Journal of Machine Learning Research 23 (2022) 1-28 Submitted 11/20; Revised 5/22; Published 11/22

Improving Bayesian Network Structure Learning in the Presence of Measurement Error

Yang Liu[1] YANGLIU@QMUL.AC.UK
Anthony C. Constantinou[1, 2] A.CONSTANTINOU@QMUL.AC.UK
Zhigao Guo[1] ZHIGAO.GUO@QMUL.AC.UK
[1] *School of Electronic Engineering and Computer Science*
Queen Mary University of London
London, E1 4NS, UK
[2] *The Alan Turing Institute*
London, NW1 2DB, UK

Editor: David Jensen

Abstract

Structure learning algorithms that learn the graph of a Bayesian network from observational data often do so by assuming the data correctly reflect the true distribution of the variables. However, this assumption does not hold in the presence of measurement error, which can lead to spurious edges. This is one of the reasons why the synthetic performance of these algorithms often overestimates real-world performance. This paper describes a heuristic algorithm that can be added as an additional learning phase at the end of any structure learning algorithm, and serves as a correction learning phase that removes potential false positive edges. The results show that the proposed correction algorithm successfully improves the graphical score of five well-established structure learning algorithms spanning different classes of learning in the presence of measurement error.

Keywords: causal discovery, data noise, directed acyclic graph, measurement error, probabilistic graphical models

1. Introduction

A Bayesian Network (BN) is a probabilistic graphic model that captures causal or conditional independence relationships between variables via a directed acyclic graph (DAG). Learning BNs from observational data is recognised as a challenging problem that has received increasing attention during the past few decades. Various algorithms have been proposed to tackle this problem and are categorised into constraint-based, score-based and hybrid learning algorithms.

The PC algorithm (Spirtes et al., 2000) is one of the earliest proposed constraint-based algorithms which attempts to recover the Completed Partial Directed Acyclic Graph (CPDAG) of the underlying true causal graph by performing conditional independence tests between variables. Many other algorithms are derived from PC, including MMPC (Tsamardinos et al., 2003) which can handle thousands of variables via sequentially choosing the variable with the maximum association with the target variable into its parents and children set, PC-fdr (Li and Wang, 2009) which controls the false discovery rate of the skeleton of

the learned graph under a user-specified level of false discovery rate at the limit of large sample sizes and PC-stable (Colombo and Maathuis, 2014) which resolves the issue of PC's output being dependent on the order of variables as they appear in the data. The GES algorithm (Chickering, 2002), on the other hand, is a well-established score-based algorithm that searches for the optimal CPDAG over two phases. In phase I, GES greedily adds edges that maximise the Bayesian score, whereas in phase II, it greedily removes edges that maximise the Bayesian score. The ILP algorithm (Cussens, 2011) is another well-established algorithm that tackles the structure learning problem with the integer linear programming approach. Lastly, hybrid learning algorithms combine both classes of learning, constraint-based and score-based, and include the MMHC algorithm (Tsamardinos et al., 2006) that combines MMPC with hill-climbing search, and the H2PC algorithm (Gasse et al., 2014) that combines HPC (Gasse et al., 2014) with hill-climbing search.

Most of these algorithms assume that their input data are accurately sampled from the true distributions. However, this assumption is often not true when working with real-world data. The assumption of an underlying measurement error has only recently attracted attention in terms of its effect on BN structure learning. Scheines et al (Scheines and Ramsey, 2016) studied the effect of Gaussian measurement error on score-based FGES (Ramsey et al., 2017) and showed that even minor levels of measurement error can considerably deteriorate its accuracy. Zhang et al (Zhang et al., 2018) investigated the linear non-Gaussian models in the presence of measurement error and presented four conditions that make the underlying structure identifiable from the observed variables that incorporate measurement error. Lastly, Blom et al (Blom et al., 2018) proposed a method to estimate the upper bound of the variance of measurement error in linear Gaussian models, and used this bound as a correction of conditional independence tests during constraint-based learning.

Traditionally, measurement error is generated and modelled under the assumption of Normally distributed and continuous data (Bollinger and van Hasselt, 2017), although various other types of synthetic noise have recently been investigated with discrete variables (Constantinou et al., 2021). In this paper, we assume the data are discrete, and that variables with measurement error are children of their underlying error-free version, and not the actual parents of other variables, essentially making them independent of other variables in the graph given their error-free version. We propose a heuristic score-based correction method called the Spurious Edge Detection (SED) algorithm which aims to identify and remove potential false positive (FP) edges learned by other structure learning algorithms, often in the presence of measurement error. The remainder of the paper is organised as follows: the terminology and underlying assumptions are described in Section 2, Section 3 illustrates the impact of measurement error on structure learning, Section 4 describes the correction algorithm, Section 5 presents the results, and we provide our conclusions along with future research directions in Section 6.

2. Preliminaries

This section presents the preliminaries and the necessary terminology and assumptions. We assume that the data are categorical and that every variable present in the data may be subject to measurement error. We refer to variables with measurement error as *noisy variables* and to variables without measurement error as *error-free variables*. Each noisy

variable is assumed to be derived from its error-free version which is not present in the data. We denote the error-free variable as V_i where i represents the index of the error-free variable, and its corresponding noisy variable as V_i^*. When a variable is error-free, V_i is observed in the data, otherwise its noisy version V_i^* would be observed. We refer to *observed variable* as the one that is observed in the data. We use lowercase letters to represent the assignment of states where v_i^l denotes the lth state of variable V_i or its corresponding V_i^*. We define three types of graphs, i.e., error-free graph, noisy graph and learned graph based on different involved variables and edges.

- *Error-free graph* $G(\boldsymbol{V}, \boldsymbol{E})$ is composed of the error-free variables $\boldsymbol{V}$ and edges $\boldsymbol{E}$ between $\boldsymbol{V}$, and represents the true graph of the error-free variables $\boldsymbol{V}$.

- *Noisy graph* $G^*\left(\boldsymbol{V}^{(*)}, \boldsymbol{E}^{(*)}\right)$ is composed of both error-free and noisy variables $\boldsymbol{V}^{(*)} = \boldsymbol{V} \cup \boldsymbol{V}^*$ and edges $\boldsymbol{E}^{(*)}$ between $\boldsymbol{V}^{(*)}$, and represents the true graph of both the error-free and noisy variables $\boldsymbol{V}^{(*)} = \boldsymbol{V} \cup \boldsymbol{V}^*$.

