EXACT DISTRIBUTED STRUCTURE-LEARNING FOR BAYESIAN NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning the structure of a Bayesian network is currently practical for only a limited number of variables. Existing distributed learning approaches approximate the true structure. We present an exact distributed structure-learning algorithm to find a P-map for a set of random variables. First, by using conditional independence, the variables are divided into sets $\mathcal{X}_1, \ldots, \mathcal{X}_I$ such that for each $\mathcal{X}_i$, the presence and absence of edges that are adjacent with any interior node (a node that is not in any other $\mathcal{X}_j, j \neq i$) can be correctly identified by learning the structure of $\mathcal{X}_i$ separately without using the information of the variables other than $\mathcal{X}_i$. Second, constraint or score-based structure learners are employed to learn the P-map of $\mathcal{X}_i$, in a decentralized way. Finally, the separately learned structures are appended by checking a conditional independence test on the boundary nodes (those that are in at least two $\mathcal{X}_i$'s). The result is proven to be a P-map. This approach allows for a significant reduction in computation time, and opens the door for structure learning for a "giant" number of variables.

1 INTRODUCTION

Bayesian networks constitute a primary subfield within the realm of probabilistic graphical models, which serve as powerful tools for data modeling. These networks leverage directed acyclic graphs (DAGs) to represent probabilistic relationships in datasets. The process of structure learning in Bayesian networks involves the derivation of a DAG from empirical data (Koller & Friedman, 2009). Two primary methodologies for learning the DAG from data are the constraint-based and score-based approaches (Kitson et al., 2023).

Constraint-based algorithms, such as the Peter-Clark (PC) algorithm (Spirtes et al., 2000), infer graphical structure from conditional independence relationships. This is achieved by employing conditional independence (CI) tests (Guo et al., 2020). Score-based algorithms aim to optimize a likelihood-based function, such as the Bayesian Information Criterion (BIC). Under the usual identifiability assumptions, observational conditional independences determine a Markov-equivalence class, represented by a completed partially directed acyclic graph (CPDAG), rather than a unique DAG (Koller & Friedman, 2009). Practical implementations of structure learners include the `bnstruct` package, which also handles missing data (Franzin et al., 2017).

Performing CI tests across all variables or optimizing the likelihood function over all potential graphs leads to computational challenges, often resulting in a computational explosion (Spirtes et al., 2000). This problem represents a significant challenge and limitation, particularly when dealing with a substantial number of variables (Peters et al., 2017; Ramsey et al., 2017). Several techniques have been developed to reduce runtime, for example by combining CI tests for complementary conditioning sets (Giudice et al., 2022), limiting the conditioning set in the CI tests, (Sondhi & Shojaie, 2019), finding an order on the variables under additional variance assumptions (Chen et al., 2019; Gao et al., 2020), and parallelizing CI tests (Zarebavani et al., 2020; Le et al., 2019). ParaLiNGAM instead parallelizes the root-selection computations of DirectLiNGAM for linear non-Gaussian models (Shahbazinia et al., 2023).

Nevertheless, exact score optimization remains computationally demanding in general. Dynamic programming and A* search avoid explicit enumeration of every DAG but still face exponential search spaces in the worst case (Singh & Moore, 2005; Yuan et al., 2011). Heuristic score search can handle thousands of variables (Scanagatta et al., 2015); exact methods with structured local models have

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

also been developed (Talvitie et al., 2019). Their applicability depends on the scoring model, graph restrictions, and available resources, rather than a universal small-network limit. Constraint-based algorithms, such as PC, start or interact with a fully connected network, which has $\mathcal{O}(n^2)$ edges for n variables. This initial representation is more compact, but the number of CI tests can grow exponentially with the sizes of the adjacency sets searched (Spirtes et al., 2000).

Reducing the structure-learning problem to several sub-problems that can be learned separately can be the key to solve this issue. The partition-estimation-fusion framework of Gu & Zhou (2020) partitions variables into clusters, learns subgraphs, and fuses the results for Gaussian Bayesian networks. Its local learners may return DAGs or CPDAGs.

The partitioning of the variables is the main step of this approach. Xie et al. (2006) use conditional independences to localize structure learning; prior knowledge can facilitate their decomposition but is not stated as a requirement. The separation-and-reunion method of Liu et al. (2017) combines CI-based separation with score-based learning. Zhang et al. (2022) recursively decompose the variables using low-order CI tests. A different line of work jointly infers edges and block structure in Gaussian graphical models (van den Boom et al., 2023). These approaches differ in their decomposition and reconstruction rules; our implementation makes the separator-size cap explicit and combines overlapping covers with tensorized CI evaluation.

We develop an exact distributed structure-learning algorithm that obtains the true P-map for a given set of random variables in three steps. First, the algorithm performs a *reduction* on the set of variables, by dividing them into sets $\mathcal{X}_1, \ldots, \mathcal{X}_I$. Each set $\mathcal{X}_i$ has a boundary $\mathrm{bd}(\mathcal{X}_i)$ that is the subset of nodes shared with other $\mathcal{X}_j$, i.e., $\cup_{j \neq i} \mathcal{X}_i \cap \mathcal{X}_j$, and an interior $\mathcal{X}_i^o$ which is the remainder, i.e., $\mathcal{X}_i \setminus \mathrm{bd}(\mathcal{X}_i)$. The reduction is such that the P-map confined to each set $\mathcal{X}_i$ is a *conditional P-map* for the marginal distribution of the variables in $\mathcal{X}_i$; namely, the presence and absence of all edges that are adjacent with the interior nodes of $\mathcal{X}_i$ are correctly learned by performing a structure-learner to find the P-map of $\mathcal{X}_i$. Roughly speaking, the "interior edges" of each set $\mathcal{X}_i$ can be learned separately, without the information about the nodes in the other $\mathcal{X}_j$'s. This naturally leads to the second step, where separate structure-learners, either constraint or score-based, are deployed to learn the local P-map structure of every $\mathcal{X}_i$. Finally, the local P-maps are concatenated to obtain the global P-map by performing a distributed PC-like algorithm on all boundary nodes. We prove that the resulting DAG is a P-map.

2 Background

Consider a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ with joint probability distribution P. Let $\mathcal{I}(P)$ denote the set of all conditional independencies implied by the distribution P, i.e., $\mathcal{I}(P) = \{(\mathcal{X}_1 \perp \mathcal{X}_2 \mid \mathcal{X}_3) : \mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3 \subseteq \mathcal{X}\}$. Let $\mathcal{G}$ be a DAG with node set $\mathcal{X}$. The DAG induces conditional independencies between the nodes using the notion of d-separation defined below. A *collider* in $\mathcal{G}$ is a subgraph consisting of three nodes $X_1 \to X_2 \leftarrow X_3$, where X_1 and X_3 are linked to X_2. The collider is an *immorality* if the ending nodes X_1 and X_3 are not adjacent (connected). Three nodes are a *non-collider* if they do not form a collider. Node Y is an *ancestor* of node X if there is a directed path (of length possibly zero) from Y to X.

Definition 2.1 (d-separation)**.** (Koller & Friedman, 2009) Consider the DAG $\mathcal{G}$ with node set $\mathcal{X}$. A trail (path) $\mathcal{T}$ between two nodes X_1 and X_2 in $\mathcal{X}$ is *active* relative to a set of nodes $\mathcal{Z}$ if *(i)* every non-collider on $\mathcal{T}$ is not a member of $\mathcal{Z}$, and *(ii)* every collider on $\mathcal{T}$ is an ancestor of some member of $\mathcal{Z}$. Otherwise, the trail is said to be *blocked by* $\mathcal{Z}$. The node subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ are *d-separated* given subset $\mathcal{Z}$, denoted $\text{d-sep}_{\mathcal{G}}(\mathcal{X}_1, \mathcal{X}_2 \mid \mathcal{Z})$, if there is no active trail between any node $X_1 \in \mathcal{X}_1$ and any node $X_2 \in \mathcal{X}_2$ given $\mathcal{Z}$.

The set of all d-separations in $\mathcal{G}$ is denoted by $\mathcal{I}(\mathcal{G})$. We assume that for the distribution P, there exists a DAG $\mathcal{G}$ that satisfies both of the following well-known conditions: *(i)* Markovness, that is, $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P)$, and *(ii)* faithfulness, that is $\mathcal{I}(P) \subseteq \mathcal{I}(\mathcal{G})$. This results in $\mathcal{I}(P) = \mathcal{I}(\mathcal{G})$; namely, all conditional independencies in P are captured by the d-separations in $\mathcal{G}$ and vice versa. DAG $\mathcal{G}$ is called a *P-map (perfect map)* for P. The problem is to find P-map $\mathcal{G}$ for distribution P. This problem is known as *structure learning*. There can be more than one P-map for a distribution P, e.g., two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ where $\mathcal{I}(\mathcal{G}_1) = \mathcal{I}(\mathcal{G}_2) = \mathcal{I}(P)$. P-maps of the same distribution have the same skeleton and immoralities (Koller & Friedman, 2009). Consequently, the set of all P-maps for a distribution P is represented by a PDAG that is a graph over nodes $\mathcal{X}$ where two nodes are adjacent, if they are adjacent in all of the P-maps and the connected edge is directed if all of the P-maps have

the same direction, otherwise the edge is undirected. This PDAG is called the *P-map class PDAG for* P or a *complete PDAG (CPDAG)*. The structure learning problem is often reduced to finding the P-map class PDAG for P.

Problem 1 (Structure learning). Consider the set of random variables $\mathcal{X}$ with distribution P that admits a P-map. Find the P-map class PDAG for P.

Several *constraint-based algorithms*, such as PC (Spirtes et al., 2000), and *score-based algorithms* with a *consistent score*, such as BIC, that perform an exhaustive search over the DAG space, are shown to solve Problem 1 (Koller & Friedman, 2009). We call an algorithm that solves Problem 1, a *P-map learner*. Problem 1 is NP-hard and cannot be practically solved for a large number of variables n (Koller & Friedman, 2009).

3 Distributed structure learning

3.1 The idea

Our goal is to solve Problem 1 in a distributed manner as explained intuitively below.

Example 1. *The DAG in Figure 1.a, denoted by $\mathcal{G}$, is a P-map for the joint distribution of random variables $X_1, \ldots, X_5$. Instead of learning the whole DAG at once, one can learn separately the P-map class PDAG of each of the sub-DAGs for $\mathcal{X}_1 = \{X_1, X_2, X_3\}$, $\mathcal{X}_2 = \{X_4, X_3\}$, and $\mathcal{X}_3 = \{X_5, X_3\}$ (Figure 1.b), and then concatenate and orient them to obtain P-map $\mathcal{G}$. The reason is that the three subsets are d-separated, and hence, independent, from one another given their shared variable X_3:*

$$X_1, X_2 \perp X_4 \mid X_3, \;\; X_1, X_2 \perp X_5 \mid X_3, \;\; X_4 \perp X_5 \mid X_3.$$

This ensures two points. First, X_1 and X_2 are not connected by a path outside of $\{X_1, X_2, X_3\}$; that is, all of their dependencies are captured by this set. Hence, a P-map learner can correctly learn the structure of these nodes without using the information from the other nodes X_4 and X_5. Second, when concatenating the graphs, no additional link between the subsets are needed.

This idea does not apply to the partitioning subsets $\{X_1, X_4\}$ and $\{X_2, X_3, X_5\}$, because X_1 and X_4 do depend on $\{X_2, X_3, X_5\}$:

$$\{X_1, X_4\} \not\perp \{X_2, X_3, X_5\}.$$

Nodes X_1 and X_4 are related by a path outside of $\{X_1, X_4\}$, e.g., $X_1 \to X_3 \to X_4$. Hence, when learning the structure of $\{X_1, X_4\}$, the P-map learner will incorrectly make X_1 and X_4 adjacent, because they are dependent. Similarly, subsets $\{X_1, X_3, X_4\}$ and $\{X_2, X_3, X_5\}$ would not work either. Because although the P-map of each subset can be learned correctly, the concatenation would require an additional link between X_1 and X_2 to recover $\mathcal{G}$.

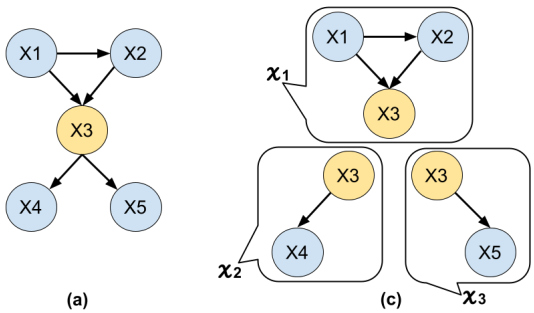

Figure 1: a) P-map $\mathcal{G}$ for $\{X_1, \ldots, X_5\}$. b) Three subsets that can be learned separately.

We formalize the reduction approach illustrated in Example 1. For any subset $\mathcal{Y} \subseteq \mathcal{X}$, let $P[\mathcal{Y}]$ denote the marginal distribution over $\mathcal{Y}$, and let $\mathcal{G}[\mathcal{Y}]$ be the subgraph of $\mathcal{G}$ induced by $\mathcal{Y}$. Define a *cover* of $\mathcal{X}$ as a family of distinct nonempty subsets $\mathcal{X}_1, \ldots, \mathcal{X}_I \subseteq \mathcal{X}$ where $I \geq 1$ such that $\cup_{i=1}^{I} \mathcal{X}_i = \mathcal{X}$. Define the *boundary of* $\mathcal{X}_i$, $i = 1, \ldots, I$, by $\mathrm{bd}(\mathcal{X}_i) = \mathcal{X}_i \cap (\cup_{j \neq i} \mathcal{X}_j)$, and the *interior of* $\mathcal{X}_i$ by $\mathcal{X}_i^o = \mathcal{X}_i \setminus \mathrm{bd}(\mathcal{X}_i)$. The union of all boundaries is called the *separator* of this cover,

denoted $\mathcal{W} = \cup_i \mathrm{bd}(\mathcal{X}_i)$. Correspondingly, the cover $\{\mathcal{X}_1, \ldots, \mathcal{X}_I\}$ for $\mathcal{X}$ is referred to as the *cover separated by* $\mathcal{W}$. In Example 1, $\mathcal{W} = \{X_3\}$.

The idea in Example 1 is to divide the graph possibly overlapping subsets $\mathcal{X}_1, \ldots, \mathcal{X}_I$, $I > 0$, such that each subgraph $\mathcal{G}[\mathcal{X}_i]$ can be learned independently, without information from other subsets $\mathcal{X}_j$ for $j \neq i$. This is the same as requiring each $\mathcal{G}[\mathcal{X}_i]$ to be a P-map for $P[\mathcal{X}_i]$. Then the union of the learned graphs for the $\mathcal{X}_i$'s should reconstruct the class PDAG for P. This is not possible though when the graph is 2-connected as explained in the next example.

