leq \alpha_n) = \alpha_n \to 0 \ (n \to \infty)$. Whereas whenever conditional independence does not hold, the consistency of the conditional independence test implies that $p_{n;i,j|\mathbf{k}}^* = o(1)$. As these statements hold for every $i, j \in \mathbf{V}$ and $\mathbf{k} \subseteq \mathbf{V} \setminus \{i, j\}$ with fixed $p = |\mathbf{V}|$, what follows is the existence of $\alpha_n \to 0$ such that the probability of making any erroneous conditional independence inference in the investigation of Eq. (4) decays to zero when $n \to \infty$. That is,

$$\lim_{n \to \infty} \Pr\left[\bigcup_{i,j,\mathbf{k}} \left(p_{n;i,j|\mathbf{k}} \leq \alpha_n \right) \cup \left(p_{n;i,j|\mathbf{k}}^* > \alpha_n \right) \right] = 0. \tag{18}$$

Given perfect conditional independence inferences asymptotically, the consistency follows straightforwardly from the correctness of Eq. (4) for skeleton identification and the soundness and completeness of Algorithm 6 for CPDAG orientation. $\qquad\square$

The proof of Theorem 3 then proceeds from Lemma 2 and its proof as follows.

Proof of Theorem 3 Equation (18) indicates the existence of some $a_n \to 0 \ (n \to \infty)$ such that

$$\lim_{n \to \infty} \Pr\left[\sup_{i,j,\mathbf{k}} p_{n;i,j|\mathbf{k}}^* \leq a_n < \inf_{i,j,\mathbf{k}} p_{n;i,j|\mathbf{k}} \right] = 1. \tag{19}$$

That is, in the large-sample limit, a_n perfectly discriminates true positives from true negatives. For conditional independence inferred with threshold $\alpha_n \geq a_n \geq \sup_{i,j,\mathbf{k}} p_{n;i,j|\mathbf{k}}^*$, only false positive edges are possible. This ensures that for every truly nonadjacent distinct node pair $i, j \in \mathbf{V}$ (that is, $i - j \notin \mathbf{E}^*$ where $\mathbf{E}^*$ is the edge set of the skeleton of $\mathcal{G}^*$), at least one conditioning set that separates i and j is considered in the investigation of Eq. (4) and stored in Eq. (10) since Eq. (19) holds asymptotically.

$$\max_{i-j \in \mathbf{E}^*} \Phi_{n;ij} \leq \sup_{i,j,\mathbf{k}} p_{n;i,j|\mathbf{k}}^* \leq a_n < \inf_{i,j,\mathbf{k}} p_{n;i,j|\mathbf{k}} \leq \min_{i-j \notin \mathbf{E}^*} \Phi_{n;ij}. \tag{20}$$

Inspecting Eq. (20) confirms the existence of a threshold value $\hat{a}_n$, in particular $\hat{a}_n = \max_{i-j \in \mathbf{E}^*} \Phi_{n;i,j}$, that perfectly discriminates true positives from true negatives. Consequently, $\mathbf{E}^* = \{i - j : \Phi_{n;ij} \leq \hat{a}_n\}$, and $\{\mathbf{S}_n(i, j) : \Phi_{n;ij} > \hat{a}_n\}$ obtains correct separation sets for complete and sound orientation to $\mathcal{G}^*$ according to Algorithm 6.

Having verified the existence of a threshold $\hat{a}_n$ that recovers $\mathcal{G}^*$ by thresholding Φ_n and $\mathbf{S}_n$, it remains to show that $\hat{a}_n$ is recovered and selected by Algorithm 3. Parameter settings $\tau = 1 + \sum_{i<j} \mathbb{1}\{\Phi_{n;ij} \leq \alpha_n\} = 1 + |\hat{\mathbf{E}}_n^{(1)}|$ and $\alpha^{(\tau)} = 0$ regulate the generation of $\alpha^{(t)}$ such that the skeletons of sequential estimates differ in sparsity by exactly one edge. The resulting values $\mathcal{A} := \{\alpha^{(1)}, \ldots, \alpha^\tau\}$ correspond to the order statistics of the upper triangular elements of Φ_n decreasing from $\alpha_n^{(1)} = \alpha_n \geq a_n \geq \hat{a}_n$ to $\alpha_n^{(\tau)} = 0$, ensuring that $\hat{a}_n = \max_{i-j \in \mathbf{E}^*} \Phi_{n;i,j} \in \mathcal{A}$ and so $\mathcal{G}^* \in \{\mathcal{G}^{(t)} : t \in \{1, 2, \ldots, \tau\}\}$. The consistency of ϕ ensures that in the large-sample limit, $t^* = \text{argmax}_{t \in \{1, \ldots, \tau\}} \phi(\mathbf{X} \mid \hat{\mathcal{G}}^{(t)}, \mathcal{D})$ selects $\alpha^{(t^*)} = \hat{a}_n$ that results in the true CPDAG $\hat{\mathcal{G}}_n^{(t^*)}(\alpha_n) = \mathcal{G}^*$, with high probability.