- *Learned graph* $G^l(\boldsymbol{V}, \boldsymbol{E})$ is composed of error-free variables $\boldsymbol{V}$ and edges $\boldsymbol{E}$ between $\boldsymbol{V}$, and represents the graph learned from observational data.

We assume that all graphs are *Directed Acyclic Graphs* (DAGs) and that the observed data are sampled from the observed variables that make up the noisy graph, independently and identically. Note that the observed data and learned graphs are presented using the error-free variable names, since the algorithms are not given any information about which variables incorporate measurement error.

Given a DAG G, if there is an edge $V_j \to V_i$, then V_j is the *parent* of V_i and V_i is the *child* of V_j. A node is the *neighbour* of V_i if it is either the parent or the child of V_i. A *path* is a sequence of distinct nodes in which each pair of adjacent nodes are neighbours in the graph. A *directed path* denotes that every node in the sequence is the parent of the following node. If there is a directed path from V_i to V_j, then V_j is a *descendant* of V_i, and V_i is an *ancestor* of V_j. A node V_i is its own descendant since there is an empty path from V_i to V_i. Given a graph G, a node V_i is a *collider* in path p, if two parents of V_i are adjacent to V_i in path p.

Based on the above definition, two nodes V_i and V_j in DAG G are *d-separated* given a node set $\boldsymbol{S}$ if there is no path p between V_i and V_j such that (i) every collider in p has a descendant in $\boldsymbol{S}$, and (ii) no other nodes on p are in $\boldsymbol{S}$ (Spirtes et al., 2000). If V_i and V_j are not d-separated given $\boldsymbol{S}$, then V_i and V_j are *d-connected* given $\boldsymbol{S}$. We refer to these two relationships as *d-separation* and *d-connection*. If two DAGs G_1 and G_2 contain the same set of d-separation relationships, we say that G_1 and G_2 are *Markov Equivalent* and in the same *Markov Equivalence Class*. We define *Partially Directed Acyclic Graph* (PDAG) as a graph with both directed and undirected edges but without directed cycles. A DAG is a *consistent DAG extension* of a PDAG if it has the same set of edges with the same orientations on the directed edges of that PDAG and the same set of v-structures (Dor and Tarsi, 1992). A PDAG can be converted into a CPDAG if it admits at least one consistent DAG extension.

The assumption that a noisy variable V_i^* has only one parent, where this parent represents its error-free version V_i, produces the following *Independence rule*:

Independence rule: In the presence of measurement error, a noisy variable V_i^* is independent of other variables conditional on its error-free version V_i.

Figure 1 presents a simple example that illustrates the relationship between error-free and noisy variables given the Independence rule, where each V_i^* becomes independent of the remaining nodes given its corresponding error-free parent V_i. Moreover, if the error-free variable V_i has value v_i^l, its corresponding noisy version will be subject to an error rate α_i^l where

$$\alpha_i^l = 1 - P\left(V_i^* = v_i^l \mid V_i = v_i^l\right) \tag{1}$$

In other words, α_i^l represents the rate of observing a value for V_i^* that is not equal to the true value v_i^l of V_i. Note that it is possible for different states of V_i to be subject to varying error rates α_i^l. We denote the error rate α_i of variable V_i^* in terms of its maximum error rate amongst all states in V_i, i.e., $\alpha_i = \max_l \alpha_i^l$.

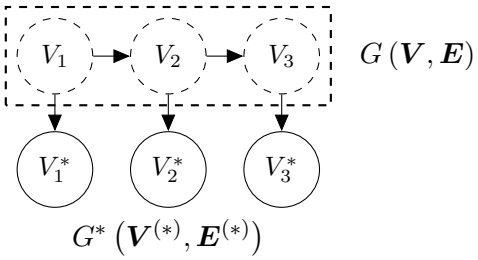


Figure 1: A hypothetical graph that illustrates the relationship between the error-free variables $\boldsymbol{V}$ and the corresponding noisy variables $\boldsymbol{V}^*$ given the Independence rule, where a noisy variable V_i^* becomes independent of other variables in G^* given V_i.

This paper also adopts the following widely used assumptions (Spirtes et al., 2000). These assumptions are applied to both error-free and noisy graphs:

(i) **Markov assumption**: Given a directed acyclic graph G over a variable set $\boldsymbol{V}$, every variable in $\boldsymbol{V}$ is independent of its non-descendants conditional on its parents.

(ii) **Causal Faithfulness assumption**: Given a directed acyclic graph G over a variable set $\boldsymbol{V}$, a probability distribution $P(\boldsymbol{V})$ is faithful to G if and only if the conditional independence relationships in $P(\boldsymbol{V})$ are exactly the same as the independence relationships inferred by *d-separation criterion* (Spirtes et al., 2000) from G.

(iii) **Causal Sufficiency assumption**: There are no unmeasured variables acting as a common cause of any two or more observed variables, and there is no selection bias.

3. The impact of measurement error on structure learning

This section illustrates that measurement error generally causes the structure learning algorithms to produce a higher number of spurious edges that tend to lead to a greater number of 3-vertex cliques, compared to the true number of such cliques in the ground truth graph. A clique is a set of nodes where each pair of nodes in the clique is adjacent. We first explain

why this phenomenon occurs in theory, and then present the effect in practise by illustrating the empirical effect of measurement error on algorithms spanning all three classes of learning. Given the Causal Faithfulness assumption, the dependencies between variables are consistent with those entailed by applying d-separation rules on the BN. Therefore, we restrict the description about the effect of measurement error on d-connections and d-separations. For the unconditional (i.e., marginal (in)dependence) case, we derive the Proposition 1.

Proposition 1 *The d-connection and d-separation relationships between two error-free variables V_1 and V_2 in a noisy graph G^* are consistent with the d-connection and d-separation relationships of their corresponding noisy versions V_1^* and V_2^*, given the Independence rule.*

Proof

1. When V_1 and V_2 are d-separated, this implies that there is either no path or at least one collider exists in every path between V_1 and V_2 in G^*. Given Independence rule, the only neighbours of V_1^* and V_2^* are V_1 and V_2 who serve as their respective error-free parents. Thus, when there is no path or at least one collider in every path between V_1^* and V_2^*, then V_1^* and V_2^* are d-separated.

2. When V_1 and V_2 are d-connected, there must be a path p that does not contain a collider between V_1 and V_2. Given Independence rule, V_1 and V_2 are the respective parents of V_1^* and V_2^*. Thus, by combining $V_1^* \leftarrow V_1, p$ and $V_2 \rightarrow V_2^*$, we can get a path that does not contain a collider between V_1^* and V_2^*, which would make V_1^* and V_2^* d-connected.