Example 2. *Consider the $\mathcal{G}$ in Figure 2.a and suppose that it is a P-map for $\mathcal{X} = \{X_1, \ldots, X_{13}\}$. The edge between X_1 and X_{13} is directed $X_{13} \to X_1$, leaving the single collider $X_1 \to X_2 \leftarrow X_3$. In $\mathcal{G}$, every pair of nodes are connected by two paths. Thus, to break the graph into separately learnable cover elements that have no inter-edges, there should be at least two boundary nodes, say X_9 and X_{12}, resulting in $\mathcal{X}_1$ and $\mathcal{X}_2$. However, then $\mathcal{G}[\mathcal{X}_1]$ in Figure 2.b is not a P-map for $P[\mathcal{X}_1]$ as that would require X_9 and X_{12} to be adjacent. Nevertheless, observing the boundary nodes X_9 and*

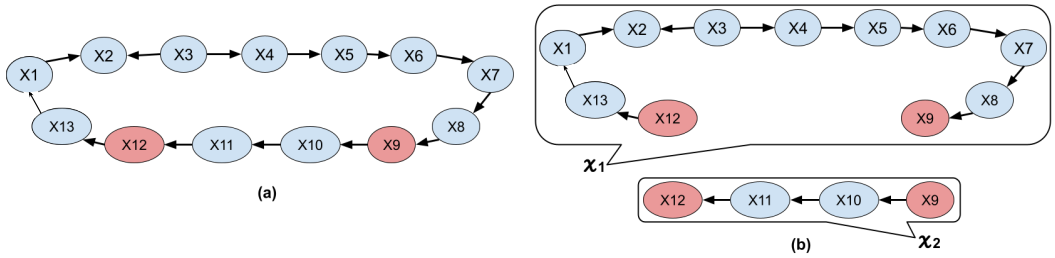

Figure 2: (a) P-map $\mathcal{G}$ for variables $X_1, \ldots, X_{13}$. (b) The cover consisting of $\mathcal{X}_1 = \{X_9, X_{10}, X_{11}, X_{12}\}$ and $\mathcal{X}_2 = \{X_{12}, X_{13}, X_1, \ldots, X_9\}$, separated by $\mathcal{W} = \{X_9, X_{12}\}$.

X_{12} blocks every path connecting any of the interior nodes of $\mathcal{X}_2$, i.e., $X_{13}, X_1, \ldots, X_8$, to any of the interior nodes of $\mathcal{X}_1$, i.e., X_{10} and X_{11}. Namely, $\mathcal{X}_1^o$ is d-separated of $\mathcal{X}_2^o$ given $\mathrm{bd}(\mathcal{X}_1^o)$. Thus, for every node in $\mathcal{X}_1$ that is not adjacent to an interior node in $\mathcal{X}_1$, there is a corresponding separating set in $\mathcal{X}_1$ that makes them d-separated. For example, X_{12} is not adjacent to X_1, and the two are d-separated given X_{13}. Since $\mathcal{G}$ is a P-map for $\mathcal{X}$, this means that the subgraph consisting of $\mathcal{X}_1$ and all the edges incident to the interior of $\mathcal{X}_1$ can be learned independently.

Define the *overlap-consensus union* of graphs $\mathcal{G}_i = (\mathcal{X}_i, \mathcal{E}_i)$ as the graph with node set $\cup_{i=1}^I \mathcal{X}_i$, in which two vertices are adjacent if and only if they are so in every $\mathcal{G}_i$ containing both vertices.

Definition 3.1 (Consensus P-map reduction)**.** Consider random variables $\mathcal{X}$ with distribution P that admits a CPDAG $\mathcal{G}$. Let $d \geq 1$ be an integer. A cover $\{\mathcal{X}_1, \ldots, \mathcal{X}_I\}$, $I > 1$, of $\mathcal{X}$, where $|\mathcal{X}_i| \leq d$ for all $i = 1, \ldots, I$, is a *(capped-d) consensus P-map reduction of P* if $\mathcal{G}$ is obtained from the overlap-consensus union of the CPDAG's of the $P[\mathcal{X}_i]$'s by successively orienting the remaining undirected edges without introducing new immoralities or directed cycles.

The idea behind the overlap-consensus is as follows. Since $\mathcal{G}$ is a CPDAG for P, any conditional independence detected between two nodes in any of the cover elements implies that they are non-adjacent in $\mathcal{G}$. Hence, when taking the union of the cover elements, for two boundary nodes to be adjacent, they should be so in every cover element containing the two, which is implied by the overlap-consensus union. Also, in the context of constraint based algorithms, this means that once an edge is removed during the P-map learning of any cover element, it will be absent in the true P-map as well.

How to find a consensus P-map reduction? Motivated by the above examples, the idea is to divide the nodes into subsets that are d-separated given their common nodes, i.e., every interior $\mathcal{X}_i^o$ is d-separated of every other interior $\mathcal{X}_j^o$ given its boundary $\mathrm{bd}(\mathcal{X}_i)$. We start by defining a separator $\mathcal{W}$ and a partition of the set $\mathcal{X} \setminus \mathcal{W}$ into some subsets $\mathcal{C}_1, \ldots, \mathcal{C}_I$ that are pairwise independent given $\mathcal{W}$.

Definition 3.2 (Separated-by cover)**.** Consider random variables $\mathcal{X}$ with distribution P and a subset $\mathcal{W} \subset \mathcal{X}$. A *cover of $\mathcal{X}$ separated by $\mathcal{W}$* is the collection of sets $\{\mathcal{W} \cup \mathcal{C}_i\}_{i=1}^I$, $I \geq 2$, such that *(i)*

Algorithm 1: Direct adaptive cover search

Input: Variables $\mathcal{X}$, CI tester, member-size-dependent limit q

$\mathcal{U} \leftarrow \{\mathcal{X}\}$; $w \leftarrow 0$; initialize exhausted-pair cache;
while $w \leq 2$ **do**
  Visit members U in decreasing size, skipping exhausted (U, w) and $w > q(|U|)$;
  For each eligible U, enumerate $W \subset U$ with $|W| = w$ deterministically;
  Find the connected components of $H(U, W)$ using batched CI queries;
  if *a candidate gives multiple components* $C_1, \ldots, C_r$ **then**
    Replace U with $\{C_j \cup W\}_{j=1}^{r}$; retain cross-component separators;
    Restart at $w \leftarrow 0$;
  else
    Mark exhausted member–size pairs; $w \leftarrow w + 1$;

$\{\mathcal{C}_1, \ldots, \mathcal{C}_I, \mathcal{W}\}$ is a partition for $\mathcal{X}$, *(ii)* $\mathcal{C}_i \perp \mathcal{C}_j \mid \mathcal{W}$ for all distinct $i, j = 1, \ldots, I$, *(iii)* and I is maximal subject to *(i)* and *(ii)*.

One can iteratively apply separators to the elements of a cover until they no longer decompose. How to find the cover separated by $\mathcal{W}$? Consider an order for variables $\mathcal{X}$, represented by vector $\boldsymbol{X} = [X_1, \ldots, X_n]^\top$. Vector $\boldsymbol{X}_\mathcal{W}$ is defined as $\boldsymbol{X}$ where the elements of $\mathcal{W}$ are removed and let $[\boldsymbol{X}_\mathcal{W}]_i$ be the ith entry of $\boldsymbol{X}_\mathcal{W}$. For a subset $\mathcal{W} \subset \mathcal{X}$, define the symmetric $(n - |\mathcal{W}|) \times (n - |\mathcal{W}|)$ *dependency matrix* $D_\mathcal{W}$ by $D_\mathcal{W}(i, j) = 1$ if $[\boldsymbol{X}_\mathcal{W}]_i \not\perp [\boldsymbol{X}_\mathcal{W}]_j \mid \mathcal{W}$ and otherwise $D_\mathcal{W}(i, j) = 0$ for all $i, j \in \{1, \ldots, n - |\mathcal{W}|\}$, with $D_\mathcal{W}(i, i) = 1$ for all i. By using a permutation matrix V, we have $\bar{D}_\mathcal{W} = V D_\mathcal{W} V^{-1}$ where $\bar{D}_\mathcal{W}$ is the block diagonal form of $D_\mathcal{W}$. Then each group of entries of the transformed vector $\bar{\boldsymbol{X}}_\mathcal{W} = V \boldsymbol{X}_\mathcal{W}$ that correspond to a block of $\bar{D}_\mathcal{W}$ constitutes one of the desired partitions, which combined with $\mathcal{W}$ form an element of the P-map reduction (see Example 1-revisited in the appendix). Equivalently, one can view $D_\mathcal{W}$ as the adjacency matrix of an undirected graph on $\mathcal{X} \setminus \mathcal{W}$. Then the blocks of $P D_\mathcal{W} P^\top$ are exactly the connected components of this graph. Hence, once $D_\mathcal{W}$ is computed, a suitable permutation can be obtained in $O((n - |\mathcal{W}|)^2)$ time when $D_\mathcal{W}$ is represented densely (Cormen et al. (2001); or $O(k + m)$ for a sparse representation).

3.2 The algorithms

The distributed procedure has the same three stages as the original construction: cover discovery, separate local structure learning, and reconciliation of the local graphs. The earlier search variants and their full explanations are retained in Appendix C. For the new experiments we use a single, direct separator search, avoiding the need to choose between these variants.

The original search tradeoff is important. When searching for separators of size w in a member U of cardinality u, an exhaustive level can check all $\binom{u}{w}$ subsets, followed by all $\binom{u}{w+1}$ subsets at the next level. The alternative in Algorithm 5 retains a discovered separator and searches for single variables to add to it, then pairs, and so on. This changes the candidate counts to $\binom{u-w}{1}$, $\binom{u-w}{2}$, and subsequent extensions. However, the conditioning sets then grow because previous separator variables remain conditioned on. Once a separator of size w has been found, the next reduction may already require at least $w + 1$ conditioning variables. In contrast, restarting the separator search inside a newly formed member permits low-order tests again. These considerations explain why reducing the member size and limiting the separator size are distinct design choices. The current implementation retains the latter low-order approach, with memoization to avoid repeating exhausted member–size searches; it does not silently substitute the historical boundary-accumulating variant.

For a current member U and separator $W \subset U$, let $H(U, W)$ be the undirected graph on $U \setminus W$ with an edge $x - y$ precisely when $x \not\perp y \mid W$. Its components C_j produce the cover members $C_j \cup W$. The maximum separator size depends on the current member: zero for $|U| < 10$, one for $10 \leq |U| < 30$, and two otherwise. Thus even the smallest members receive a marginal split attempt. The search does not skip every member below a fixed 32-variable threshold.

The implementation memoizes exhausted searches and uses a frontier traversal to find components. Cover search is sequential, with tensorized CI batches; this is distinct from parallel enumeration of separators. After cover construction, local PC skeleton learners run in parallel. Their edges are

unioned, and recorded local CI witnesses remove edges from the union. The evaluated implementation then performs global PC cleanup over all variables, retaining local and cross-cover separating sets, followed by orientation. Unlike an exclusively boundary-based construction, global cleanup can use a separator outside an individual cover member. If there is no split, the procedure falls back to the corresponding full-variable learner.

Tensorized conditional-independence tests. For discrete variables X_i, X_j and conditioning stratum s, let O_{abs} denote the observed contingency count and $E_{abs} = O_{a+s}O_{+bs}/O_{++s}$ its expected count. Both scalar and tensor implementations evaluate

$$\chi^2_{ij|W} = \sum_s \sum_{a,b:E_{abs}>0} (O_{abs} - E_{abs})^2/E_{abs}. \tag{1}$$

Tensorization batches histogram construction and reductions; it does not change the statistical test. Stratum encodings and marginal counts are reused across pairs. The full tensor/counting derivation from the manuscript is preserved in Appendix E.1. Local learning is parallelized across cover members, and eligible global CI batches use the same worker allocation.

3.3 Supporting theory

The original theoretical development and its proof arguments are retained in the appendix. The following separate guarantee states the assumptions needed for the global-cleanup implementation used in the new experiments; it does not rely on the stronger boundary-only claim.

Lemma 3.3 (Preservation of true adjacencies)**.** *For a distribution faithful to a DAG G, Algorithm 1 leaves every true adjacency together in at least one cover member. Oracle local PC, union of local skeletons, and removal by recorded CI witnesses preserve every such adjacency.*

Theorem 3.4 (Oracle global reconstruction)**.** *Under causal sufficiency, faithfulness, and exact CI answers, the cover construction and local PC stage followed by unrestricted global PC cleanup, testing neighbor subsets from both endpoints, and sound and complete CPDAG orientation recover the CPDAG of G.*

Proof. A direct edge cannot be blocked by any conditioning set excluding its endpoints. If both endpoints lie outside a candidate separator W, they therefore lie in the same component of $H(U, W)$. If either lies in W, it is included in the child containing the other endpoint. Every split thus preserves a common cover member for each true adjacency. A local oracle test cannot remove that adjacency, so the merged graph is a supergraph of the true skeleton.

For a nonadjacent pair x, y, choose the endpoint x for which y is a nondescendant; such a choice exists because G is acyclic. Since y is not a parent of x, the local Markov property gives $x \perp y \mid \mathrm{Pa}_G(x)$. Every parent remains a current neighbor of x, because all true adjacencies are retained. Unrestricted global cleanup therefore encounters a separating set for each remaining nonedge. This establishes exact skeleton recovery without assuming that every local marginal is faithful to the induced subgraph.

For an unshielded triple $x - z - y$, a valid separating set must contain z when z is a noncollider, since otherwise the length-two path remains active. It cannot contain z when z is a collider, since conditioning on z activates that path. Retained separating sets therefore identify precisely the unshielded colliders. The skeleton and these colliders determine the Markov-equivalence class, and sound and complete orientation rules produce its CPDAG. Further details appear in Appendix A. □

The scalar and tensor Pearson calculations are algebraically equivalent given the same counts, degrees of freedom, and threshold. Floating-point differences near the threshold remain possible. Oracle correctness does not guarantee finite-sample accuracy, a runtime improvement, or correctness of local Tabu substitutions. The latter is an empirical ablation, not an instance of the theorem.

4 EXPERIMENTS

4.1 BENCHMARK DESIGN AND IMPLEMENTATION

Following the organization of the submitted experiments, we distinguish the benchmark setup, implementation, comparisons, and results. All experimental tables below report the new controlled rerun; historical measurements are not substituted for uncompleted cells.