Given that the arguments hold for any α_n as long as $\alpha_n \geq a_n$ when n is large, it is not necessary for α_n to converge to zero. In particular, Eq. (12) holds for any fixed $\alpha_n = \alpha \in (0, 1)$. □

Note that while Theorem 3 generously allows for any $\alpha_n \in (0, 1)$, in practice a small threshold is desirable in the interest of efficiency. Since $\Pr(p_{n;i,j|\mathbf{k}} \leq \alpha_n) = \alpha_n$, the expected length of the solution path τ may be approximately bounded between $|\mathbf{E}^*|$ and $|\mathbf{E}^*| + \frac{p(p-1)}{2}\alpha_n$. The bounds may be further regulated if approximate knowledge of the sparsity of $\mathcal{G}^*$ is known.

Furthermore, an inspection of the proof of Theorem 3 more generally indicates that the solution path returned by Algorithm 3 with the same τ and $\alpha^{(\tau)}$ contains the CPDAG of the underlying DAG $\mathcal{G}^*$ if there exists some $a \leq \alpha$ such that

$$\max_{i,j\in\mathbf{E}^*} \Phi_{ij} \leq a < \min_{i,j\notin\mathbf{E}^*} \Phi_{ij}$$

holds when Eq. (4) is investigated with threshold α, which can hold even for a finite sample size n.

B Constraint-based edge orientation

In this section, we review established constraint-based strategies for determining edge orientations, summarized in Algorithm 6.

We begin by discussing the work of Verma and Pearl (1991) in determining edge orientations. Under faithfulness, knowledge about the conditional independence relationships between variables can be used to detect the existence of v-structures (defined in Sect. 2.1) as follows. A triplet of nodes (i, k, j) configured $i - k - j$ with i and j not adjacent, called an *unshielded triple*, is a v-structure if and only if

$$\exists \mathbf{k} \subseteq \mathbf{V} \setminus \{i, j, k\} \text{ such that } (X_i \perp\!\!\!\perp X_j | \mathbf{X_k})_P. \tag{21}$$

It is easy to see that any other directed configuration of $i - k - j$ requires k to separate i and j (Spirtes et al. 2000, Lemma 5.1.3). To investigate this criterion, the separation sets $\mathbf{S}$ are recorded throughout the estimation process (e.g., line 9 of Algorithm 1), defined as $\mathbf{S}(i,j) = \mathbf{S}(j,i) \subseteq \mathbf{V} \setminus \{i, j\}$ such that $(X_i \perp\!\!\!\perp X_j | \mathbf{X}_{\mathbf{S}(i,j)})_P$ for distinct nodes i and j. Assuming accurate separation sets, every v-structure in $\mathcal{G}$ can be recovered according to Eq. (21), guaranteeing extension of the skeleton of $\mathcal{G}$ to its pattern, to which Meek's rules can be straightforwardly applied to obtain its CPDAG (see Sect. 2.1).

Alternatively, considering the Markov condition, an unshielded triple (i, k, j) in $\mathcal{G}$ with adjacencies $\mathbf{N}_i^{\mathcal{G}}$ and $\mathbf{N}_j^{\mathcal{G}}$ can be correctly identified as a v-structure by investigating the criteria Eq. (21) limited to sets $\mathbf{k}$ such that $\mathbf{X_k} \subseteq \mathbf{N}_i^{\mathcal{G}}$ or sets $\mathbf{k}$ such that $\mathbf{X_k} \subseteq \mathbf{N}_j^{\mathcal{G}}$. Margaritis (2003) proposed the following criteria for identifying (i, k, j) as a v-structure:

$$(X_i \not\perp\!\!\!\perp X_j | \mathbf{X_k})_P \quad \text{for all } \mathbf{X_k} \subseteq \mathbf{N}_i^{\mathcal{G}} \quad \text{or for all } \mathbf{X_k} \subseteq \mathbf{N}_j^{\mathcal{G}} \quad \text{with } k \in \mathbf{k}, \tag{22}$$