■

According to Proposition 1, the unconditional (in)dependence relationship between noisy variables should be consistent with the unconditional (in)dependence relationship of their corresponding error-free variables given the Causal Faithfulness assumption. However, the conditional independence between error-free variables may not always hold for their corresponding noisy versions. Figure 2 illustrates two different causal classes with measurement error. Specifically, Figure 2a represents the causal class of common-effect where V_1 and V_2 are d-connected conditional on either V_3 or its noisy version V_3^*. Figure 2b represents the causal class of causal-chain where V_1 and V_2 are d-separated conditional on V_3, whereas they are d-connected conditional on V_3^* (this observation also holds for the causal class of common-cause).

These lead to Propositions 2 and 3 which state that although the conditional d-connection relations between noisy variables are consistent with those given by the error-free variables, it is likely that some conditional d-separations will no longer hold when the observed variables incorporate measurement error such as the causal-chain example in Figure 3. In other words, under the large sample limit, the learned graph will contain all the conditional dependence relationships that are entailed by the error-free graph, but may miss some conditional independence relationships that appear in the error-free graph.

Proposition 2 *In a noisy graph G^*, if two error-free variables V_1 and V_2 are d-connected given a variable set $\boldsymbol{S}$, this d-connection will also hold for the noisy variables V_1^* and V_2^* conditional on the noisy variable set $\boldsymbol{S}^*$.*

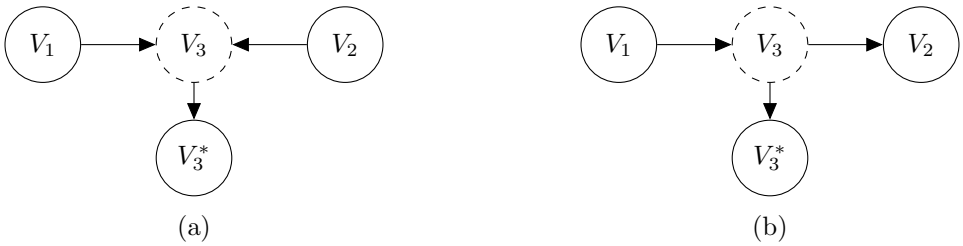


Figure 2: Modelling the presence of measurement error on the two different causal equivalence classes where case (a) represents the common-effect class, where V_1 and V_2 are d-connected conditional on either V_3 or V_3^*, and (b) represents the causal-chain class where V_1 and V_2 are d-separated conditional on V_3, whereas they are d-connected conditional on V_3^* (this also holds for the causal class of common-cause).

Proof If V_1 and V_2 are d-connected given $\boldsymbol{S}$, there must be a path p between V_1 and V_2 such that (i) every collider in p has a descendant in $\boldsymbol{S}$, and (ii) no other nodes in p are in $\boldsymbol{S}$. Because the nodes in $\boldsymbol{S}^*$ are descendants of their corresponding error-free variables in $\boldsymbol{S}$, the descendant of every collider in p would be in $\boldsymbol{S}^*$. Besides, since the nodes in $\boldsymbol{S}^*$ are leaf nodes in G^*, no nodes in p should be in $\boldsymbol{S}^*$. By combining $V_1^* \leftarrow V_1, p$ and $V_2 \rightarrow V_2^*$, we can get a path p' between V_1^* and V_2^* such that (i) every collider in p' has a descendant in $\boldsymbol{S}^*$, and (ii) no other nodes in p' are in $\boldsymbol{S}^*$. Therefore, V_1^* and V_2^* are also d-connected conditional on $\boldsymbol{S}^*$. ■

According to Propositions 1 and 2, if two nodes V_i and V_j are adjacent in the noisy graph, they are d-connected conditional on any observed variable set in the noisy graph. Therefore they will be adjacent in the learned graph given the Causal Faithfulness assumption, and under large sample limit. However, some of the conditional independence relationships derived from the error-free graph might not hold in the observed data that contain measurement error and hence, would lead to spurious edges in the learned graph.

Consider the simple noisy graph shown in Figure 3a which is composed by three error-free variables V_1, V_2 and V_3, and one noisy variable V_3^*. According to Propositions 1 and 2, and with reference to the example in Figure 3a, the unconditional and conditional dependencies between error-free variables V_1 and V_3 extent to their observed versions V_1 and V_3^*. Therefore structure learning algorithms tend to produce an edge between V_1 and V_3 in the graph learned from observed data under large sample limit and similarly for V_2 and V_3). However, the only conditional independence relationship amongst the error-free variables, i.e., $V_1 \perp\!\!\!\perp V_2 \mid V_3$ would not hold when conditional on V_3^*. Therefore, we get $V_1 \not\perp\!\!\!\perp V_2 \mid V_3^*$ and the incorrect fully connected graph shown in Figure 3b as the learned graph. In other words, the measurement error on an unshielded non-collider misleads structure learning algorithms towards producing a spurious edge between its neighbours, resulting in a 3-vertex clique. Note that structure learning can reconstruct up to the CPDAG $V_1 - V_3 - V_2$, or one of its corresponding DAGs, when the input data does not incorporate measurement error.

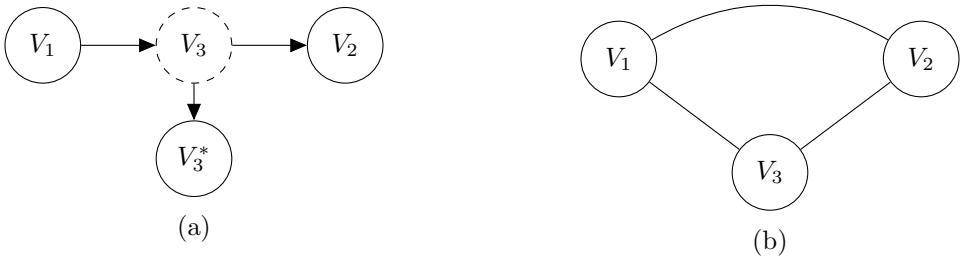


Figure 3: (a) A noisy graph that contains three error-free variables V_1, V_2 and V_3, and a noisy variable V_3^*. (b) The learned graph that entails the same dependence and independence relationships derived from the observed variables.

We refer to a path p between error-free variables V_1 and V_2 in a noisy graph G^* as a *connecting path* if (i) there are no colliders in p and (ii) all intermediate nodes in p incorporate measurement error.