The controlled grid contains 12 algorithms on 16 reference networks at $n \in \{100, 1000, 10000\}$, with sampling seed 42. The networks are CANCER, EARTHQUAKE, SURVEY, ASIA, CHILD, INSURANCE, WATER, MILDEW, ALARM, HEPAR2, WIN95PTS, PATHFINDER, ANDES, DIABETES, PIGS, and LINK. All algorithms within a cell use identical sampled data; file hashes are recorded. Comparators are PC, PC-stable, Parallel PC, Tensor PC, Distributed Tensor PC (D-T-PC), Distributed Scalar PC (D-S-PC), Distributed Tensor Tabu (D-T-Tabu), MMPC, Parallel MMPC, hill climbing, Tabu, and GES. Simple parallel PC/MMPC, tensor-only PC, scalar distributed PC, and local-Tabu substitutions explicitly separate the proposed components.

System configuration. Two Linux compute nodes each provide two 64-core Intel Xeon 6760P processors, 128 physical cores and approximately 1000 GiB of scheduler-visible RAM. Parallel methods receive 16 CPU cores per job and serial methods one; a dataset is assigned to a fixed node for paired comparisons. New jobs request 5 GiB, although some earlier baseline jobs requested more, so a universal measured 5-GiB bound is not claimed. No GPU is used. The environment uses Python 3.12.5, NumPy 2.4.2, SciPy 1.17.1, pandas 2.3.3, and pgmpy 0.1.25. Numerical-library threads are limited to one per worker.

Matched tests and metrics. All CI-based methods use Pearson chi-square with $\alpha = 0.05$, no continuity correction, and observed-stratum degrees of freedom. Conditional zero-degree-of-freedom tests are treated as dependent; the marginal zero-degree-of-freedom case is treated as independent. Score-based comparators use BIC. Runtime includes cover search, local learning, merging, cleanup, and orientation, but excludes data loading and metric calculation. We report raw CPDAG SHD and adjacency F1. MMPC returns a skeleton, so its SHD also includes missing reference orientations; F1 is the direct adjacency comparison.

Distributed methods have six-hour caps for networks of at most 32 variables and twelve-hour caps otherwise; baselines have twelve-hour caps. Verified timeouts are reported as bounds, not successful completion times. All 67 timeouts currently recorded reached twelve hours and are labeled $> 12\,\mathrm{h}$. NR denotes no verified result and is not recoded as a timeout. Invalid graphs and execution errors remain separate, and no accuracy is imputed for them.

4.2 RESULTS

The snapshot collected on 26 September 2026 at 11:05 UTC contains 339 valid completed cells, 67 timeouts, 12 invalid outputs, three errors, and 155 cells without a result, out of 576 planned cells. Table 1 reports all 12 algorithms on all 16 networks at $n = 100$. The runtime tables for $n = 1000$ and $n = 10000$, and SHD/F1 tables at all three sample sizes, are in Appendix F. Table 2 summarizes pairs for which both algorithms completed validly; each row can cover a different subset of network–sample combinations.

Tensorization helps substantially within the distributed pipeline: D-T-PC is faster on 33 of 34 completed pairs against D-S-PC, with about a 6.1-fold geometric-mean speedup and identical SHD and F1 on all 34. However, decomposition currently does not improve tensor-only PC: D-T-PC is slower in all 21 matched completed pairs, with a runtime ratio of 16.1. Against ordinary PC, it is faster in nine of 19 pairs but slower overall by the geometric-mean ratio. These results do not support a blanket superiority claim.

Cover telemetry explains the gap. Of 39 completed D-T-PC runs, 29 split and ten do not. On PIGS at $n = 100$, total time is approximately 25,013 seconds, including 25,005 seconds of unsuccessful cover search, versus 6.32 seconds for Tensor PC. On ANDES at $n = 100$, the cover does split, but search consumes approximately 23,251 of 23,252 seconds. Useful balanced decomposition, rather than merely obtaining a split, is therefore essential. Three D-T-PC runs failed out of memory: PIGS

Table 1: Runtime (seconds for completed runs), $n = 100$ samples. Hour bounds denote confirmed timeouts; NR denotes no verified result. INV and ERR are invalid outputs and execution errors.

Algorithm	CANCER	EARTH	SURVEY	ASIA	CHILD	INSUR.	WATER	MILDEW
PC	0.0143	0.033	0.0226	0.0648	0.367	0.423	0.238	NR
PC-stable	0.0146	0.0349	0.0236	0.0768	0.412	0.453	0.215	NR
Parallel PC	0.206	NR	0.216	NR	0.877	NR	0.521	NR
Tensor PC	0.00719	NR	0.0124	NR	0.275	NR	0.238	NR
D-T-PC	0.136	0.163	0.156	0.194	0.531	0.794	0.406	1.43
D-S-PC	0.11	0.431	0.379	0.555	2.58	4.06	1.36	26.4
D-T-Tabu	0.148	0.775	0.4	3.42	5.05	79.9	4.6	4.91
MMPC	0.0172	0.0515	0.0426	0.128	0.648	1.07	0.661	2.81
Parallel MMPC	0.0821	0.124	0.165	0.231	0.465	0.504	0.338	NR
HC	0.148	0.316	0.422	1.45	23.4	53.1	66.3	39.9
Tabu	0.892	0.827	1.45	4.89	34.6	75.9	98.8	141
GES	0.181	0.457	0.327	1.62	20.6	56.2	148	39.3

Algorithm	ALARM	HEPAR	WIN95	PATH	ANDES	DIAB.	PIGS	LINK
PC	NR	NR	NR	NR	NR	137	NR	$> 12\,\mathrm{h}$
PC-stable	NR	NR	NR	NR	NR	223	NR	$> 12\,\mathrm{h}$
Parallel PC	0.872	NR	1.05	NR	5.61	NR	39.6	NR
Tensor PC	0.483	NR	0.481	NR	1.45	NR	6.32	NR
D-T-PC	12.7	30.4	34.3	189	23252	$> 12\,\mathrm{h}$	25013	NR
D-S-PC	117	301	283	2812	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR
D-T-Tabu	157	INV	921	INV	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR
MMPC	1.99	3.82	6.95	8099	53.9	NR	261	NR
Parallel MMPC	NR	NR	NR	7965	NR	NR	NR	$> 12\,\mathrm{h}$
HC	149	1002	1363	3503	$> 12\,\mathrm{h}$	NR	$> 12\,\mathrm{h}$	NR
Tabu	191	1156	1518	3707	$> 12\,\mathrm{h}$	NR	$> 12\,\mathrm{h}$	NR
GES	141	1554	1627	4761	$> 12\,\mathrm{h}$	NR	$> 12\,\mathrm{h}$	NR

Table 2: D-T-PC versus matched completed baselines. Runtime ratio is the geometric mean of D-T-PC time divided by baseline time; below one is faster.

Baseline	Pairs	D-T-PC faster	Time ratio	F1 better/tied/worse
PC	19	9	1.32	3/14/2
Parallel PC	21	12	3.44	1/15/5
Tensor PC	21	0	16.1	1/15/5
D-S-PC	34	33	0.164	0/34/0

at $n = 1000$, and PATHFINDER and DIABETES at $n = 10000$. Nine of the 12 invalid outputs are from D-T-Tabu.

Statistical interpretation. Exploratory two-sided Wilcoxon checks for the new runs are reported with the new tables, using paired log runtimes and raw SHD/F1 on valid completed pairs. Values are unadjusted. One seed, incomplete coverage, and related sample sizes from the same network limit their interpretation. We therefore emphasize matched effect sizes and preserve all unavailable and unsuccessful outcomes.

5 Conclusion

We developed a distributed approach for structure learning applicable to both constraint-based and score-based algorithms. The main concept is to identify a cover set for the set of variables using conditional independence (CI) tests. Two key parameters, the upper bound of the cardinality of cover elements d and the number of conditioning variables W play crucial roles in determining an appropriate cover set. Reducing the value of d while increasing W can decrease the cardinality of the greatest cover element and increase the number of cover elements. However, this adjustment

may lead to an increase in the number of CI tests, which in turn could raise the runtime of the cover-finding algorithms. Algorithms 3 and 5 are executed in parallel across multiple CPUs. Consequently, increasing the number of CPUs can help reduce runtime. Thus, it is essential to select the values of d, W, and the number of CPUs carefully to ensure that the cardinality of the greatest cover element remains small, enabling the runtime to be less than that of the standard version of the structure learning algorithm. We developed a distributed approach to Bayesian-network structure learning based on finding overlapping cover elements, learning local structures, and reconciling them. The original explanations and search variants clarify the tradeoff between cover-member size and separator size: richer separator searches may yield smaller subproblems but can also dominate runtime. The new direct size-adaptive search makes this choice explicit and retains the possibility of marginal splitting in small networks.

The new controlled experiments distinguish decomposition from tensorized CI testing and ordinary parallelization. Tensorization substantially improves the scalar distributed implementation without changing its SHD or F1 on matched completed pairs. The present decomposition strategy is not consistently faster than ordinary PC and is slower than tensor-only PC on available completed pairs. Improving separator-search cost and split quality remains necessary. Oracle exactness is a structural guarantee under stated assumptions, not a claim of universal finite-sample accuracy or speed.

AI ASSISTANCE STATEMENT

AI assistance was used for manuscript editing, code inspection, table preparation, and drafting proof arguments. Responsibility for the final scientific claims, correctness, and submission remains with the authors.

REFERENCES

Wenyu Chen, Mathias Drton, and Y. Samuel Wang. On causal discovery with an equal-variance assumption. *Biometrika*, 106(4):973–980, 2019. doi: 10.1093/biomet/asz049.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2nd edition, 2001.

Alberto Franzin, Francesco Sambo, and Barbara Di Camillo. bnstruct: an R package for Bayesian network structure learning in the presence of missing data. *Bioinformatics*, 33(8):1250–1252, 2017. doi: 10.1093/bioinformatics/btw807.

Ming Gao, Yi Ding, and Bryon Aragam. A polynomial-time algorithm for learning nonparametric causal graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11599–11611, 2020.

Enrico Giudice, Jack Kuipers, and Giusi Moffa. The dual PC algorithm for structure learning. In *Proceedings of the 11th International Conference on Probabilistic Graphical Models*, volume 186 of *Proceedings of Machine Learning Research*, pp. 301–312. PMLR, 2022.

Jiaying Gu and Qing Zhou. Learning big Gaussian Bayesian networks: Partition, estimation and fusion. *Journal of Machine Learning Research*, 21(158):1–31, 2020.

Ruocheng Guo, Lu Cheng, Jundong Li, P. Richard Hahn, and Huan Liu. A survey of learning causality with data: Problems and methods. *ACM Computing Surveys (CSUR)*, 53(4):1–37, 2020. doi: 10.1145/3397269.

Neville Kenneth Kitson, Anthony C. Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of Bayesian network structure learning. *Artificial Intelligence Review*, 56:8721–8814, 2023. doi: 10.1007/s10462-022-10351-w.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: principles and techniques*. MIT Press, 2009.

Thuc Duy Le, Tao Hoang, Jiuyong Li, Lin Liu, Huawen Liu, and Shu Hu. A fast PC algorithm for high dimensional causal discovery with multi-core PCs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(5):1483–1495, 2019. doi: 10.1109/TCBB.2016.2591526.

Hui Liu, Shuigeng Zhou, Wai Lam, and Jihong Guan. A new hybrid method for learning Bayesian networks: Separation and reunion. *Knowledge-Based Systems*, 121:185–197, 2017. doi: 10.1016/j.knosys.2017.01.029.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 3:121–129, 2017.

Mauro Scanagatta, Cassio P. de Campos, Giorgio Corani, and Marco Zaffalon. Learning Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1864–1872, 2015.

Amirhossein Shahbazinia, Saber Salehkaleybar, and Matin Hashemi. ParaLiNGAM: Parallel causal structure learning for linear non-Gaussian acyclic models. *Journal of Parallel and Distributed Computing*, 176:114–127, 2023. doi: 10.1016/j.jpdc.2023.01.007.

Ajit P Singh and Andrew W Moore. Finding optimal Bayesian networks by dynamic programming. Technical Report CMU-CALD-05-106, Carnegie Mellon University, School of Computer Science, 2005.

Arjun Sondhi and Ali Shojaie. The reduced PC-algorithm: Improved causal structure learning in large random networks. *Journal of Machine Learning Research*, 20(164):1–31, 2019.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT Press, 2nd edition, 2000.

Topi Talvitie, Ralf Eggeling, and Mikko Koivisto. Learning Bayesian networks with local structure, mixed variables, and exact algorithms. *International Journal of Approximate Reasoning*, 115: 69–95, 2019. doi: 10.1016/j.ijar.2019.09.002.

Willem van den Boom, Maria De Iorio, and Alexandros Beskos. Bayesian learning of graph substructures. *Bayesian Analysis*, 18(4):1311–1339, 2023. doi: 10.1214/22-BA1338.

Xianchao Xie, Zhi Geng, and Qiang Zhao. Decomposition of structural learning about directed acyclic graphs. *Artificial Intelligence*, 170(4-5):422–439, 2006.

Changhe Yuan, Brandon Malone, and Xiaojian Wu. Learning optimal Bayesian networks using A* search. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 2186–2191, 2011.

Behrooz Zarebavani, Foad Jafarinejad, Matin Hashemi, and Saber Salehkaleybar. cuPC: CUDA-based parallel PC algorithm for causal structure learning on GPU. *IEEE Transactions on Parallel and Distributed Systems*, 31(3):530–542, 2020. doi: 10.1109/TPDS.2019.2939126.

Hao Zhang, Shuigeng Zhou, Chuanxu Yan, Jihong Guan, Xin Wang, Ji Zhang, and Jun Huan. Learning causal structures based on divide and conquer. *IEEE Transactions on Cybernetics*, 52(5): 3232–3243, 2022. doi: 10.1109/TCYB.2020.3010004.

A PROOFS FOR THE EVALUATED GLOBAL-CLEANUP METHOD

Edge preservation. If $x - y$ is a true adjacency, the length-one path between its endpoints is active given every set excluding x, y. By faithfulness, no such conditioning set separates them. Consider a member U containing both endpoints. If neither is in W, they are adjacent in $H(U, W)$ and hence belong to the same component. If exactly one is in W, it is added to the component containing the other. If both are in W, they belong to every child. Thus some child contains both endpoints. Other cover members are unchanged, so induction over all splits proves the claim. □

Proof of Lemma 3.3. By Lemma 3.3, each true adjacency is considered by at least one local learner. Removing it would require a separating set excluding its endpoints, which does not exist under faithfulness. Local CI statements are computed from marginal distributions of the same joint distribution P; a statement $x \perp y \mid S$ does not become false merely because other variables are unmeasured by that local query. Consequently, every true adjacency survives at least one local skeleton, and no valid local CI witness can remove it from the union. For a pair in distinct components of $H(U, W)$, there is no edge between them, so the definition of H supplies the valid witness W. All stored witnesses are therefore sound. □

Proof of Theorem 3.4. The initial candidate graph for global cleanup contains every true adjacency by Lemma 3.3. Such adjacencies are never removed by an oracle test. For any nonadjacent pair x, y in the true DAG, choose the endpoint x for which y is a nondescendant; such an endpoint exists because the graph is acyclic. Since the pair is not adjacent, y is not a parent of x. The local Markov property gives $x \perp y \mid \mathrm{Pa}_G(x)$. All parents of x remain among its current neighbors, since their edges are true adjacencies. Unrestricted global PC, testing subsets from both endpoints, therefore eventually encounters a separating set for this pair if its candidate edge is still present. The final skeleton is exactly the true skeleton.