investigating the smaller of $|\mathbf{N}_i^{\mathcal{G}}|$ and $|\mathbf{N}_j^{\mathcal{G}}|$ for efficiency. For constraint-based algorithms that do not record $\mathbf{S}$, we empirically prefer investigating Eq. (22) over Eq. (21) for both general well-performance and efficiency. As with Eq. (21), the correctness of Eq. (22) guarantees detection of all and only the v-structures in a DAG $\mathcal{G}$ faithful to P given its skeleton and conditional independence oracles.

Algorithm 6 skel-to-cpdag($\mathcal{G}$, **S**, $\{\perp\!\!\!\perp_P\}$)

---

input: undirected graph $\mathcal{G}$, and separation sets **S** or conditional independence information $\{\perp\!\!\!\perp_P\}$
output: CPDAG $\mathcal{G}$
1: initialize $\mathbf{U} = \emptyset$ to store v-structures
2: **for all** triplets of nodes configured $i - k - j$ in $\mathcal{G}$ where i and j are not adjacent **do**
3:      **if S** is supplied **then**
4:          **if** $k \notin \mathbf{S}(i, j)$ **then**
5:              store (i, k, j) in $\mathbf{U}$
6:          **end if**
7:      **else**
8:          let $q = \operatorname{argmin}_{l \in \{i,j\}} |\mathbf{N}_l^{\mathcal{G}}|$
9:          **if** $(X_i \not\perp\!\!\!\perp X_j | \mathbf{X_k})_P$ for all $\mathbf{X_k} \subseteq \mathbf{N}_q^{\mathcal{G}}$ with $k \in \mathbf{k}$ **then**
10:             store (i, k, j) in $\mathbf{U}$
11:          **end if**
12:      **end if**
13: **end for**
14: **for all** triplets $(i, k, j) \in \mathbf{U}$ **do**
15:      orient $i \to k$ and $j \to k$ in $\mathcal{G}$
16: **end for**
17: repeatedly apply Meek's rules R1-4 to $\mathcal{G}$ until no rule is applicable

---

C Details regarding statistical evaluations

In this section, we provide basic details omitted from the body of the paper for the sake of brevity.

We develop here the notation and evaluation of the popular G^2 log-likelihood ratio test of independence for empirical estimation of conditional independence in P (Spirtes et al. 2000). For further details and examples, we refer to Neapolitan (2004) 10.3.1. Let $n[x_i]$ denote the number of counts for which random variable $X_i = x_i$ in n data samples $\mathcal{D}$, with the definition extending similarly to $n[x_i, x_j]$ and so on. For $\mathbf{k} \subseteq \mathbf{V}$, let $n[x_i, x_j, \mathbf{x_k}]$ denote the number of counts for which $X_i = x_i$, $X_j = x_j$, and $\mathbf{X_k}$ attains one of its $\prod_{k \in \mathbf{k}} r_k$ state configurations $\mathbf{x_k}$ in $\mathcal{D}$. Then the G^2 test statistic for testing $H_0 : (X_i \perp\!\!\!\perp X_j | X_k)_P$ is calculated as

$$G^2_{ij|\mathbf{k}} = 2 \sum_{x_i, x_j, \mathbf{x_k}} n[x_i, x_j, \mathbf{x_k}] \log \left(\frac{n[x_i, x_j, \mathbf{x_k}] \cdot n[\mathbf{x_k}]}{n[x_i, x_j] \cdot n[x_j, \mathbf{x_k}]} \right). \tag{23}$$

The equation can be applied to test marginal independence with $\mathbf{k} = \emptyset$. Under H_0, the G^2 statistic is asymptotically χ_f^2 distributed with

$$f = (r_i - 1)(r_j - 1) \prod_{k \in \mathbf{k}} r_k$$

degrees of freedom. Then for a chosen significance level α,

$$\Pr(\chi_f^2 > G_{ij|\mathbf{k}}^2) \leq \alpha \ \Rightarrow \ \text{reject } H_0 : (X_i \not\!\perp X_j | \mathbf{X_k})_P.$$

Note that the PC algorithm (Algorithm 1) operates in a somewhat backwards fashion where the initialized complete graph on $\mathbf{V}$ assumes all distinct pairs of variables dependent rather than independent. A node pair i and j is then disconnected if

$$\Pr(\chi_f^2 > G_{ij|\mathbf{k}}^2) > \alpha \ \Rightarrow \ \text{retain } H_0 : (X_i \perp X_j | \mathbf{X_k})_P$$

for some considered conditioning set $\mathbf{X_k}$.