Proposition 3 *Given the causal faithfulness assumption and large sample limit, for any two non-adjacent error-free variables V_1 and V_2 in a noisy graph G^*, if V_1 and V_2 are adjacent in the learned graph G^l, then there must be at least one connecting path p between V_1 and V_2 in G^*. Besides, for each connecting path p, there is a 3-vertex clique $\{V_1, V_2, V_k\}$ in the learned graph G^l, where V_k is a variable in p.*

Proof Without loss of generality, we consider the situation where both V_1 and V_2 incorporate measurement error in the observed data. Since V_1 and V_2 are adjacent in the learned graph G^l but not adjacent in the noisy graph G^*, there must be at least one path p between V_1 and V_2 in G^* such that (i) no node in p is a collider, and (ii) all intermediate nodes in p incorporate measurement error. Otherwise, a set of observed variables could d-separate V_1^* and V_2^* in G^*, and this would contradict with the adjacency between V_1 and V_2 in the learned graph. If there is a connecting path $p = \{V_1, S_1, S_2, \ldots, S_n, V_2\}$, we can obtain another connecting path $p' = \{V_1, S_1, S_2, \ldots, S_n\}$ between V_1 and S_n. Thus, V_1^* and S_n^* would also be d-connected in the noisy graph given any observed variable set which leads to the presence of an edge between V_1 and S_n in the learned graph. Besides, V_2 and S_n should still be adjacent in the learned graph given the Proposition 1. Therefore, V_1, V_2 and S_n form a 3-vertex clique in the learned graph. ■

We, therefore consider a 3-vertex clique as a sign for the presence of measurement error in at least one of the variables that make up the clique. When a learned graph contains such a clique, we need to determine whether the clique is in the error-free graph or the outcome of measurement error. If we could distinguish between these two possibilities, then we would be able to remove the spurious edges in the graph learned from noisy data. This challenge can be viewed as a type of a hidden variable problem. In our case, a potential hidden variable represents the error-free parent of its corresponding noisy version.

Figure 4 presents an example based on the PC-stable algorithm and the classic Asia network[1] Specifically, Figure 4a represents the true Asia network, Figure 4b represents the

1. The variables in the Asia network are all binary. This example assumes the sample size of the data is 10,000.

graph learned from the error-free data set, and Figure 4c represents the graph learned from the noisy data set with 5% measurement error on variable *bronc*, as defined by Equation 1, i.e., 5% of the value in *bronc* data are recorded by another valid but incorrect state. This relatively small rate of error has led to the spurious edge between *smoke* and *dysp*. This is because while *smoke* and *dysp* are independent conditional on the error-free variables *bronc* and *either*, this conditional independence is relaxed in the presence of measurement error on variable *bronc* and hence, the algorithm produces the additional FP edge. Moreover, this additional edge produces the 3-vertex clique $\{smoke, bronc, dysp\}$ that does not exist in the true Asia network nor in the graph learned from the error-free data set.

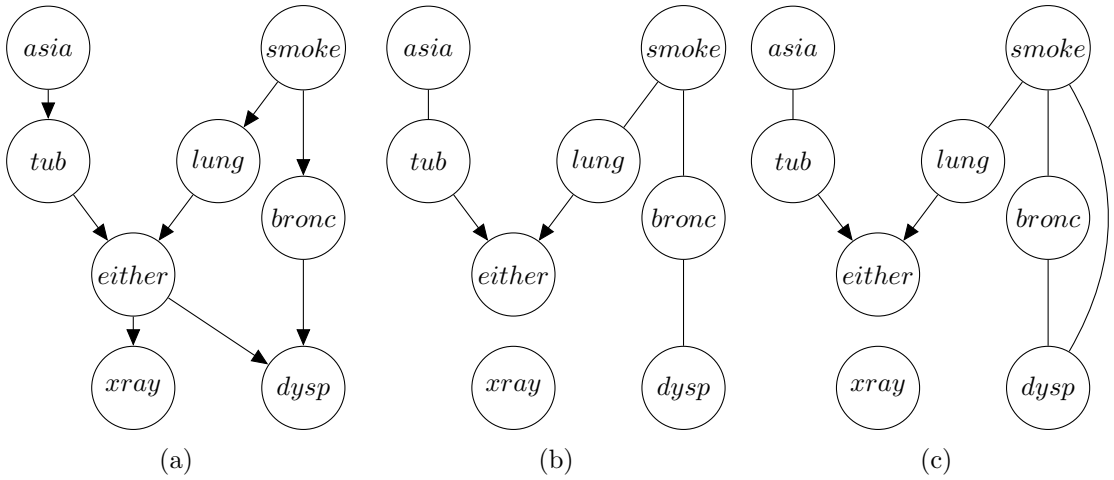


Figure 4: (a) The true Asia network. (b) The CPDAG learned by PC-stable given the error-free synthetic data set. (c) The CPDAG learned by PC-stable given the same synthetic data set but with 5% measurement error on variable *bronc*.

To investigate the impact of measurement error on BN structure learning in general, we have extended these experiments to four algorithms spanning different classes of learning. Namely, in addition to constraint-based PC-stable (Colombo and Maathuis, 2014), to the score-based HC (Bouckaert, 1994) and ILP (Cussens, 2011), and to hybrid H2PC (Gasse et al., 2014). We have used each of these algorithms to reconstruct 50 randomly generated BNs consisting of 20 Boolean nodes, using the method described in (Ide and Cozman, 2002). Each random network was used to generate two synthetic data sets of 10,000 sample size each; one error-free data set and another noisy data set with 10% measurement error on each variable.

Figure 5 compares the average number of 3-vertex cliques produced by each of the algorithms with and without measurement error, and with reference to the average number of 3-vertex cliques present in the ground truth graphs. These initial empirical results show that structure learning algorithms tend to produce more 3-vertex cliques in the graphs learned from noisy data sets compared to both the true graph and the graphs learned from error-free data sets. Moreover, score-based learning seems to be more sensitive to the measurement error compared to constraint-based learning, although this observation is based on the default hyperparameters as implemented in the corresponding packages (Scutari et al., 2010; Wongchokprasitti, 2019; Cussens, 2011) that we have used

to test the algorithms. These results support our hypothesis that 3-vertex clique can be viewed as a sign for the presence of measurement error in the input data.