Every absent pair has a valid witness from a cover split, a local removal, or a global removal. For an unshielded triple $x - z - y$, if z is a noncollider, the length-two path is active unless z belongs to a separating set. If z is a collider, including z activates that path, so no valid separating set contains it. Thus the oracle collider rule identifies exactly the unshielded colliders. A DAG's skeleton and unshielded colliders determine its Markov-equivalence class; sound and complete orientation rules return its CPDAG, as in standard oracle PC (Spirtes et al., 2000). □

Scalar–tensor equivalence. Both implementations obtain the same $N_{ab,s}$ by counting the same samples in the same strata. The marginal counts, expected counts, and degrees of freedom are therefore equal. Equation equation 1 differs only in the grouping of arithmetic operations, which does not change its exact-arithmetic value. Applying the same reference distribution and threshold yields the same decision. Finite-precision summation can introduce numerical differences and is not covered by this algebraic argument. □

Termination and scope. Each successful split with at least two nonempty components replaces a member by strictly smaller members. With a finite variable set, bounded separator size, and memoized exhausted searches, cover construction terminates. This statement does not bound practical runtime by a small constant. If no split is found, the singleton cover $\{V\}$ preserves the oracle guarantee, but provides no decomposition speedup. Restricting separator sizes does not restrict the conditioning-set sizes in the theorem's global cleanup. The theorem makes no guarantee for a score-based local learner or a run stopped by a resource cap.

B ORIGINAL COVER DEFINITIONS AND MATRIX EXPLANATION

The original cover definitions and dependency-matrix explanation are retained in Section 3 of the main text.

C ORIGINAL SEARCH VARIANTS AND EXPLANATIONS

These earlier variants are retained to document the original construction. The controlled rerun uses the direct adaptive search described in the main text rather than all of these alternatives.

We provide Algorithm 2 as the distributed learning algorithm to solve Problem 1. The algorithm consists of three sub-algorithms: One that performs a conditional P-map reduction (solves Problem 3.1), one that learns each of the elements of the reduction (interiors and between interiors and boundaries), and finally, one to append the elements of the reduction (learning the boundaries).

Algorithm 2: Distributed structure learner

Input: Set of random variables $\mathcal{X}$ with joint probability distribution P that admit a P-map; the maximum number of variables in each element of the cover, d; the maximum number of separator variables, W

Output: A P-map $\mathcal{G}$ for P

$\{\mathcal{X}_1, \ldots, \mathcal{X}_I\} \leftarrow$ Algorithm 3 (d, W);
`// Alternatively, Algorithm 5 can be used.`
for $i = 1, \cdots, I$
  $\mathcal{G}_i \leftarrow$ P-map learner($\mathcal{X}_i$)
$\bar{\mathcal{G}} = \bigcup_{i=1}^{I} \mathcal{G}_i$
for $i = 1, \cdots, I$
  $\mathcal{G} \leftarrow$ Boundary PC $(\mathcal{X}_i, \bar{\mathcal{G}})$

We provide Algorithms 3 and 5 for the first part, i.e., to solve Theorem 3.1. Both are based on the idea to iteratively find separators that would decompose the components of a conditional P-map reduction of $\mathcal{X}$ into another reduction.

Algorithm 3 goes through the cover $\mathcal{X}_\mathcal{I}$ (initially set to $\{\mathcal{X}\}$), picks the greatest component $\mathcal{U} \in \mathcal{X}_\mathcal{I}$, and checks if any subset $\mathcal{W} \subset \mathcal{U}$ separates the component into a cover of cardinality greater than one (which is a conditional P-map reduction for $\mathcal{U}$). If so, then the algorithm updates the cover $\mathcal{X}_\mathcal{I}$ by replacing $\mathcal{U}$ with its cover and moves to the next greatest component in $\mathcal{X}_\mathcal{I}$. This process continues until either all components of the cover have a size less than d or none of the components can be further reduced. The notation $\mathcal{P}(\mathcal{U})$ is the power set of the set $\mathcal{U}$, i.e., the set of all subsets of $\mathcal{U}$.

The other conditional P-map finder is Algorithm 5. The problem with Algorithm 3 is that it may run too many CI tests. When searching for separators of size w of a cover element $\mathcal{U}$ with cardinality u, all $\binom{u}{w}$ subsets of $\mathcal{U}$ are checked for being a separator; next all $\binom{u}{w+1}$ subsets are checked and so on. However, in Algorithm 5, once a separator of size w is found, the algorithm searches for all single nodes to be added to this separator, those are, $\binom{u-w}{1}$; next all $\binom{u-w}{2}$, and so on, until another reduction happens. The problem with Algorithm 5 is that the number of conditioning variables in the CI tests grows quickly, because the separators are never removed from the conditioning part. Namely, once a cover element is reduced into another cover by a separator of size w, the next reduction will require a CI test with a conditioning of size at least $w + 1$ as both the previous separator and a new separator of size one will be conditioned on. This does not happen in Algorithm 3; namely, once a separator of size w is found, the CI test required for finding the next separator will have again w conditioning variables, as the algorithm does not condition on the previously found separator. For the same reason, Algorithm 5 may be unable to find a capped-d cover for small values of d.

After the cover finding process, a structural learning method is individually applied to each set of covering variables, resulting in the discovery of the network structure for each subset of variables. Let $\{\mathcal{X}_1, \cdots, \mathcal{X}_I\}$ denote a covering of $\mathcal{X}$ that is derived from Algorithms 3 or 5. Each subset of variables, $\mathcal{X}_i$ for $i = 1, \cdots, I$, can be independently learned using either score-based or constraint-based algorithms. This leads to the identification of the local structures, denoted as $\mathcal{G}_i$ for $i = 1, \cdots, I$.

Algorithm 3: Parallel conditional P-map reduction finder CI based

Input: $\mathcal{X} = \{X_1, \cdots, X_n\}$, d, W, and the number of processors N_p.
Output: A conditional P-map reduction $\mathcal{X}_\mathcal{I}$ of $\mathcal{X}$

$\mathcal{X}_\mathcal{I} \leftarrow \{\mathcal{X}\}$;
$w \leftarrow 0$; // w: the size of the separator
$\mathcal{K} \leftarrow \emptyset$
while $\max_{\mathcal{U} \in \mathcal{X}_\mathcal{I}} |\mathcal{U}| > d$ *and* $w \leq W$ **do**
  $\mathcal{X}'_\mathcal{I} \leftarrow \mathcal{X}_\mathcal{I}$; // $\mathcal{X}'_\mathcal{I}$: potentially decomposable cover members (w.r.t. w)
  while $\mathcal{X}'_\mathcal{I} \neq \emptyset$ *and* $\max_{\mathcal{U}' \in \mathcal{X}'_\mathcal{I}} |\mathcal{U}'| > d$ **do**
    $\mathcal{U} \leftarrow \arg\max_{\mathcal{U}' \in \mathcal{X}'_\mathcal{I}} |\mathcal{U}'|$; // $\mathcal{U}$: greatest cover member
    for $\mathcal{M} \subset \mathcal{P}(\mathcal{U}) \setminus \mathcal{K}$ *and* $|\mathcal{M}| = N_p$ *and* $|\mathcal{W}| = w$ *for* $\mathcal{W} \in \mathcal{M}$
      for $\mathcal{W} \in \mathcal{M}$
        $\{\mathcal{W} \cup \mathcal{C}^i_\mathcal{W}\}_{i=1}^{I_\mathcal{W}} \leftarrow$ the cover for $\mathcal{U}$ separated by $\mathcal{W}$;
      $\mathcal{K} \leftarrow \mathcal{K} \cup \mathcal{M}$
      if $\sum_{\mathcal{W} \in \mathcal{M}} I_\mathcal{W} > N_p$ // the cover was a reduction
        $\mathcal{X}_\mathcal{I} \leftarrow$ Cover finding for $\mathcal{U}$ by intersection method on $\{\mathcal{W} \cup \mathcal{C}^i_\mathcal{W}\}_{i=1}^{I_\mathcal{W}}$ if $I_\mathcal{W} > 1$; // update the cover
        $\mathcal{X}_\mathcal{I} \leftarrow \mathcal{X}_\mathcal{I} \setminus \{\mathcal{U}\}$
        $\mathcal{X}'_\mathcal{I} \leftarrow \mathcal{X}_\mathcal{I}$
        Break;
    $\mathcal{X}'_\mathcal{I} \leftarrow \mathcal{X}'_\mathcal{I} \setminus \{\mathcal{U}\}$
  $w \leftarrow w + 1$;

Algorithm 4: Recursive conditional P-map reduction finder CI based

Input : $\mathcal{X} = \{X_1, \ldots, X_n\}$, d, W, and the number of processors N_p
Output : A conditional P-map reduction $\mathcal{X}_\mathcal{I}$ of $\mathcal{X}$

Function REDUCE($\mathcal{X}_\mathcal{I}, w$)
  if $w > W$ ***or*** $\max_{\mathcal{X}_i \in \mathcal{X}_\mathcal{I}} |\mathcal{X}_i| \leq d$ **then**
    return $\mathcal{X}_\mathcal{I}$
  $\mathcal{X}'_\mathcal{I} \leftarrow \{\mathcal{X}_i \in \mathcal{X}_\mathcal{I} : |\mathcal{X}_i| > d\}$ ordered by decreasing cardinality
  foreach $\mathcal{X}_{\max} \in \mathcal{X}'_\mathcal{I}$ **do**
    Partition $\{\mathcal{W} \subseteq \mathcal{X}_{\max} : |\mathcal{W}| = w\}$ into batches $\mathcal{M}$ of size at most N_p
    foreach $\mathcal{M}$ **do**
      foreach $\mathcal{W} \in \mathcal{M}$ **do** // in parallel
        $\{\mathcal{W} \cup \mathcal{C}^i_\mathcal{W}\}_{i=1}^{I_\mathcal{W}} \leftarrow$ the cover for $\mathcal{X}_{\max}$ separated by $\mathcal{W}$
      if $\exists\, \mathcal{W} \in \mathcal{M}$ *such that* $I_\mathcal{W} > 1$ **then**
        $\{\tilde{\mathcal{X}}_1, \ldots, \tilde{\mathcal{X}}_J\} \leftarrow$ Cover finding for $\mathcal{X}_{\max}$ by intersection method on $\{\mathcal{W} \cup \mathcal{C}^i_\mathcal{W}\}_{i=1}^{I_\mathcal{W}}$ for all $\mathcal{W} \in \mathcal{M}$ with $I_\mathcal{W} > 1$
        $\mathcal{X}_\mathcal{I} \leftarrow (\mathcal{X}_\mathcal{I} \setminus \{\mathcal{X}_{\max}\}) \cup \{\tilde{\mathcal{X}}_1, \ldots, \tilde{\mathcal{X}}_J\}$
        return REDUCE($\mathcal{X}_\mathcal{I}, w$)
  return REDUCE($\mathcal{X}_\mathcal{I}, w + 1$)
return REDUCE($\{\mathcal{X}\}, 0$)

In $\mathcal{G}_i$, the all edges between every two interior variables, and between a variable in the interior and another in the boundary set equal to the edges of a P-map $\mathcal{G}$ for P. In the process of constructing the comprehensive network structure, the local structures obtained in the prior stage must be concatenated together. Each cover set shares a common variable set, denoted as $\mathrm{bd}(\mathcal{X}_i)$, representing the observed variables that correspond to specific nodes in the local networks. By learning the edges between the boundary variables and assembling the local networks using these common nodes, the comprehensive network structure for all variables is constructed.

Algorithm 5: Parallel Non-monotone conditional P-map reduction finder CI based

Input: $\mathcal{X} = \{X_1, \cdots, X_n\}$, d, W, and the number of processors N_p.
Output: A conditional P-map reduction $\mathcal{X}_\mathcal{I}$ of $\mathcal{X}$