The mutual information $I(X_i, X_j)$ serves as a similarity measure between discrete random variables X_i and X_j. It may be interpreted as the Kullback–Leibler divergence between the joint probability distribution and the product of the marginals, and it is empirically calculated for n data observations as

$$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \frac{n[x_i, x_j]}{n} \log \left(\frac{n[x_i, x_j]/n}{n[x_i]/n \cdot n[x_j]/n} \right),$$

where $n[x_i, x_j]$ is the counts of the instances of the n observations that satisfy $X_i = x_i$ and $X_j = x_j$, with corresponding definitions for $n[x_i]$ and $n[x_j]$. It is easy to see that $G_{ij}^2 = 2n \cdot \hat{I}(X_i, X_j)$.

Lastly, we include the penalized multinomial log-likelihood score with penalty parameter λ. Following the notation from Eq. (23), let $\boldsymbol{\pi}_i^{\mathcal{G}}$ represent one of the q_i unique state configurations of $\boldsymbol{\Pi}_i^{\mathcal{G}}$. Given the Bayesian network DAG structure $\mathcal{G}$ and empirical data $\mathcal{D}$ from P, the penalized score is computed as

$$
\begin{aligned}
\phi(\mathcal{G}, \mathcal{D}) &= \sum_{i, x_i, \boldsymbol{\pi}_i^{\mathcal{G}}} n[x_i, \boldsymbol{\pi}_i^{\mathcal{G}}] \log \left(\frac{n[x_i, \boldsymbol{\pi}_i^{\mathcal{G}}]}{n[\boldsymbol{\pi}_i^{\mathcal{G}}]} \right) - \lambda \sum_i (r_i - 1) q_i \\
&= \sum_{i=1}^{p} \left(\sum_{x_i, \boldsymbol{\pi}_i^{\mathcal{G}}} n[x_i, \boldsymbol{\pi}_i^{\mathcal{G}}] \log \left(\frac{n[x_i, \boldsymbol{\pi}_i^{\mathcal{G}}]}{n[\boldsymbol{\pi}_i^{\mathcal{G}}]} \right) - \lambda(r_i - 1) q_i \right) \quad (24) \\
&= \sum_{i=1}^{p} \phi(X_i, \boldsymbol{\Pi}_i^{\mathcal{G}}).
\end{aligned}
$$

In the penalty term, $\sum_{i=1}^{p} (r_i - 1) q_i$ encodes model complexity, encouraging sparsity in the structure of $\mathcal{G}$. In the BIC score, $\lambda = \frac{1}{2} \log(n)$ (Schwarz 1978).

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control, 19*(6), 716–723. https://doi.org/10.1109/tac.1974.1100705

Aliferis, C. F., Statnikov, A., Tsamardinos, I., Mani, S., & Koutsoukos, X. D. (2010). Local causal and Markov blanket induction for causal discovery and feature selection for classification Part I: Algorithms and empirical evaluation. *Journal of Machine Learning Research, 11*(7), 171–234. http://jmlr.org/papers/v11/aliferis10a.html.

Aliferis, C. F., Tsamardinos, I., & Statnikov, A. (2003). HITON: A novel Markov Blanket algorithm for optimal variable selection. In *AMIA annual Symposium proceedings* (Vol. 2003, p. 21–25). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1480117/.

Aragam, B., Gu, J., & Zhou, Q. (2019). Learning large-scale Bayesian networks with the sparsebn package. *Journal of Statistical Software, 91*(11), 1–38. https://doi.org/10.18637/jss.v091.i11

Buntine, W. (1991). Theory refinement on Bayesian networks. In B. D. D'Ambrosio, P. Smets & P. P. Bonissone (Eds.), *Uncertainty proceedings 1991* (p. 52–60). Morgan Kaufmann. https://doi.org/10.1016/B978-1-55860-203-8.50010-3.

Chickering, D. M. (2002a). Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research, 2*(Feb), 445–498. https://www.jmlr.org/papers/v2/chickering02a.html.

Chickering, D. M. (2002b). Optimal structure identification with greedy search. *Journal of Machine Learning Research, 3*(Nov), 507–554. https://jmlr.org/papers/v3/chickering02b.html.