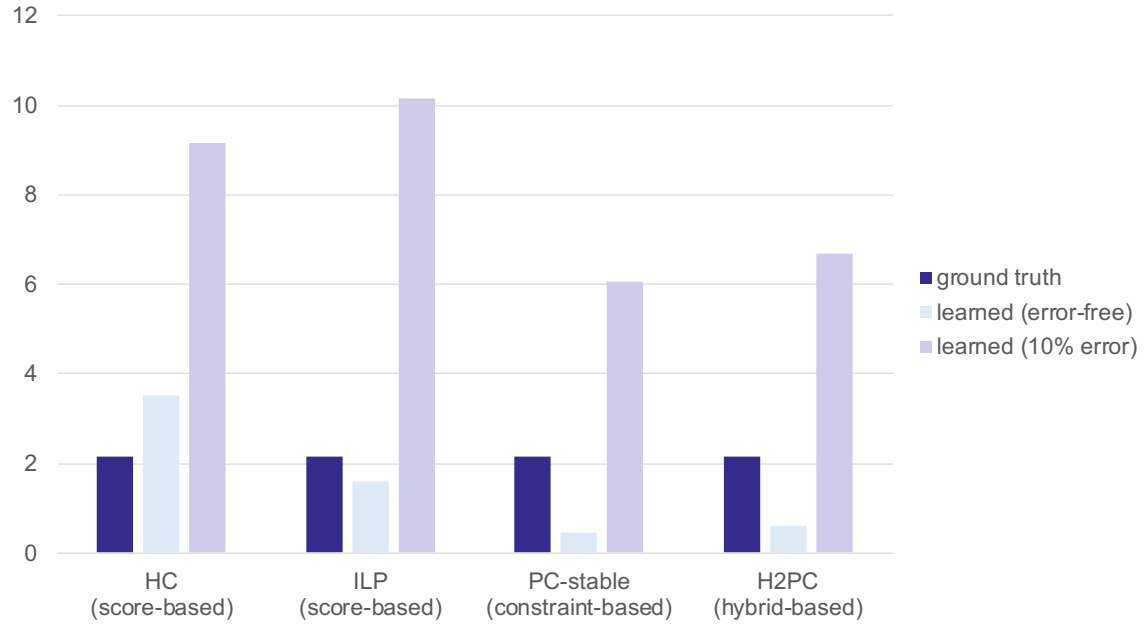


Figure 5: The average number of 3-vertex cliques in the ground truth graphs, the graphs learned from error-free data sets, and the graphs learned from observed data sets with 10% measurement error on each variable.

4. The Spurious Edge Detection (SED) algorithm

This section describes the Spurious Edge Detection (SED) algorithm which can be applied to the output graph produced by any other BN structure learning algorithm to discover and eliminate potential FP edges that tend to be the outcome of measurement error. The implementation of SED is available online [2]. Further to what has been discussed in Section 3, SED focuses its search for FP edges on the induced subgraph of 3-vertex cliques.

We consider every edge that connects two variables V_i and V_j in a 3-vertex clique $\{V_i, V_j, V_k\}$ in the learned graph G^l to be a *candidate spurious edge*. According to Proposition 3, the node V_k is likely to be a variable on a connecting path between V_i and V_j in the underlying noisy graph G^*, as long as V_i and V_j are not adjacent in G^*. Therefore, the conditional independence between V_i and V_j is likely to be retrieved by introducing an error-free variable of V_k in the learned graph, and treat the observed data of V_k as the observation of its noisy version. We define the Candidate Spurious Edge-Nodes pair $CSE(E_i)$ for a candidate spurious edge E_i as the set of nodes that are adjacent to both the endpoints of E_i. For instance, the CSE for Figure 6 is:

2. Our code is publicly available at `https://github.com/Enderlogic/Spurious-Edge-Detection`.

$$CSE = \begin{Bmatrix} V_1 \rightarrow V_2 : \{V_3, V_4\}, \\ V_1 \rightarrow V_3 : \{V_2\}, \\ V_1 \rightarrow V_4 : \{V_2\}, \\ V_2 \rightarrow V_3 : \{V_1, V_5\}, \\ V_2 \rightarrow V_4 : \{V_1\}, \\ V_2 \rightarrow V_5 : \{V_3\}, \\ V_3 \rightarrow V_5 : \{V_2\} \end{Bmatrix}$$

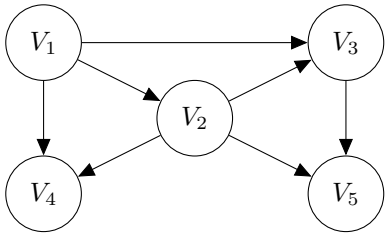


Figure 6: An example of a graph that contains 3-vertex cliques

Next, let us revisit the Asia network example in Figure 4c to investigate the possibility of a spurious edge in the presence of a single 3-vertex clique in the learned graph. Recall that this is the graph learned by PC-stable in the presence of 5% measurement error on variable *bronc*. Since the graph contains a single 3-vertex clique, the CSE for this graph is:

$$CSE = \begin{Bmatrix} bronc - dysp : \{smoke\}, \\ smoke - dysp : \{bronc\}, \\ smoke - bronc : \{dysp\} \end{Bmatrix}$$

We then perform three graphical reconstructions as shown in Figure 7, one for each candidate spurious edge, given the Independence rule defined in Section 2, to identify and eliminate a spurious edge. During the graph reconstruction process, we build the reconstructed graphs that entail all the independences and conditional independences of the learned graph, and test for an additional conditional independence relationship between the endpoints of the candidate spurious edge. For example, the graph in Figure 7a investigates the possibility of the edge $bronc - dysp$ being spurious and of the variable *smoke* incorporating measurement error[3], which is why it is replaced with a hidden unmeasured variable representing its error-free version, with the noisy version $smoke^*$ restructured as a child of the hidden variable. Moreover, the edge between *bronc* and *dysp* is removed such that the conditional independence $bronc \perp\!\!\!\perp dysp \mid smoke$ is introduced in the Figure 7a, by assuming that the data for the variable *smoke* are noisy, which could also explain the presence of clique $\{bronc, dysp, smoke\}$ in the learned graph shown in Figure 4c. Similarly, Figures 7b and 7c repeat this process for the remaining two variables in clique $\{bronc, dysp, smoke\}$.