$\mathcal{X}_\mathcal{I} \leftarrow \{\mathcal{X}\}$;
$w \leftarrow 0$; `// w: the size of the separator`
while $w \leq W$ *and* $\max_{\mathcal{U} \in \mathcal{X}_\mathcal{I}} |\mathcal{U}| > d$ **do**
  $\mathcal{X}'_\mathcal{I} \leftarrow \mathcal{X}_\mathcal{I}$; `// X'_I: potentially decomposable cover members (w.r.t. w)`
  $\mathcal{K} \leftarrow \emptyset$;
  while $\mathcal{X}'_\mathcal{I} \neq \emptyset$ *and* $\max_{\mathcal{U}' \in \mathcal{X}'_\mathcal{I}} |\mathcal{U}'| > d$ **do**
    $\mathcal{U} = \arg\max_{\mathcal{U}' \in \mathcal{X}'_\mathcal{I}} |\mathcal{U}'|$; `// U: greatest cover member`
    $\mathrm{bd}(\mathcal{U}) \leftarrow \bigcup_{\mathcal{U}' \in \mathcal{X}_\mathcal{I} \setminus \{\mathcal{U}\}} \mathcal{U} \cap \mathcal{U}'$
    for $\mathcal{M} \subset \mathcal{P}(\mathcal{U} \setminus (\mathrm{bd}(\mathcal{U}) \cup \mathcal{K}))$ *and* $|\mathcal{M}| = N_p$ *and* $|\mathcal{W}| = w$ *for* $\mathcal{W} \in \mathcal{M}$
      for $\mathcal{W} \in \mathcal{M}$
        $\mathcal{X}_\mathcal{W} \leftarrow \{\mathcal{U}\}$
        $\{\mathcal{W} \cup \mathrm{bd}(\mathcal{U}) \cup \mathcal{C}^i_{\mathcal{W} \cup \mathrm{bd}(\mathcal{U})}\}_{i=1}^{I_\mathcal{W}} \leftarrow$ the cover for $\mathcal{U}$ separated by $\mathcal{W} \cup \mathrm{bd}(\mathcal{U})$;
        if $I_\mathcal{W} > 1$ `// the cover was a reduction`
          for $U \in \mathrm{bd}(\mathcal{U})$
            for $i = 1, \cdots, I_\mathcal{W}$
              if $U \not\perp \mathcal{C}^i_{\mathcal{W} \cup \mathrm{bd}(\mathcal{U})} | \mathcal{W} \cup \mathrm{bd}(\mathcal{U}) \setminus \{U\}$
                $\mathcal{C}^i_\mathcal{W} \leftarrow \{U\} \cup \mathcal{C}^i_{\mathcal{W} \cup \mathrm{bd}(\mathcal{U})}$
          if $\mathrm{bd}(\mathcal{U}) \setminus \cup \mathcal{C}^i_\mathcal{W} \neq \emptyset$
            $\mathcal{X}_\mathcal{W} \leftarrow \{\mathcal{W} \cup (\mathrm{bd}(\mathcal{U}) \setminus \cup \mathcal{C}^i_\mathcal{W})\}$;
          $\mathcal{X}_\mathcal{W} \leftarrow (\mathcal{X}_\mathcal{W} \setminus \{\mathcal{U}\}) \cup \{\mathcal{W} \cup \mathcal{C}^i_\mathcal{W}\}_{i=1}^{I_\mathcal{W}}$;
        if $\sum_{\mathcal{W} \in \mathcal{M}} I_\mathcal{W} > N_p$ `// the cover was a reduction`
          $\mathcal{X}_\mathcal{I} \leftarrow$ Covering $\mathcal{U}$ by intersection method on $\mathcal{X}_\mathcal{W}$ if $I_\mathcal{W} > 1$; `// update the cover`
          $\mathcal{X}_\mathcal{I} \leftarrow \mathcal{X}_\mathcal{I} \setminus \{\mathcal{U}\}$
          $\mathcal{X}'_\mathcal{I} \leftarrow \mathcal{X}_\mathcal{I}$
          $w \leftarrow 1$
          Break;
        else
          $\mathcal{K} \leftarrow \mathcal{K} \cup \mathcal{M}$
    $\mathcal{X}'_\mathcal{I} \leftarrow \mathcal{X}'_\mathcal{I} \setminus \{\mathcal{U}\}$
  $w \leftarrow w + 1$;

Algorithm 6: Separator-based reduction finder with parallel CI tests

Input : Variable set $\mathcal{X} = \{X_1, \ldots, X_n\}$; access to conditional-independence tests; target cover-member size $d \geq 1$; maximum separator size $W \geq 0$; number of processors $N_p \geq 1$

Output : A separator-based candidate P-map reduction $\mathcal{X}_{\mathcal{I}}$

Function REDUCE($\mathcal{U}$)
  if $|\mathcal{U}| \leq d$ **then**
    return $\{\mathcal{U}\}$
  for $w = 0, \ldots, \min\{W, |\mathcal{U}| - 2\}$ **do**
    foreach $\mathcal{W} \subseteq \mathcal{U}$ *with* $|\mathcal{W}| = w$ **do**
      `// Process one separator at a time`
      List the remaining variables as $\mathcal{U} \setminus \mathcal{W} = \{Y_1, \ldots, Y_m\}$
      $D_{\mathcal{W}} \leftarrow I_m$
      `//` I_m `is the identity matrix`
      `// Distribute the following tests over at most` N_p `processors`
      parallel for each *pair* (a, b) *with* $1 \leq a < b \leq m$ **do**
        Test $Y_a \perp Y_b \mid \mathcal{W}$
        if *the test reports dependence* **then**
          $D_{\mathcal{W}}(a, b) \leftarrow D_{\mathcal{W}}(b, a) \leftarrow 1$
      Wait until all tests for $\mathcal{W}$ finish
      Find the connected components $\mathcal{C}^1_{\mathcal{W}}, \ldots, \mathcal{C}^{I_{\mathcal{W}}}_{\mathcal{W}}$ of the graph represented by $D_{\mathcal{W}}$
      if $I_{\mathcal{W}} > 1$ **then**
        `// Form the cover separated by` $\mathcal{W}$
        $\mathcal{S}_{\mathcal{W}} \leftarrow \{\mathcal{W} \cup \mathcal{C}^i_{\mathcal{W}}\}_{i=1}^{I_{\mathcal{W}}}$
        `// Apply the same procedure to each cover member`
        $\mathcal{R} \leftarrow \emptyset$
        foreach $\mathcal{V} \in \mathcal{S}_{\mathcal{W}}$ **do**
          $\mathcal{R} \leftarrow \mathcal{R} \cup$ REDUCE($\mathcal{V}$)
        return $\mathcal{R}$
      `// Otherwise, try the next separator`
  `// No tested separator splits` $\mathcal{U}$
  return $\{\mathcal{U}\}$
$\mathcal{X}_{\mathcal{I}} \leftarrow$ REDUCE($\mathcal{X}$)
return $\mathcal{X}_{\mathcal{I}}$

Algorithm 7: The Boundary PC Algorithm

Input: The union of $\{\mathcal{G}_i\}_{i=1}^{I}$, cover sets $\{\mathcal{X}_i\}_{i=1}^{I}$ and joint probability distribution P

Output: A P-map $\bar{\mathcal{G}}$ for P

$\mathrm{Sep}(X, Y) = \emptyset$ for all $X, Y \in \mathrm{bd}(\mathcal{X}_i)$;
for $X \in \mathrm{bd}(\mathcal{X}_i)$
  for $Y \in \mathrm{bd}(\mathcal{X}_i) \cap \mathrm{Adj}(\bar{\mathcal{G}}, X)$
    $\mathcal{U} = (\bigcup_{X \in \mathcal{X}_k, Y \notin \mathcal{X}_k} \mathrm{bd}(\mathcal{X}_k)) \cup (\bigcup_{Y \in \mathcal{X}_k, X \notin \mathcal{X}_k} \mathrm{bd}(\mathcal{X}_k)) \setminus \{X, Y\}$
    if $X \perp Y \mid \mathcal{U}$
      Remove the edge $X - Y$ from $\bar{\mathcal{G}}$;
      $\mathrm{Sep}(X, Y) \leftarrow \mathcal{U}$;
For $X - Z - Y$ that $X, Y \in \mathrm{bd}(\mathcal{X}_i)$ are not adjacent in $\bar{\mathcal{G}}$, if $Z \notin \mathrm{Sep}(X, Y)$ then orient $X - Z - Y$ as an immolarity $X \rightarrow Z \leftarrow Y$.
Orient the other edges using the orientation rules in (Spirtes et al., 2000).

D ORIGINAL BOUNDARY-SPECIFIC THEORETICAL DEVELOPMENT

The following original statements and proof arguments concern the boundary-specific formulation. They are preserved here separately from the global-cleanup guarantee in Theorem 3.4; the new experiments do not validate stronger claims about arbitrary local learners or a single fixed boundary conditioning set.

Proposition D.1. *Consider random variables $\mathcal{X}$ with distribution P. A cover $\mathcal{X}_\mathcal{I} = \{\mathcal{X}_1, \ldots, \mathcal{X}_I\}$ satisfying*

$$\forall i, j \neq i \qquad \mathcal{X}_i \perp \mathcal{X}_j \mid \mathrm{bd}(\mathcal{X}_i) \tag{2}$$

is a conditional P-map reduction.

Proposition D.2. *Consider random variables $\mathcal{X}$ with distribution P that admits a P-map class PDAG $\mathcal{G}$. Assume that $\mathcal{X}_\mathcal{I} = \{\mathcal{X}_i\}_{i=1}^I$ is a conditional P-map reduction for $\mathcal{X}$, and consider an arbitrary $i \in \{1, \ldots, I\}$. Two nodes $X_1, X_2 \in \mathcal{X}_i$ where at least one of which is from $\mathcal{X}_i^o$ are adjacent (resp. non-adjacent) in $\mathcal{G}[\mathcal{X}_i]$ if and only if they are adjacent (resp. non-adjacent) in the P-map class PDAG of $P[\mathcal{X}_i]$. Moreover, every triple of nodes $X_1, X_2, Z \in \mathcal{X}_i$, where at least two of which are in $\mathcal{X}_i^o$, form an immorality in $\mathcal{G}$ if and only if they do so in the P-map class PDAG of $P[\mathcal{X}_i]$. Finally, if a triple of nodes $X_1, X_2, Z \in \mathcal{X}_i$, where $X_1 \in \mathcal{X}_i^o$, form an immorality $X_1 \to Z \leftarrow X_2$ in the P-map class PDAG of $P[\mathcal{X}_i]$, then the immorality also exists in $\mathcal{G}$.*

Lemma D.3. *The cover output by Algorithms 3 and 5 satisfies Condition equation 2.*

Lemma D.4. *Consider random variables $\mathcal{X}$ with distribution P that admits a P-map class PDAG $\mathcal{G}$. Assume that $\mathcal{X}_\mathcal{I} = \{\mathcal{X}_i\}_{i=1}^I$ is a conditional P-map reduction for $\mathcal{X}$, and consider an arbitrary $J \subseteq \{1, \ldots, I\}$. Two nodes $X_1, X_2 \in \mathrm{bd}(\mathcal{X}_j)$ for all $j \in J$, are non-adjacent in $\mathcal{G}$ if and only if they are non-adjacent in a P-map class PDAG of $P[\mathcal{X}_{j_0}]$ for at least a $j_0 \in J$.*

Lemma D.5. *Under Algorithms 3 and 5, every edge in $\mathcal{G}_i$ belongs to a cover component.*

Lemma D.6. *Let $\mathcal{X}_\mathcal{I} = \{\mathcal{X}_1, \ldots, \mathcal{X}_I\}$ be the conditional P-map reduction returned by Algorithms 3 or 4, and let G be the true P-map for P. Consider two boundary nodes $X, Y \in \mathrm{bd}(\mathcal{X}_i)$. Define*

$$\mathcal{U}_{XY} = \left(\bigcup_{k:\, X \in \mathcal{X}_k,\, Y \notin \mathcal{X}_k} \mathrm{bd}(\mathcal{X}_k) \right) \cup \left(\bigcup_{k:\, Y \in \mathcal{X}_k,\, X \notin \mathcal{X}_k} \mathrm{bd}(\mathcal{X}_k) \right) \setminus \{X, Y\}.$$

Then X and Y are non-adjacent in G if and only if

$$X \perp Y \mid \mathcal{U}_{XY}.$$

proof. Since P is faithful to the true P-map G, it is enough to prove the equivalent d-separation statement in G.

First, suppose that X and Y are adjacent in G. Then the edge $X - Y$ itself is a trail between X and Y with no intermediate node. Since $X, Y \notin \mathcal{U}_{XY}$, this trail cannot be blocked by conditioning on $\mathcal{U}_{XY}$. Therefore, X and Y are d-connected given $\mathcal{U}_{XY}$. By faithfulness,

$$X \not\perp Y \mid \mathcal{U}_{XY}.$$

Thus, if $X \perp Y \mid \mathcal{U}_{XY}$, then X and Y cannot be adjacent in G.

Conversely, suppose that X and Y are not adjacent in G. We show that every trail between X and Y is blocked by conditioning on $\mathcal{U}_{XY}$. Let π be an arbitrary trail between X and Y.

If π is entirely contained in a cover element that contains both X and Y, then the absence of the edge $X - Y$ is detected by the corresponding local P-map learner, because the local learner correctly identifies the structure inside each cover component.

It remains to consider the case where π is not contained in a single cover element containing both X and Y. Then π must pass from a cover element containing X but not Y, or from a cover element containing Y but not X, to another cover element. Hence, by construction of the cover, π must pass through at least one boundary node belonging to

$$\left(\bigcup_{k:\, X \in \mathcal{X}_k,\, Y \notin \mathcal{X}_k} \mathrm{bd}(\mathcal{X}_k) \right) \cup \left(\bigcup_{k:\, Y \in \mathcal{X}_k,\, X \notin \mathcal{X}_k} \mathrm{bd}(\mathcal{X}_k) \right).$$

After excluding X and Y, this node belongs to $\mathcal{U}_{XY}$.

By the boundary property proved for the cover, such a boundary node cannot be a collider on a trail connecting two cover components. Therefore, the trail π contains a non-collider node that belongs to $\mathcal{U}_{XY}$. Since conditioning on a non-collider blocks a trail, π is blocked by $\mathcal{U}_{XY}$.

Because π was arbitrary, every trail between X and Y is blocked by $\mathcal{U}_{XY}$. Hence,

$$d\text{-sep}_G(X, Y \mid \mathcal{U}_{XY}).$$

Using faithfulness of P to G, we obtain

$$X \perp Y \mid \mathcal{U}_{XY}.$$

Combining the two directions proves the claim. □

Theorem D.7. *Algorithm 2 outputs a P-map for P.*

Proof. It suffices to prove that the output of the algorithm, say $\hat{\mathcal{G}}$, has the same skeleton and immoralities as the P-map PDAG class for P, say $\mathcal{G}$. In view of Lemma D.3, Proposition D.1, Algorithm 2 outputs a cover $\mathcal{X}_{\mathcal{I}} = \{\mathcal{X}_i\}_{i=1}^{I}$ that is a conditional P-map reduction. Hence, the interior nodes of no two elements of the cover are adjacent in both $\hat{\mathcal{G}}$ and $\mathcal{G}$. On the other hand, Proposition D.2 guarantees that all edges between every interior node of element $\mathcal{X}_i$ and another node in $\mathcal{X}_i$ are correctly identified for every i. So it only remains to show that the algorithm also correctly identifies the edges between the boundary nodes of every $\mathcal{X}_i$. Denote by $\mathcal{G}'$ the graph obtained by Algorithm 2, before executing (sub-)Algorithm 7. Assume that there is an edge between two boundary nodes $X, Y \in \mathcal{X}_i$ in $\mathcal{G}'$ that does not exist in $\mathcal{G}$. Since the edge does not exist in the P-map $\mathcal{G}$, there is some $\mathcal{U} \in \mathcal{X}$ such that $X_1 \perp X_2 \mid \mathcal{U}$. In view of Lemma E.4, the set $\mathcal{U}$ can be chosen such that all nodes in $\mathcal{U}$ are either adjacent with X_1 or adjacent with X_2 in $\mathcal{G}$. Denote the nodes adjacent with X_1 (resp. X_2) in $\mathcal{G}$ by $\mathcal{N}_{X_1}^{\mathcal{G}}$ (resp. $\mathcal{N}_{X_2}^{\mathcal{G}}$). It follows from Lemma D.5 that $\mathcal{N}_{X_i}^{\mathcal{G}} = \cup_{j=1}^{I} \mathcal{N}_{X_i}^{\mathcal{G}[\mathcal{X}_j]}$ for $i = 1, 2$. Since $\mathcal{G}'$ includes all of the edges in $\mathcal{G}$, it follows that the the set $\mathcal{U}$ can be found by searching through the union of the neighbors of X_1 (resp. X_2) in each $\mathcal{X}_j$, which is what Algorithm 7 does. Hence, the edge will be detected and eliminated from $\mathcal{G}'$. Finally, Algorithm 2 does not delete an edge that actually exists in $\mathcal{G}$ as the elimination of an edge in Algorithm 7 happens only if a conditional independence holds between the variables.