Chickering, D. M., Heckerman, D., & Meek, C. (2004). Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research, 5*(Oct), 1287–1330.

Colombo, D., & Maathuis, M. H. (2014). Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research, 15*(116), 3921–3962. http://jmlr.org/papers/v15/colombo14a.html.

Cooper, G. F., & Herskovits, E. (1991). A Bayesian method for constructing Bayesian belief networks from databases. In B. D. D'Ambrosio, P. Smets & P. P. Bonissone (Eds.), *Uncertainty proceedings 1991* (pp. 86–94). Morgan Kaufmann. https://doi.org/10.1016/B978-1-55860-203-8.50015-2.

Cressie, N., & Read, T. R. C. (1989). Pearson's χ^{2} and the loglikelihood ratio statistic G^{2}: A comparative review. *International Statistical Review/Revue Internationale de Statistique, 57*(1), 19–43. https://doi.org/10.2307/1403582

Dor, D., & Tarsi, M. (1992). *A simple algorithm to construct a consistent extension of a partially oriented graph*. Technical Report R-185. Cognitive Systems Laboratory, UCLA.

Friedman, N., Nachman, I., & Peér, D. (1999). Learning Bayesian network structure from massive datasets: The "Sparse Candidate" algorithm. In *Proceedings of the fifteenth conference on uncertainty in artificial intelligence* (pp. 206–215). Morgan Kaufmann Publishers Inc. https://doi.org/10.5555/2073796.2073820.

Fu, F., & Zhou, Q. (2013). Learning sparse causal Gaussian networks with experimental intervention: Regularization and coordinate descent. *Journal of the American Statistical Association, 108*(501), 288–300. https://doi.org/10.1080/01621459.2012.754359

Gámez, J. A., Mateo, J. L., & Puerta, J. M. (2011). Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery, 22*(1–2), 106–148. https://doi.org/10.1007/s10618-010-0178-6

Gasse, M., Aussem, A., & Elghazel, H. (2014). A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications, 41*(15), 6755–6772. https://doi.org/10.1016/j.eswa.2014.04.032

Gu, J., Fu, F., & Zhou, Q. (2019). Penalized estimation of directed acyclic graphs from discrete data. *Statistics and Computing, 29*(1), 161–176. https://doi.org/10.1007/s11222-018-9801-y

Gu, J., & Zhou, Q. (2020). Learning big Gaussian Bayesian networks: Partition, estimation and fusion. *Journal of Machine Learning Research, 21*(158), 1–31. http://jmlr.org/papers/v21/19-318.html.

Hartigan, J. A. (1981). Consistency of single linkage for high-density clusters. *Journal of the American Statistical Association, 76*(374), 388–394. https://doi.org/10.1080/01621459.1981.10477658

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*(3), 197–243. https://doi.org/10.1023/A:1022623210503

Kalisch, M., & Bühlmann, P. (2007). Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research, 8*(22), 613–636. http://jmlr.org/papers/v8/kalisch07a.html.

Kalisch, M., Mächler, M., Colombo, D., Maathuis, M. H., & Bühlmann, P. (2012). Causal inference using graphical models with the R Package pcalg. *Journal of Statistical Software, 47*(11), 1–26. https://doi.org/10.18637/jss.v047.i11.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

Kraskov, A., Stögbauer, H., Andrzejak, R. G., & Grassberger, P. (2005). Hierarchical clustering using mutual information. *Europhysics Letters, 70*(2), 278–284. https://doi.org/10.1209/epl/i2004-10483-y

Le, T. D., Hoang, T., Li, J., Liu, L., Liu, H., & Hu, S. (2016). A fast PC algorithm for high dimensional causal discovery with multi-core PCs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 16*, 1483–1495. https://doi.org/10.1109/TCBB.2016.2591526

Liu, Z., Malone, B., & Yuan, C. (2012). Empirical evaluation of scoring functions for Bayesian network model selection. *BMC Bioinformatics, 13*, 1–16. https://doi.org/10.1186/1471-2105-13-S15-S14

Madsen, A. L., Jensen, F., Salmerón, A., Langseth, H., & Nielsen, T. D. (2017). A parallel algorithm for Bayesian network structure learning from large data sets. *Knowledge-Based Systems, 117*, 46–55. https://doi.org/10.1016/j.knosys.2016.07.031 Volume, Variety and Velocity in Data Science.