Each reconstructed graph is then evaluated in terms of model selection between the learned and observed distributions using the well-established Bayesian Information Criterion (BIC) (Suzuki, 1993). While the true model is not present in the candidate collection of graphs, the BIC function should still select the model that converges with probability one to the quasi-true model as the sample size grows to infinite (Claeskens et al., 2008; Neath and Cavanaugh, 2012). The quasi-true model in a candidate collection is the most parsimonious model that is closest to the true model, as measured by the Kullback-Leibler information. Therefore, when a reconstructed graph obtains a higher BIC score than the learned graph, we consider the removal of that specific candidate spurious edge to produce a graph that is

3. In assessing whether $bronc - dysp$ is spurious, we do not check for measurement error on variables *bronc* and *dysp*, and this is because we assume that measurement error on the endpoints of the spurious edge cannot be the cause of that spurious edge. For example, if the error-free graph is $A \rightarrow B \rightarrow C$, measurement error on nodes A or C would not produce a spurious edge between A and C in the learned graph, whereas measurement error on node B would do.

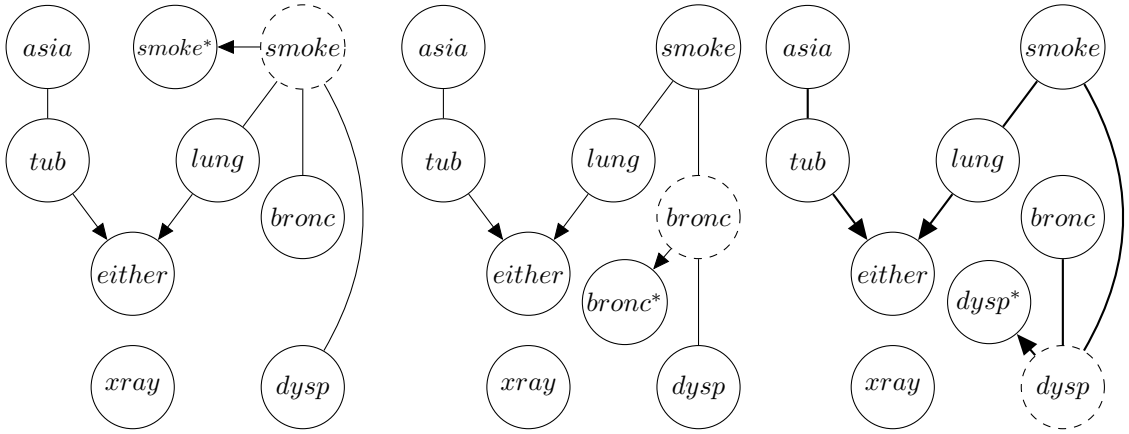


Figure 7: The three reconstructed graphs for clique $\{bronc, dysp, smoke\}$, based on the learned graph in Figure 4c. Dotted nodes represent possible hidden error-free parents of the suspected noisy node under assessment.

closer to the error-free graph. Because the reconstructed graphs include additional hidden variables, we adopt the Expectation-Maximization (EM) learning (Dempster et al., 1977) to compute the Log-Likelihood (LL) score of the BIC for each of the reconstructed graphs, and we describe this process in Appendix A.

When the endpoints of an edge in the learned graph are present in multiple 3-vertex cliques simultaneously, it is possible that there is more than one connecting paths between them in the underlying noisy graph, such that we may need to import multiple hidden variables in the reconstructed graph to retrieve the missing conditional independence relationship. For example, in Figure 8a, V_1 and V_3 are conditionally independent given V_2 and V_4. However, this conditional independency does not hold in the corresponding observed data, and thus there is a spurious edge between V_1 and V_3 in the learned graph shown in Figure 8b. In order to identify the true conditional independency between V_1 and V_3, i.e., $V_1 \perp\!\!\!\perp V_3 \mid \{V_2, V_4\}$, it requires that both the error-free nodes V_2 and V_4 are included in the reconstructed graph, as illustrated in Figure 8c.

The process we have used to reconstruct graphs is described in Algorithm 1, that takes as input a learned graph G^l, a set of noisy variables $\boldsymbol{V}$, a candidate spurious edge E, and a data set D. As described by Algorithm 1, a reconstructed graph is produced for each candidate spurious edge by replacing each involved noisy variable with an error-free variable, and adding the noisy variable as a child of its corresponding error-free variable. The hidden error-free variable has the same state space as the corresponding noisy variable. Moreover, the candidate spurious edge is removed from the reconstructed graph. Therefore, a reconstructed graph entails all the independences of the learned graph, plus an additional independency corresponding to the endpoints of the candidate spurious edge. The output of Algorithm 1 represents the difference in BIC score between the reconstructed graph and the input learned graph. If the input graph is a CPDAG, we compute the BIC score of that graph based on one of its valid DAGs, since all the DAGs that are part of the same Markov

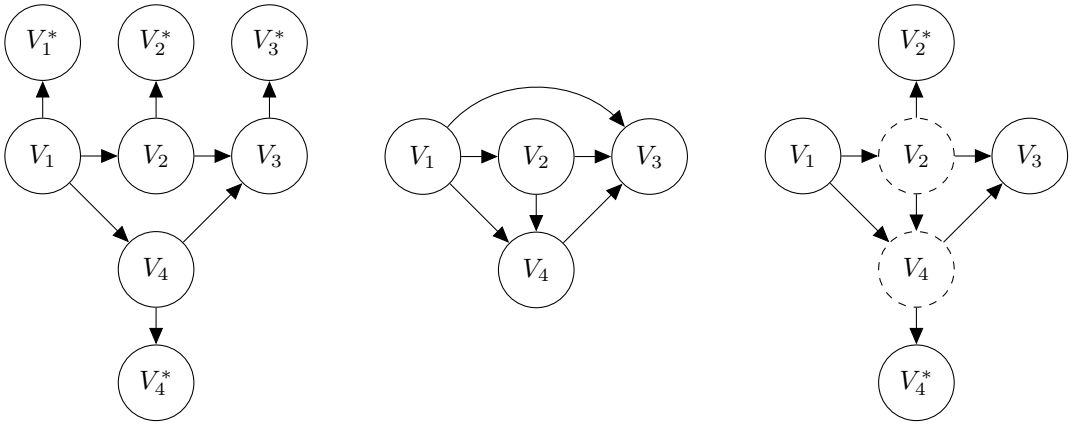


Figure 8: (a) A noisy graph with four variables that incorporate measurement error. (b) The learned graph with respect to (a). (c) The reconstructed graph that contains the conditional independence $V_1 \perp\!\!\!\perp V_3 \mid \{V_2, V_4\}$.

equivalence class would return the same BIC score. If the input graph is a PDAG which has no consistent DAG extensions, Algorithm 1 returns 0.