As with the immoralities, it follows from Proposition D.2, that every immorality in $\mathcal{G}'$ with at least one interior node of some cover element $\mathcal{X}_i$ also exists in $\mathcal{G}$. Now if the edges of an immorality in $\mathcal{G}'$ with all three nodes being a boundary node of some cover element $\mathcal{X}_i$ are not eliminated in $\hat{\mathcal{G}}$, then the immorality also exists in $\mathcal{G}$. So all immoralities in $\mathcal{G}'$ that also appear in $\hat{\mathcal{G}}$ belong to $\mathcal{G}$. On the other hand, if an immorality emerges after executing Algorithm 7, i.e., it belongs to $\hat{\mathcal{G}}$ but not $\mathcal{G}'$, then it should also belong to $\mathcal{G}$, because Algorithm 7 is basically the PC algorithm that starts from the graph $\mathcal{G}'$ that is a superset $\mathcal{G}$ (and PC is known to correctly identify the immoralities). Therefore, every immorality in $\hat{\mathcal{G}}$ is included in $\mathcal{G}$. Now we show that that every immorality in $\mathcal{G}$ is included in $\hat{\mathcal{G}}$. In view of Lemma D.5 every edge is on a cover element. Moreover, it is impossible to have three boundary nodes X, Y, and Z forming a collider $X \to Z \leftarrow Y$, and the three nodes do not belong to the same cover element, because then the element including X and that including Y will not be d-separated conditioned on Z. Hence, according to Proposition D.2, we only need to show that the immoralities in $\mathcal{G}$ with all three nodes belonging to the boundary of some element, or when exactly one node is an interior and the other two are the boundary of the same element. The proof of the first part is similar to the previous case (Algorithm 7 being basically the PC algorithm) and it checks the existence of every boundary edge. For the second part, we have a node $X \in \mathcal{X}_i^o$ for some i, and two boundary nodes $Y, Z \in \mathrm{bd}(\mathcal{X}_i)$, such that $Y \to X \leftarrow Z$ is an immorality in $\mathcal{G}$. If the immorality also exists in $\mathcal{G}'$, there is nothing to prove. Otherwise, Y and Z are adjacent in $\mathcal{G}'$, but the edge will be eliminated by Algorithm 7 and then checked for such immorality. This completes the proof. □

E Additional original definitions and proofs

The following definition is a reformulation of Definition 3.1 in terms of P-map class PDAGs.

Definition E.1. Consider integer $d \geq 0$ and the set of random variables $\mathcal{X}$ with distribution P that admits a P-map, and let $\mathcal{G}$ be the P-map class PDAG for P. A cover $\{\mathcal{X}_1, \ldots, \mathcal{X}_I\}$ of $\mathcal{X}$ is a d-capped P-map reduction if for every $i = 1, \ldots, I$, *(i)* $|\mathcal{X}_i| \leq d$ and *(ii)* the P-map class PDAG for $P[\mathcal{X}_i]$ equals $\mathcal{G}[\mathcal{X}_i]$, and *(iii)* the P-map class PDAG for P equals $\cup_{i=1}^{I} \mathcal{G}[\mathcal{X}_i]$.

Remark E.2. Let $\mathcal{I}(\mathcal{G})[\mathcal{Y}]$ denote those conditional independencies in $\mathcal{I}(\mathcal{G})$ that are over nodes $\mathcal{Y}$, i.e., $\mathcal{I}(\mathcal{G})[\mathcal{Y}] = \{(\mathcal{X}_1 \perp \mathcal{X}_2 \mid \mathcal{X}_3) \in \mathcal{I}(\mathcal{G}) : \mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3 \subseteq \mathcal{Y}\}$. In Example 1, the cover $\{\{X_1, X_2, X_3\}, \{X_4, X_3\}, \{X_5, \mathcal{X}_3\}\}$ (Figure 1.b) is a capped-d P-map reduction for $\{X_1, \ldots, X_5\}$ for any $d \geq 3$. Conditions *(i)* and *(iii)* in Definition E.1 are clearly met. For Condition *(ii)*, we show that the set of conditional independencies of each subset in the cover, e.g., $\{X_1, X_2, X_3\}$, matches the set of d-separations of the corresponding sub-graph, i.e., $\mathcal{I}(P[X_1, X_2, X_3]) = \mathcal{I}(\mathcal{G}[X_1, X_2, X_3])$. According to the d-separations in $\mathcal{G}$, $\mathcal{I}(\mathcal{G}[X_1, X_2, X_3]) = \mathcal{I}(\mathcal{G}[X_4, X_3]) = \mathcal{I}(\mathcal{G}[X_5, X_3]) = \emptyset$. On the other hand, since $\mathcal{G}$ is a P-map for P, $\mathcal{I}(P[X_1, X_2, X_3]) = \mathcal{I}(\mathcal{G})[X_1, X_2, X_3] = \emptyset$, $\mathcal{I}(P[X_4, X_3]) = \mathcal{I}(\mathcal{G})[X_4, X_3] = \emptyset$, and $\mathcal{I}(P[X_5, X_3]) = \mathcal{I}(\mathcal{G})[X_5, X_3] = \emptyset$.

Remark E.3. In Figure E.3, every pair of $\mathcal{X}_i$'s are d-separated given the union of the boundary nodes $\mathcal{W}$, so the $\mathcal{X}_i$'s can be shown to be a conditional P-map reduction. Now the interior nodes X_1 and X_3 in $\mathcal{X}_2$ are not d-separated given the boundary nodes $\{X_2, X_5, X_{12}\}$. However, X_1 and X_3 are d-separated in the sub-graph $\mathcal{G}[\mathcal{X}_2]$. This is why enforcing the boundary nodes to be always observed does not help to find the d-separations of the sub-graphs of the cover elements $\mathcal{G}[\mathcal{X}_i]$ – it may be that only the d-separation itself appears in $\mathcal{I}(P)$.

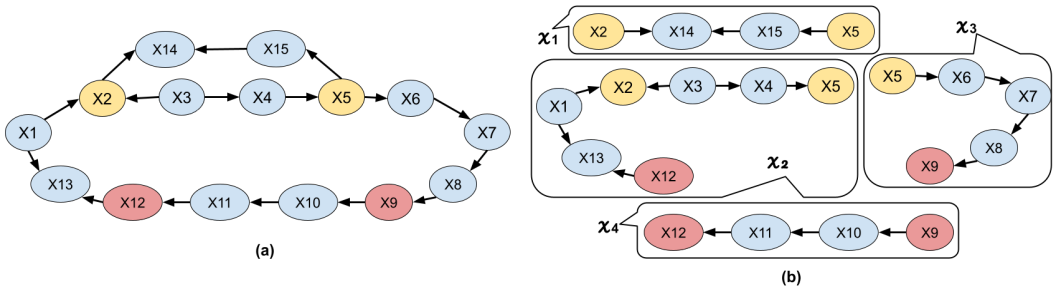

Figure 3: (a) The P-map $\mathcal{G}$ for variables $X_1, \ldots, X_{15}$. (b) The cover consisting of $\mathcal{X}_1 = \{X_5, X_{15}, X_{14}, X_2\}$, $\mathcal{X}_2 = \{X_{12}, X_{13}, X_1, \ldots, X_5\}$, $\mathcal{X}_3 = \{X_5, \ldots, X_9\}$, and $\mathcal{X}_4 = \{X_9, X_{10}, X_{11}, X_{12}\}$.

Example 1 (revisited)**.** *Let $\mathcal{W} = \{X_3\}$ and consider the vector $\boldsymbol{X} = [X_1, X_2, X_3, X_4, X_5]^\top$. Then $\boldsymbol{X}_{\{X_3\}} = [X_1, X_2, X_4, X_5]^\top$. The dependency matrix then equals*

$$\bar{D}_{\{X_3\}} = D_{\{X_3\}} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

which is already in a block-diagonal form. Hence, the P-map reduction consists of $\{X_1, X_2\} \cup \{X_3\}$, $\{X_4\} \cup \{X_3\}$, and $\{X_5\} \cup \{X_3\}$. Should, instead, the order $\boldsymbol{X} = [X_1, X_4, X_2, X_3, X_5]^\top$ was used, yielding $\boldsymbol{X}_{\{X_3\}} = [X_1, X_4, X_2, X_5]^\top$, then $\bar{D}_{\{X_3\}}$ would be obtained as above by using the following permutation matrix applied to the dependency matrix $D_{\{X_3\}}$:

$$D_{\{X_3\}} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Algorithm 8: The PC Algorithm

Input: A Covering set $\mathcal{X}_i$ and their joint probability distribution P
Output: An undirected graph

Form the complete undirected graph $\mathcal{G}_i$ over nodes $\mathcal{X}_i$;
$\mathrm{Sep}(X, Y) = \emptyset$ for all $X, Y \in \mathcal{X}_i$;
$m = 0$;
while *maximum node degree in* $\mathcal{G}_i$ *is greater than* m **do**
  for $X \in \mathcal{X}_i$
    for $Y \in \mathrm{Adj}(\mathcal{G}_i, X)$
      for $\mathcal{U} \subseteq \mathrm{Adj}(\mathcal{G}_i, X) \setminus \{Y\}$ *and* $|\, \mathcal{U} | = m$
        if $X \perp Y \mid \mathcal{U}$
          Remove the edge $X - Y$ from $\mathcal{G}_i$;
          $\mathrm{Sep}(X, Y) \leftarrow \mathcal{U}$;
  $m = m + 1$;
Orient the edges using the orientation rules in (Spirtes et al., 2000).

Lemma E.4. *(Based on (Pearl, 2009)) Consider random variables $\mathcal{X}$ with joint distribution P that admits a P-map $\mathcal{G}$. Vertices X and Y are not adjacent in $\mathcal{G}$ if and only if $X \perp Y \mid \mathcal{U}$ for $\mathcal{U} = \mathrm{Pa}_X$ (parents of X in $\mathcal{G}$) or Pa_Y (parents of Y in $\mathcal{G}$).*

Lemma E.5. *(Based on (Koller & Friedman, 2009)) Let $\mathcal{G}$ be a P-map of a distribution P and assume that X, Y and Z are a potential immorality, i.e., X and Y are not adjacent but both are adjacent with Z. Then X, Y, Z form an immorality, i.e., $X \to Z \leftarrow Y$ if and only if $X \not\perp Y \mid \mathcal{U}$ for any set $\mathcal{U} \ni Z$.*

Lemma E.6. *(Based on (Koller & Friedman, 2009)) Let $\mathcal{G}$ be a P-map of a distribution P, and assume that there exists three nodes X, Y, Z, where X and Y are adjacent with Z but with themselves, and the three do not form an immorality, i.e., $X \to Z \leftarrow Y$ is not in $\mathcal{G}$. If $\mathcal{U}$ is such that $X \perp Y \mid \mathcal{U}$, then $Z \in \mathcal{U}$.*

Proof of Proposition D.1 Let $\mathcal{W} = \mathrm{bd}(\mathcal{X}_i)$. Condition *(iii)* in Definition 3.1 follows the fact that $\mathcal{X}_i^o \perp \mathcal{X}_j^o \mid \mathcal{W}$ and the fact that two nodes are not adjacent in a P-map should they be conditionally independent. So it suffices to prove Condition *(ii)*. It is straightforward to show that $\mathcal{G}[\mathcal{X}_i]$ is faithful to $P[\mathcal{X}_i]$ for every $i = 1, \ldots, I$: the subgraph $\mathcal{G}[\mathcal{X}_i]$ is obtained by removing some nodes and edges from the P-map $\mathcal{G}$, which does not add a new path between two nodes; so nodes without a connecting path in $\mathcal{G}$ remains so in $\mathcal{G}[\mathcal{X}_i]$. Now we show that $\mathcal{G}[\mathcal{X}_i]$ is a conditional I-map for P. Consider the d-separation $\text{d-sep}_{\mathcal{G}[\mathcal{X}_i]}(\mathcal{Y}_1, \mathcal{Y}_2 \mid \mathcal{Y}_3)$, where $\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3 \subseteq \mathcal{X}_i$. Let $\mathcal{W}_v \subseteq \mathcal{W}$ denote the set of separator nodes that form a collider with a node in $\mathcal{Y}_1$ and a node in $\mathcal{Y}_2$ or are a descendent node of such a collider. Define $\mathcal{W}_n = \mathcal{W} \setminus \mathcal{W}_v$. We prove by contradiction that $\text{d-sep}_{\mathcal{G}}(\mathcal{Y}_1, \mathcal{Y}_2 \mid \mathcal{Y}_3 \cup \mathcal{W}_n)$, where $\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3 \subseteq \mathcal{X}_i$. Assume the contrary, implying that there is an active path $\mathcal{T}$ from a node $Y_1 \in \mathcal{Y}_1$ to a node $Y_2 \in \mathcal{Y}_2$ when observing $\mathcal{W}_n$. Path $\mathcal{T}$ cannot include any of the nodes $\mathcal{W}_n$ as they would block the path. Also, since observing $\mathcal{W}_n$ does not activate any collider, $\mathcal{T}$ must include a node $S \notin \mathcal{X}_i$ out of $\mathcal{X}_i$. On the other hand, the separator drives the cover elements independent, yielding $\mathcal{X}_i \perp S \mid \mathcal{W}$, meaning that the nodes in $\mathcal{W}$ block all paths such as $\mathcal{T}$ that leave $\mathcal{X}_i$ and have two end nodes in $\mathcal{X}_i$. Since $\mathcal{T}$ does not include $\mathcal{W}_n$, it includes some of the nodes in $\mathcal{W}_v$. Namely, path $\mathcal{T}$ leaves $\mathcal{X}_i$ from a node $W_1 \in \mathcal{W}_v$, reaches S and returns to $\mathcal{X}_i$ by another node $W_2 \in \mathcal{W}_v$. Hence, for $\mathcal{T}$ to be active, its edge adjacent to W_1 must be an outgoing edge and the same holds for W_2. However, then that part of path $\mathcal{T}$ with ends W_1 and W_2 that passes through S will have a collider which blocks the whole path $\mathcal{T}$, a contradiction, completing the proof. □

Proof of Proposition D.2 In view of Lemma E.4, if X_1 and X_2 are adjacent in $\mathcal{G}[\mathcal{X}_i]$, then they are not independent conditioned on any subset of other values, including those in $\mathcal{X}_i$. Hence, the dependence also reveals in $P[\mathcal{X}_i]$, implying the existence of the link in the P-map class PDAG of $P[\mathcal{X}_i]$. If X_1 and X_2 are not adjacent in $\mathcal{G}[\mathcal{X}_i]$, then they are independent conditioned on some subset $\mathcal{U} \in \mathcal{X}$. On the other hand, $\mathcal{X}_i^o \perp (\mathcal{X} \setminus \mathcal{X}_i) \mid \mathrm{bd}(\mathcal{X}_i)$, implying that $\mathrm{bd}(\mathcal{X}_i)$ blocks all paths between $\mathcal{X}_i^o$ and nodes other than $\mathcal{X}_i$. On the other hand, similar to the proof of Proposition D.1 it can be shown that the above independence also holds when we only condition on those boundary nodes $\mathcal{W}_n$ that do not form a collider with X_1 and X_2 and are not a descendent node that would activate such a collider, i.e., $\mathcal{X}_i^o \perp (\mathcal{X} \setminus \mathcal{X}_i) \mid \mathcal{W}_n$. Thus, $X_1 \perp X_2 \mid \mathcal{U}$ yields $X_1 \perp X_2 \mid (\mathcal{X}_i \cap \mathcal{U}) \cup \mathcal{W}_n$ as those nodes of $\mathcal{U}$ that are out of $\mathcal{X}_i$ and have an active path to X_1 or X_2, their path can be blocked by

observing $\mathcal{W}_n$. Hence, there X_1 and X_2 become independent also by conditioning on nodes that are only in $\mathcal{X}_i$. Therefore, in view of Lemma E.4, they will not be adjacent in the P-map class PDAG of $P[\mathcal{X}_i]$.