Margaritis, D. (2003). Learning Bayesian network model structure from data. Unpublished Doctoral Dissertation, Carnegie Mellon University School of Computer Science.

Meek, C. (1995). Causal inference and causal explanation with background knowledge. In *Proceedings of the eleventh conference on uncertainty in artificial intelligence* (pp. 403–410). Morgan Kaufmann Publishers, Inc. https://doi.org/10.5555/2074158.2074204.

Meek, C. (1997). Graphical Models: Selecting causal and statistical models. Unpublished Doctoral Dissertation, Carnegie Mellon University School of Computer Science.

Nandy, P., Hauser, A., & Maathuis, M. H. (2018). High-dimensional consistency in score-based and hybrid structure learning. *The Annals of Statistics, 46*(6A), 3151–3183. https://doi.org/10.1214/17-AOS1654

Neapolitan, R. E. (2004). *Learning Bayesian networks* (Vol. 38). Pearson Prentice Hall.

R Core Team. (2021). *R: A language and environment for statistical computing (computer software manual)*. https://www.Rproject.org/.

Ramsey, J. D. (2015). Scaling up greedy equivalence search for continuous variables. Computing Research Repository. arXiv:1507.07749.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica, 14*(5), 465–471. https://doi.org/10.1016/0005-1098(78)90005-5

Robinson, R. W. (1977). Counting unlabeled acyclic digraphs. In C. H. C. Little (Ed.), *Combinatorial mathematics V* (pp. 28–43). Springer. https://doi.org/10.1007/BFb0069178.

Russell, S., & Norvig, P. (2009). *Artificial intelligence: A modern approach* (3rd ed.). Pearson Prentice Hall.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., & Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single cell data. *Science, 308*(5721), 523–529. https://doi.org/10.1126/science.1105809

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*(2), 461–464. https://doi.org/10.1214/aos/1176344136

Scutari, M. (2010). Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software, 35*(3), 1–22. https://doi.org/10.18637/jss.v035.i03

Scutari, M. (2017). Bayesian network constraint-based structure learning algorithms: Parallel and optimized implementations in the bnlearn R package. *Journal of Statistical Software, 77*(2), 1–20. https://doi.org/10.18637/jss.v077.i02

Shao, J. (2003). *Mathematical statistics* (2nd ed., pp. 91–160). Springer.

Spirtes, P. (2010). Introduction to causal inference. *Journal of Machine Learning Research, 11*(54), 1643–1662. http://jmlr.org/papers/v11/spirtes10a.html.

Spirtes, P., & Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review, 9*(1), 62–72. https://doi.org/10.1177/089443939100900106

Spirtes, P., Glymour, C., Scheines, R., & Heckerman, D. (2000). *Causation, prediction, and search* (2nd ed.). MIT Press.

Tsamardinos, I., Aliferis, C. F., & Statnikov, A. R. (2003a). Algorithms for large scale Markov blanket discovery. In *FLAIRS conference* (pp. 376–380).

Tsamardinos, I., Aliferis, C. F., & Statnikov, A. (2003b). Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 673–678). Association for Computing Machinery. https://doi.org/10.1145/956750.956838.

Tsamardinos, I., Brown, L. E., & Aliferis, C. F. (2006). The max–min hill-climbing Bayesian network structure learning algorithm. *Machine Learning, 65*(1), 31–78. https://doi.org/10.1007/s10994-006-6889-7

Tsamardinos, I., Statnikov, A. R., Brown, L. E., & Aliferis, C. F. (2006). Generating realistic large Bayesian networks by tiling. In *FLAIRS conference* (pp. 592–597).

Verma, T., & Pearl, J. (1991). *Equivalence and synthesis of causal models*. UCLA Computer Science Department.

Wongchokprasitti, C. (2019). R-causal: R Wrapper for Tetrad Library. v1.2.1. https://github.com/bd2kcd/r-causal.

Yaramakala, S., & Margaritis, D. (2005). Speculative Markov blanket discovery for optimal feature selection. In *Fifth IEEE international conference on data mining (ICDM'05)*. https://doi.org/10.1109/ICDM.2005.134.

Zarebavani, B., Jafarinejad, F., Hashemi, M., & Salehkaleybar, S. (2020). cuPC: CUDA-based parallel PC algorithm for causal structure learning on GPU. *IEEE Transactions on Parallel and Distributed Systems, 31*(3), 530–542. https://doi.org/10.1109/TPDS.2019.2939126