Algorithm 1 Graph reconstruction procedure

procedure RECONSTRUCTION($G^l, E, \boldsymbol{V}, D$)
  Input: learned graph G^l, candidate spurious edge E, noisy variables $\boldsymbol{V}$, data D
  Output: difference in BIC score between reconstructed graph and input graph
  Compute the BIC score $score_o$ of the input learned graph G^l
  Create a copy of graph G^l as G_r
  Replace each observed variable V in $\boldsymbol{V}$ in G_r with a hidden error-free variable that preserves the state space of V
  Reintroduce the observed variables $\boldsymbol{V}$ as noisy variables $\boldsymbol{V}^*$ in G_r, as the child of their corresponding error-free variables
  Remove edge E from G_r
  Compute the BIC score $score_o$ of the reconstructed graph G_r
  return $score_r - score_o$
end procedure

Next, we introduce the complete SED algorithm, which represents an iterative process that searches for spurious edges by recursively executing the aforementioned Algorithm 1, and produces a modified graph that does not contain the edges identified as possible false positives. The pseudocode of the SED algorithm is described in Algorithm 2. Firstly, SED initialises the modified graph G_{mod} as a copy of the learned graph G^l, and generates the candidate spurious edge-nodes pairs CSE from G^l. Then, SED recursively removes the candidate edges ordered by with the highest positive output as determined by Algorithm 1 given the modified graph, and by assuming that there are l connecting paths between the endpoints of the candidate edge in the underlying noisy graph, where l is initially set to 1

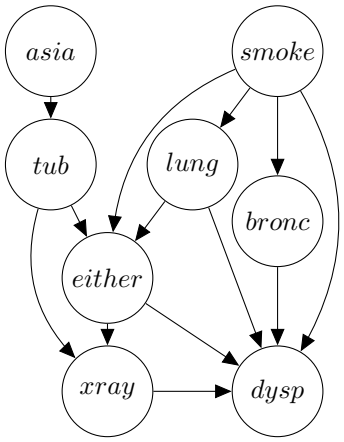
and iteratively increased by 1 when no more edges can be identified as spurious give the current value of l. Therefore, SED is able to explore multiple connecting paths between the endpoints of every candidate spurious edge. The whole process is terminated when l is larger than the maximal size of $CSE(E_i)$. Note that when an edge between V_1 and V_2 in the learned graph is detected as spurious by assuming that there is a connecting path between V_1 and V_2 via V_3 in the underlying noisy graph, it implies that V_1 and V_3 are either adjacent or connected through another connecting path that does not contain V_2. If V_1 and V_3 are adjacent in the noisy graph, it is not necessary to test whether the edge between them is spurious in the learned graph. If there is a connecting path between V_1 and V_3 that does not contain V_2, then conditioning on the error-free variable V_2 will not d-separate V_1 and V_3 in the noisy graph. Therefore, SED will check for multiple connecting paths between V_1 and V_3.

Algorithm 2 Spurious Edge Detection (SED) algorithm

procedure SED(G^l, D)
  Input: learned graph G^l, data set D
  Output: modified graph G_{mod}
  $G_{mod} = G^l$
  initialise CSE from G^l
  $l = 1$
  repeat
    $CSE_l = \{\}$
    for $E_i \in CSE$ **do**
      $CSE_l(E_i)$ = all subsets of $CSE(E_i)$ with length l
    end for
    while $\max_{E_i \in CSE, \boldsymbol{V}_j \in CSE_l(E_i)} Reconstruction\left(G^l, E_i, \boldsymbol{V}_j, D\right) > 0$ **do**
      $E_m, \boldsymbol{V_m} = \arg\max_{E_i \in CSE, \boldsymbol{V}_j \in CSE_l(E_i)} Reconstruction\left(G^l, E_i, \boldsymbol{V}_j, D\right)$
      remove E_m from G_{mod}
      remove E_m from CSE
      if $l == 1$ **then**
        prune the edges between each endpoint of E_m and V_m from CSE_l
      end if
    end while
    $l = l + 1$
  until $l > \max_{E_i \in CSE} |CSE(E_i)|$
  return G_{mod}
end procedure

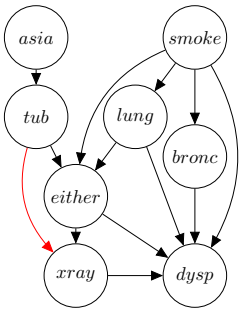
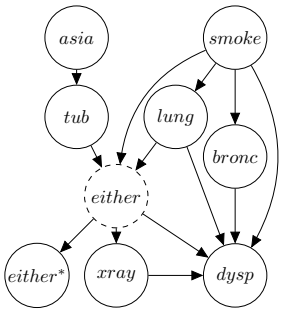
Table 1 presents a worked example that illustrates the different steps of SED when applied to the graph shown in Figure 9. This experiment is based on the Asia network learned by the HC algorithm from a synthetic data set with 10,000 samples and 5% measurement error on each observed variable. In Table 1, the modified graph represents the state of modified graph at the given iteration, the optimal reconstructed graph represents the reconstructed graph with the highest positive output, and CSE represents the candi-

date spurious edge set that contains edges that continue to be tested for false positives. The red edges depicted in Table 1 represent the edges classified as spurious by SED, whereas the blue edges represent the edges pruned (i.e., no longer being considered as candidate spurious edges) after each iteration. A candidate spurious edge is pruned when no valid reconstructed graph is found to have a score that is higher than the score of the learned graph.



$$CSE = \left\{ \begin{array}{l} tub \to either : \{xray\}, \\ tub \to xray : \{either\}, \\ smoke \to lung : \{either, dysp\}, \\ smoke \to either : \{lung, dysp\}, \\ smoke \to bronc : \{dysp\}, \\ smoke \to dysp : \{lung, either, bronc\}, \\ lung \to either : \{smoke, dysp\}, \\ lung \to dysp : \{smoke, either\}, \\ bronc \to dysp : \{smoke\}, \\ either \to xray : \{tub, dysp\}, \\ either \to dysp : \{smoke, lung, xray\}, \\ xray \to dysp : \{either\} \end{array} \right\}$$

Figure 9: Left: The Asia graph learned by the HC algorithm from a synthetic data set with 10,000 samples and 5% measurement error on each observed variable. Right: the candidate spurious edge-nodes pairs CSE of the left graph.