Now we prove the second part. Suppose that X_1 and X_2 form an immorality with another node $Z \in \mathcal{X}_i$ in $\mathcal{G}[\mathcal{X}_i]$, i.e., $X_1 \to Z \leftarrow X_2$, and that at least two of X_1 ,X_2, and Z are in $\mathcal{X}_i^o$. Then X_1 and X_2 are not d-separated in $\mathcal{G}[\mathcal{X}_i]$ given any $\mathcal{U} \subseteq \mathcal{X}_i$ that contains Z. This implies that X_1 and X_2 are not d-separated in $\mathcal{G}$ given any $\mathcal{U} \subseteq \mathcal{X}$ that contains Z as adding more edges and vertices to $\mathcal{G}[\mathcal{X}_i]$ does not make an already active path inactive. Now due to $\mathcal{G}$ being a P-map class PDAG for P, it holds that $X_1 \not\perp X_2 \mid \mathcal{U}$ for any $\mathcal{U} \subseteq \mathcal{X}$ that contains Z. On the other hand, based on what we proved earlier, X_1 and X_2 are connected to Z and are not adjacent with each other in the P-map PDAG class of $P[\mathcal{X}_i]$. Hence, in view of Lemma E.5, X_1 and X_2 form an immorality with Z in the P-map. Now suppose that X_1 and X_2 are not adjacent, both connected to $Z \in \mathcal{X}_i$, do not form an immorality in $\mathcal{G}[\mathcal{X}_i]$, and that again at least two of X_1 ,X_2, and Z are in $\mathcal{X}_i^o$. Clearly, the same holds in $\mathcal{G}$. In view of Lemma E.6, if $X_1 \perp X_2 \mid \mathcal{U}$ for some $\mathcal{U} \subseteq \mathcal{X}$, then $Z \in \mathcal{U}$. Thus, if $X_1 \perp X_2 \mid \mathcal{U}$ for some $\mathcal{U} \subseteq \mathcal{X}_i$, then $Z \in \mathcal{U}$, meaning that the condition holds also in $P[\mathcal{X}_i]$, which completes the proof according to Lemma E.5.

Now consider the triple X_1, X_2, and Z, where only one of them, say X_1, is in $\mathcal{X}_i^o$ and the other two are in $\mathrm{bd}(\mathcal{X}^i)$. Consider the case where the three nodes form the immorality $X_1 \to Z \leftarrow X_2$ in the P-map PDAG class of $P[\mathcal{X}_i]$. Then there exists a $\mathcal{U} \subseteq \mathcal{X}_i$ not including Z, such that $X_1 \perp X_2 \mid \mathcal{U}$, which implies that there is no active path between X_1 and X_2 that has a node out of $\mathcal{X}_i$. We prove by contradiction that X_2 and Z are adjacent in $\mathcal{G}$. Otherwise, there exists an active path $\mathcal{T}$ of length at least two between X_2 and Z regardless of whether any subset $\mathcal{U} \subseteq \mathcal{X}_i$ is observed. Therefore, every node in $\mathcal{T}$ is out of $\mathcal{X}_i$. Let $V \in \mathcal{T}$ be the node in $\mathcal{T}$ that is adjacent to Z. The direction of the edge between V and Z cannot be from Z to V, because then by observing both Z and the aforementioned $\mathcal{U}$, X_1 and X_2 will become d-separated, which is impossible. For the same reason, X_1 is linked to Z. Hence, X_1, Z, and V form the collider $X_1 \to Z \leftarrow V$, implying that $X_1 \not\perp V \mid Z$. This, however, contradicts equation 2. Hence, X_2 and Z are adjacent in $\mathcal{G}$. Then the immorality $X_1 \to Z \leftarrow X_2$ exists in $\mathcal{G}$ as well as otherwise, there cannot exist a $\mathcal{U} \subseteq \mathcal{X}_i$ not including Z such that $X_1 \perp X_2 \mid \mathcal{U}$, a contradiction. □

Proof of Lemma D.3 We prove by induction on the cardinality k of the cover, where $k = K_1, K_2 \ldots$. For both algorithms, the base case $k = K_1 > 1$ holds trivially. Assume that the result holds for $k = m$. Consider that iteration in the algorithms where the cover has cardinality m, denoted by $\{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ and let element $\mathcal{X}_i$ be the next cover that will be reduced. According to equation 2, $\mathcal{X}_i \perp \mathcal{X}_j \mid \mathrm{bd}(\mathcal{X}_i)$. This implies that the boundary nodes of $\mathcal{X}_i$, block every path that connect the interior nodes of $\mathcal{X}_i$ to other elements of the cover. In Algorithm 3, $\mathcal{X}_i$ will be reduced to a cover $\{\mathcal{W} \cup \mathcal{C}^i_{\mathcal{W}}\}_{i=1}^I$ where $\mathcal{C}^i_{\mathcal{W}} \perp \mathcal{C}^j_{\mathcal{W}} \mid \mathcal{W}$ for all $i \neq j$. Now consider an arbitrary $\mathcal{C}_i$. Should $\mathrm{bd}(\mathcal{X}_i) \subseteq \mathcal{W}$, then $\mathcal{C}^i_{\mathcal{W}} \perp \mathcal{X}_j | \mathcal{W}$ for all j. Otherwise, some of the nodes in $\mathrm{bd}(\mathcal{X}_i)$ are in $\cup_{j \neq i} \mathcal{C}^j_{\mathcal{W}}$, and hence, are d-separated from $\mathcal{C}^i_{\mathcal{W}}$ after observing $\mathcal{W}$. In other words, $\mathcal{W}$ either directly or indirectly blocks all the paths from $\mathcal{C}^i_{\mathcal{W}}$ to $\mathcal{X}_j$ for every $j \neq i$. This is because observing $\mathcal{W}$ does not activated any collider that would in turn activate a path between $\mathcal{X}_i$ and $\mathcal{X}_j$ (every node in $\mathcal{X}_i$ that is adjacent to another $\mathcal{X}_j$ is included in $\mathrm{bd}(\mathcal{X}_i)$ as otherwise equation 2 is violated). This completes the proof for Algorithm 3. The proof for Algorithm 5 is similar. □

E.1 Efficient conditional χ^2 tests

Assume a complete discrete dataset with N observations of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. For each variable X_v, let r_v denote its number of categories, and relabel its observed values as

$$x_{tv} \in \{0, \ldots, r_v - 1\}, \qquad t = 1, \ldots, N, \;\; v = 1, \ldots, n.$$

We describe an exact conditional independence test for

$$X_i \perp X_j \mid \mathcal{W}, \qquad \mathcal{W} \subseteq \mathcal{X} \setminus \{X_i, X_j\},$$

that avoids one-hot encoding and is suitable for blocked CPU/GPU histogram implementations.

Let

$$\mathcal{W} = \{X_{w_1}, \ldots, X_{w_q}\}.$$

For each observation t, define its conditioning-state vector

$$\mathbf{z}_t = (x_{tw_1}, \ldots, x_{tw_q}).$$

Let

$$\mathbf{z}^{(0)}, \ldots, \mathbf{z}^{(m-1)}$$

be the distinct *observed* values among $\mathbf{z}_1, \ldots, \mathbf{z}_N$, and define the stratum index

$$s_t = s \quad \Longleftrightarrow \quad \mathbf{z}_t = \mathbf{z}^{(s)}, \qquad s \in \{0, \ldots, m-1\}.$$

Thus $m \leq N$, so only observed strata are materialized, rather than all $\prod_{X_u \in \mathcal{W}} r_u$ possible configurations.

The size of stratum s is

$$n_s = \sum_{t=1}^{N} \mathbf{1}\{s_t = s\}.$$

For each variable $X_i \notin \mathcal{W}$, define its variable–stratum count matrix

$$C^{(i|\mathcal{W})} \in \mathbb{N}^{r_i \times m}, \qquad C^{(i|\mathcal{W})}_{as} = \sum_{t=1}^{N} \mathbf{1}\{x_{ti} = a\}\mathbf{1}\{s_t = s\}.$$

Equivalently, in vectorized form, if

$$d_t^{(i|\mathcal{W})} = x_{ti} + r_i s_t,$$

then

$$\mathrm{vec}\Big(C^{(i|\mathcal{W})}\Big) = \mathrm{bincount}\Big(d^{(i|\mathcal{W})},\, r_i m\Big),$$

where $\mathrm{bincount}(\cdot, L)$ denotes the length-L histogram of a vector of integer bin IDs, and $\mathrm{vec}(\cdot)$ uses the flattening rule $a + r_i s$.

For a pair (X_i, X_j) with $X_i, X_j \notin \mathcal{W}$, define the conditional contingency tensor

$$O^{(ij|\mathcal{W})} \in \mathbb{N}^{r_i \times r_j \times m}$$

by

$$O^{(ij|\mathcal{W})}_{abs} = \sum_{t=1}^{N} \mathbf{1}\{x_{ti} = a\}\mathbf{1}\{x_{tj} = b\}\mathbf{1}\{s_t = s\}.$$

Equivalently, if

$$b_t^{(ij|\mathcal{W})} = x_{ti} + r_i x_{tj} + r_i r_j s_t,$$

then

$$\mathrm{vec}\Big(O^{(ij|\mathcal{W})}\Big) = \mathrm{bincount}\Big(b^{(ij|\mathcal{W})},\, r_i r_j m\Big),$$

with flattening rule $a + r_i b + r_i r_j s$.

The conditional marginals are

$$U^{(i|\mathcal{W})}_{as} = C^{(i|\mathcal{W})}_{as}, \qquad V^{(j|\mathcal{W})}_{bs} = C^{(j|\mathcal{W})}_{bs}.$$

Hence, under the null hypothesis $X_i \perp X_j \mid \mathcal{W}$, the expected counts are

$$E^{(ij|\mathcal{W})}_{abs} = \frac{U^{(i|\mathcal{W})}_{as}\, V^{(j|\mathcal{W})}_{bs}}{n_s} = \frac{C^{(i|\mathcal{W})}_{as}\, C^{(j|\mathcal{W})}_{bs}}{n_s}, \qquad n_s > 0.$$

To avoid divisions by zero, define the active category sets in stratum s by

$$A_s^{(i)} = \{a : C^{(i|\mathcal{W})}_{as} > 0\}, \qquad B_s^{(j)} = \{b : C^{(j|\mathcal{W})}_{bs} > 0\}.$$

Only cells with $a \in A_s^{(i)}$ and $b \in B_s^{(j)}$ have positive expected count. The conditional Pearson statistic is

$$\chi^2_{ij|\mathcal{W}} = \sum_{s=0}^{m-1} \sum_{a \in A_s^{(i)}} \sum_{b \in B_s^{(j)}} \frac{\Big(O^{(ij|\mathcal{W})}_{abs} - E^{(ij|\mathcal{W})}_{abs}\Big)^2}{E^{(ij|\mathcal{W})}_{abs}}.$$

Strata with $|A_s^{(i)}| \le 1$ or $|B_s^{(j)}| \le 1$ contribute zero.

The corresponding degrees of freedom are

$$\nu_{ij|\mathcal{W}} = \sum_{s=0}^{m-1} \big(|A_s^{(i)}| - 1\big)\big(|B_s^{(j)}| - 1\big)\mathbf{1}\{|A_s^{(i)}| \ge 2,\ |B_s^{(j)}| \ge 2\}.$$

The asymptotic p-value is then

$$p_{ij|\mathcal{W}} = 1 - F_{\chi^2_{\nu_{ij|\mathcal{W}}}}\left(\chi^2_{ij|\mathcal{W}}\right),$$

where $F_{\chi^2_\nu}$ is the cdf of the chi-square distribution with ν degrees of freedom.

Accordingly, the dependency matrix for the variables outside $\mathcal{W}$ is defined by

$$D_{\mathcal{W}}(i,j) = \mathbf{1}\{p_{ij|\mathcal{W}} \le \alpha\}, \qquad D_{\mathcal{W}}(i,i) = 1,$$

where $D_{\mathcal{W}}(i,j) = 1$ indicates rejection of the conditional independence $X_i \perp X_j \mid \mathcal{W}$ at significance level α.

For a fixed conditioning set $\mathcal{W}$, the quantities s_t, n_s, and $C^{(i|\mathcal{W})}$ are computed once and reused for all pairs (i,j). Thus the only pair-specific computation is the histogram $O^{(ij|\mathcal{W})}$, which can be carried out in blocks of pairs by batched `bincount`/histogram kernels. This avoids the memory blow-up of one-hot encoding and is the preferred exact implementation for large n.

F CURRENT CONTROLLED RERUN: ADDITIONAL TABLES

The full 12-algorithm runtime table for $n = 100$ appears in Table 1 in the main text. The remaining runtime and accuracy tables follow here. Completed runtimes are in seconds. Bounds in hours denote verified timeouts; NR means no verified result, INV an invalid graph, and ERR an execution error. Accuracy is raw SHD / adjacency F1. D-T-PC, D-S-PC, and D-T-Tabu denote the distributed tensor-PC, scalar-PC, and tensor-Tabu variants.

Table 3: Accuracy (SHD / adjacency F1), $n = 100$ samples. TO denotes a timeout with no accuracy score; NR denotes no verified result. INV and ERR are invalid outputs and execution errors.

Algorithm	CANCER	EARTH	SURVEY	ASIA	CHILD	INSUR.	WATER	MILDEW
PC	4/0.000	3/0.571	6/0.500	9/0.462	20/0.649	47/0.466	71/0.400	NR
PC-stable	4/0.000	3/0.571	6/0.500	9/0.462	19/0.649	48/0.444	70/0.386	NR
Parallel PC	4/0.000	NR	6/0.500	NR	19/0.649	NR	70/0.386	NR
Tensor PC	4/0.000	NR	6/0.500	NR	19/0.649	NR	70/0.386	NR
D-T-PC	4/0.000	3/0.571	6/0.500	9/0.462	18/0.718	48/0.444	70/0.386	45/0.492
D-S-PC	4/0.000	3/0.571	6/0.500	9/0.462	18/0.718	48/0.444	70/0.386	45/0.492
D-T-Tabu	4/0.000	4/0.667	6/0.286	8/0.545	20/0.611	50/0.412	63/0.341	46/0.264
MMPC	4/0.000	5/0.571	6/0.500	10/0.462	22/0.595	51/0.486	66/0.382	46/0.562
Parallel MMPC	4/0.000	5/0.571	6/0.500	10/0.462	22/0.595	51/0.486	66/0.382	NR
HC	4/0.000	4/0.857	7/0.250	7/0.571	23/0.585	53/0.411	61/0.429	47/0.291
Tabu	4/0.000	4/0.857	7/0.250	5/0.714	23/0.585	53/0.411	61/0.429	47/0.291
GES	4/0.400	4/0.750	7/0.250	5/0.714	23/0.585	55/0.421	79/0.408	47/0.291

Algorithm	ALARM	HEPAR	WIN95	PATH	ANDES	DIAB.	PIGS	LINK
PC	NR	NR	NR	NR	NR	567/0.593	NR	TO
PC-stable	NR	NR	NR	NR	NR	547/0.597	NR	TO
Parallel PC	44/0.464	NR	120/0.449	NR	343/0.437	NR	549/0.508	NR
Tensor PC	44/0.464	NR	120/0.449	NR	343/0.437	NR	549/0.508	NR
D-T-PC	44/0.429	129/0.234	121/0.447	218/0.313	344/0.433	TO	549/0.508	NR
D-S-PC	44/0.429	129/0.234	121/0.447	218/0.313	TO	TO	TO	NR
D-T-Tabu	49/0.563	INV	121/0.366	INV	TO	TO	TO	NR
MMPC	52/0.500	138/0.250	137/0.448	221/0.304	409/0.423	NR	665/0.554	NR
Parallel MMPC	NR	NR	NR	221/0.304	NR	NR	NR	TO
HC	49/0.590	137/0.300	120/0.464	215/0.275	TO	NR	TO	NR
Tabu	49/0.590	137/0.300	120/0.464	215/0.275	TO	NR	TO	NR
GES	52/0.617	156/0.302	139/0.394	254/0.257	TO	NR	TO	NR

Table 4: Runtime (seconds for completed runs), $n = 1,000$ samples. Hour bounds denote confirmed timeouts; NR denotes no verified result. INV and ERR are invalid outputs and execution errors.

Algorithm	CANCER	EARTH	SURVEY	ASIA	CHILD	INSUR.	WATER	MILDEW
PC	0.0708	0.0941	0.0315	0.121	3.96	3.46	0.471	NR
PC-stable	0.0716	0.0882	0.0225	0.0895	5.62	4.75	0.487	NR
Parallel PC	0.285	NR	0.238	NR	6.32	NR	0.724	NR
Tensor PC	0.0212	NR	0.0153	NR	0.521	NR	0.394	NR
D-T-PC	0.0569	0.0397	0.0908	0.0959	1.12	0.962	0.677	7.3
D-S-PC	0.375	0.4	0.362	0.396	10.9	4.51	2.71	224
D-T-Tabu	10.9	9.06	6.64	39.5	INV	1150	783	1015
MMPC	0.0781	0.125	0.0411	0.124	10.5	7.57	1.15	85.1
Parallel MMPC	0.208	0.23	0.155	0.293	8.39	1.82	0.469	NR
HC	3.25	2.91	3.48	22	289	830	854	951
Tabu	10.6	8.87	14.1	39.5	466	1178	1174	2004
GES	3.25	3.41	2.99	15.4	263	583	1336	946

Algorithm	ALARM	HEPAR	WIN95	PATH	ANDES	DIAB.	PIGS	LINK
PC	NR	NR	NR	NR	NR	11404	> 12 h	NR
PC-stable	NR	NR	NR	NR	NR	13643	> 12 h	NR
Parallel PC	2.3	NR	2.31	NR	34.2	NR	NR	NR
Tensor PC	0.628	NR	1.27	NR	6.67	NR	NR	NR
D-T-PC	4.73	68	219	2043	20811	7968	ERR	NR
D-S-PC	49.8	945	2119	> 12 h	> 12 h	> 12 h	NR	NR
D-T-Tabu	INV	2017	INV	> 12 h	> 12 h	> 12 h	NR	NR
MMPC	9.61	15.9	21.9	> 12 h	NR	NR	3097	NR
Parallel MMPC	NR	NR	NR	> 12 h	NR	1022	1070	1396
HC	2088	11002	20681	> 12 h	> 12 h	NR	> 12 h	NR
Tabu	2882	14061	22493	> 12 h	> 12 h	NR	> 12 h	NR
GES	2078	10209	NR	> 12 h	> 12 h	NR	> 12 h	NR

Table 5: Accuracy (SHD / adjacency F1), $n = 1,000$ samples. TO denotes a timeout with no accuracy score; NR denotes no verified result. INV and ERR are invalid outputs and execution errors.

Algorithm	CANCER	EARTH	SURVEY	ASIA	CHILD	INSUR.	WATER	MILDEW
PC	2/0.857	1/0.889	6/0.500	5/0.824	13/0.917	45/0.729	59/0.460	NR
PC-stable	2/0.857	1/0.889	6/0.500	5/0.824	15/0.917	40/0.732	59/0.460	NR
Parallel PC	2/0.857	NR	6/0.500	NR	15/0.917	NR	59/0.460	NR
Tensor PC	2/0.857	NR	6/0.500	NR	15/0.917	NR	59/0.460	NR
D-T-PC	2/0.857	1/0.889	6/0.500	5/0.824	16/0.917	40/0.732	59/0.460	31/0.734
D-S-PC	2/0.857	1/0.889	6/0.500	5/0.824	16/0.917	40/0.732	59/0.460	31/0.734
D-T-Tabu	4/0.857	4/1.000	6/0.500	7/0.750	INV	42/0.698	60/0.465	46/0.606
MMPC	4/0.857	5/0.889	6/0.500	8/0.824	15/0.894	43/0.700	61/0.460	47/0.730
Parallel MMPC	4/0.857	5/0.889	6/0.500	8/0.824	15/0.894	43/0.700	61/0.460	NR
HC	4/0.857	4/1.000	6/0.500	7/0.750	8/0.913	45/0.682	53/0.511	49/0.580
Tabu	4/0.857	4/1.000	6/0.500	7/0.750	6/0.936	42/0.698	53/0.511	50/0.571
GES	4/0.857	4/1.000	6/0.500	3/0.857	11/0.889	47/0.617	68/0.451	50/0.571

Algorithm	ALARM	HEPAR	WIN95	PATH	ANDES	DIAB.	PIGS	LINK
PC	NR	NR	NR	NR	NR	587/0.639	TO	NR
PC-stable	NR	NR	NR	NR	NR	536/0.648	TO	NR
Parallel PC	20/0.833	NR	96/0.677	NR	255/0.745	NR	NR	NR
Tensor PC	20/0.833	NR	96/0.677	NR	255/0.745	NR	NR	NR
D-T-PC	21/0.833	113/0.447	89/0.667	227/0.419	247/0.745	536/0.648	ERR	NR
D-S-PC	21/0.833	113/0.447	89/0.667	TO	TO	TO	NR	NR
D-T-Tabu	INV	115/0.457	INV	TO	TO	TO	NR	NR
MMPC	45/0.843	131/0.442	127/0.647	TO	NR	NR	611/0.936	NR
Parallel MMPC	NR	NR	NR	TO	NR	651/0.609	611/0.936	1205/0.531
HC	31/0.848	112/0.557	81/0.704	TO	TO	NR	TO	NR
Tabu	26/0.857	112/0.565	81/0.704	TO	TO	NR	TO	NR
GES	29/0.831	90/0.571	NR	TO	TO	NR	TO	NR

Table 6: Runtime (seconds for completed runs), $n = 10,000$ samples. Hour bounds denote confirmed timeouts; NR denotes no verified result. INV and ERR are invalid outputs and execution errors.

Algorithm	CANCER	EARTH	SURVEY	ASIA	CHILD	INSUR.	WATER	MILDEW
PC	0.173	0.206	0.167	0.179	NR	NR	NR	NR
PC-stable	0.17	0.208	0.167	0.224	NR	NR	NR	NR
Parallel PC	0.547	NR	0.616	NR	26.9	NR	4.01	NR
Tensor PC	0.13	NR	0.119	NR	3.63	NR	1.54	NR
D-T-PC	0.132	0.176	0.18	0.176	4.05	8.18	2.84	44.3
D-S-PC	0.484	0.562	0.783	0.788	28.1	62.7	9.93	1228
D-T-Tabu	154	89.8	249	617	INV	INV	INV	INV
MMPC	0.201	0.185	0.182	0.3	284	407	11.3	4217
Parallel MMPC	0.33	0.347	0.41	0.366	259	134	3.57	1153
HC	41.1	28.4	89.8	239	3315	10940	12752	14538
Tabu	153	90.1	250	621	5065	15861	15412	23135
GES	53.6	38	88.7	209	2800	10415	21108	15494

Algorithm	ALARM	HEPAR	WIN95	PATH	ANDES	DIAB.	PIGS	LINK
PC	NR	NR	NR	$> 12\,\mathrm{h}$	NR	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR
PC-stable	NR	NR	NR	$> 12\,\mathrm{h}$	NR	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR
Parallel PC	10.5	NR	28	NR	INV	NR	NR	NR
Tensor PC	3.34	NR	16	NR	INV	NR	NR	NR
D-T-PC	30	416	589	ERR	INV	ERR	NR	NR
D-S-PC	193	2904	2451	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR	NR	NR
D-T-Tabu	19660	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR	NR	NR
MMPC	112	926	3422	$> 12\,\mathrm{h}$	1362	NR	$> 12\,\mathrm{h}$	NR
Parallel MMPC	NR	535	1561	$> 12\,\mathrm{h}$	918	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$
HC	23430	NR	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR	$> 12\,\mathrm{h}$	NR
Tabu	33312	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR	$> 12\,\mathrm{h}$	NR
GES	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	$> 12\,\mathrm{h}$	NR	NR	NR

Table 7: Accuracy (SHD / adjacency F1), $n = 10,000$ samples. TO denotes a timeout with no accuracy score; NR denotes no verified result. INV and ERR are invalid outputs and execution errors.

Algorithm	CANCER	EARTH	SURVEY	ASIA	CHILD	INSUR.	WATER	MILDEW
PC	2/1.000	0/1.000	2/1.000	2/1.000	NR	NR	NR	NR
PC-stable	2/1.000	0/1.000	2/1.000	2/1.000	NR	NR	NR	NR
Parallel PC	2/1.000	NR	2/1.000	NR	11/1.000	NR	48/0.612	NR
Tensor PC	2/1.000	NR	2/1.000	NR	11/1.000	NR	48/0.612	NR
D-T-PC	2/1.000	0/1.000	2/1.000	2/1.000	2/1.000	27/0.884	48/0.612	14/0.894
D-S-PC	2/1.000	0/1.000	2/1.000	2/1.000	2/1.000	27/0.884	48/0.612	14/0.894
D-T-Tabu	0/1.000	0/1.000	6/1.000	4/0.750	INV	INV	INV	INV
MMPC	4/1.000	4/1.000	6/1.000	5/1.000	13/1.000	40/0.844	60/0.680	50/0.780
Parallel MMPC	4/1.000	4/1.000	6/1.000	5/1.000	13/1.000	40/0.844	60/0.680	50/0.780
HC	4/1.000	0/1.000	0/1.000	7/0.706	3/0.960	53/0.708	48/0.641	40/0.691
Tabu	0/1.000	0/1.000	6/1.000	4/0.750	3/0.960	48/0.758	48/0.641	38/0.716
GES	0/1.000	0/1.000	0/1.000	4/0.750	11/0.894	28/0.817	79/0.500	32/0.725

Algorithm	ALARM	HEPAR	WIN95	PATH	ANDES	DIAB.	PIGS	LINK
PC	NR	NR	NR	TO	NR	TO	TO	NR
PC-stable	NR	NR	NR	TO	NR	TO	TO	NR
Parallel PC	3/0.978	NR	52/0.836	NR	INV	NR	NR	NR
Tensor PC	3/0.978	NR	52/0.836	NR	INV	NR	NR	NR
D-T-PC	3/0.978	90/0.728	52/0.836	ERR	INV	ERR	NR	NR
D-S-PC	3/0.978	90/0.728	52/0.836	TO	TO	NR	NR	NR
D-T-Tabu	22/0.955	TO	TO	TO	TO	NR	NR	NR
MMPC	42/0.955	125/0.713	118/0.808	TO	349/0.889	NR	TO	NR
Parallel MMPC	NR	125/0.713	118/0.808	TO	349/0.889	TO	TO	TO
HC	14/0.903	NR	TO	TO	TO	NR	TO	NR
Tabu	13/0.913	TO	TO	TO	TO	NR	TO	NR
GES	TO	TO	TO	TO	TO	NR	NR	NR

F.1 EXPLORATORY PAIRED CHECKS

Only valid completed pairs are used. Runtime checks use paired log times; SHD and F1 use raw differences. Two-sided Wilcoxon values are unadjusted and exploratory.

Table 8: Exploratory unadjusted two-sided paired p-values for D-Tensor-PC.

Baseline	Pairs	Log runtime	SHD	F1
PC	19	0.312	0.293	0.893
Parallel PC	21	0.374	0.389	0.345
Tensor PC	21	9.54e-07	0.389	0.345
D-S-PC	34	2.33e-10	1	1