Iteration	Modified graph	Optimal reconstructed graph	CSE
1			$tub \to either : \{xray\}$ $tub \to xray : \{either\}$ $smoke \to lung : \{either, dysp\}$ $smoke \to either : \{lung, dysp\}$ $smoke \to bronc : \{dysp\}$ $smoke \to dysp : \{lung, either, bronc\}$ $lung \to either : \{smoke, dysp\}$ $lung \to dysp : \{smoke, either\}$ $bronc \to dysp : \{smoke\}$ $either \to xray : \{tub, dysp\}$ $either \to dysp : \{smoke, lung, xray\}$ $xray \to dysp : \{either\}$

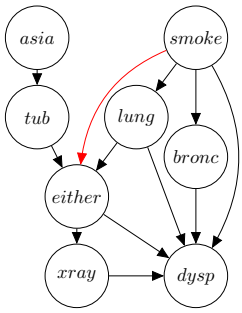
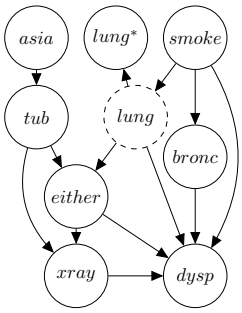
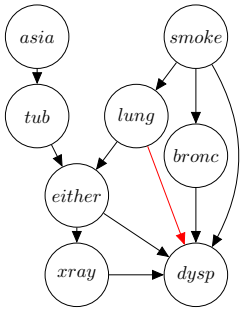
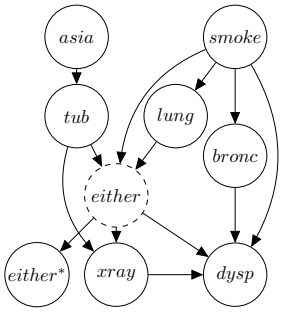
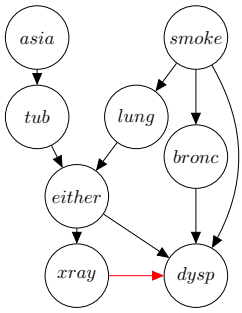
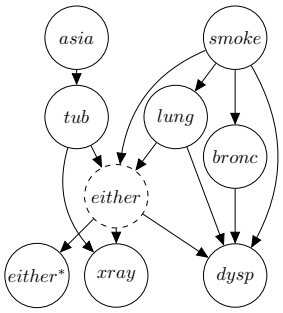
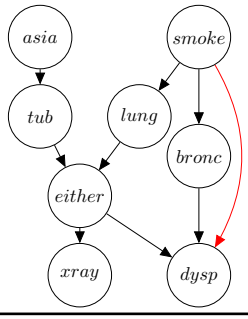
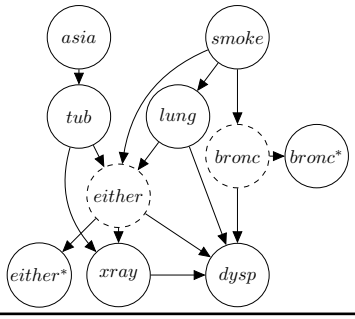
2			$smoke \rightarrow lung : \{either, dysp\}$ $smoke \rightarrow either : \{lung, dysp\}$ $smoke \rightarrow bronc : \{dysp\}$ $smoke \rightarrow dysp : \{lung, either, bronc\}$ $lung \rightarrow either : \{smoke, dysp\}$ $lung \rightarrow dysp : \{smoke, either\}$ $bronc \rightarrow dysp : \{smoke\}$ $either \rightarrow xray : \{tub, dysp\}$ $either \rightarrow dysp : \{smoke, lung, xray\}$ $xray \rightarrow dysp : \{either\}$
3			$smoke \rightarrow lung : \{either, dysp\}$ $smoke \rightarrow bronc : \{dysp\}$ $smoke \rightarrow dysp : \{lung, either, bronc\}$ $lung \rightarrow either : \{smoke, dysp\}$ $lung \rightarrow dysp : \{smoke, either\}$ $bronc \rightarrow dysp : \{smoke\}$ $either \rightarrow xray : \{tub, dysp\}$ $either \rightarrow dysp : \{smoke, lung, xray\}$ $xray \rightarrow dysp : \{either\}$
4			$smoke \rightarrow lung : \{either, dysp\}$ $smoke \rightarrow bronc : \{dysp\}$ $smoke \rightarrow dysp : \{lung, either, bronc\}$ $lung \rightarrow either : \{smoke, dysp\}$ $bronc \rightarrow dysp : \{smoke\}$ $either \rightarrow xray : \{tub, dysp\}$ $either \rightarrow dysp : \{smoke, lung, xray\}$ $xray \rightarrow dysp : \{either\}$
5			$smoke \rightarrow lung : \{either, dysp\}$ $smoke \rightarrow dysp : \{lung, either, bronc\}$ $lung \rightarrow either : \{smoke, dysp\}$ $either \rightarrow xray : \{tub, dysp\}$ $either \rightarrow dysp : \{smoke, lung, xray\}$

Table 1: The steps of the SED algorithm in modifying the Asia graph learned by HC from synthetic data of sample size 10,000 with 5% measurement error on all variables. The red edges represent the edges classified as spurious in each iteration, whereas the blue edges represent the edges being pruned (i.e., no longer being considered as candidate spurious edges) after each iteration.

The illustration in Table 1 starts by initialising the modified graph as a copy of the learned graph shown in Figure 9, and determining the CSE based on that graph. Then, SED iterates over the candidate spurious edges by assuming that there is one connecting path between the endpoints of the candidate edges, i.e., assuming one variable which is adjacent to the endpoints of the candidate spurious edge as noisy, and importing its error-free parent in the reconstructed graph. During the iterative process, SED first identifies $tub \rightarrow xray$ as spurious since the reconstructed graph that does not contain $tub \rightarrow xray$ returns the highest score. SED then removes $tub \rightarrow xray$ from the modified graph and further prunes $tub \rightarrow either$ from CSE (i.e., it is no longer considered as a candidate spurious edge). This is because SED finds that there is a connecting path between tub and $xray$ via $either$, which in turn implies that tub and $either$ are either adjacent or connected through a connecting path that does not contain $xray$ in the true noisy graph. Therefore, it is not necessary to examine whether $tub \rightarrow either$ is spurious under the assumption of a single connecting path between tub and $either$ via $xray$.

In the following iterations, SED repeats the above process and detects another three spurious edges $smoke \rightarrow either, lung \rightarrow dysp$ and $xray \rightarrow dysp$. At that point, no further edges are identified as spurious under the assumption that the cause is a single noisy variable. SED continues assessing the remaining candidate spurious edges by assuming there are two connecting paths between the endpoints, i.e., assuming two noisy variables which are both adjacent to the endpoints of the candidate spurious edge, and importing their error-free parents in the reconstructed graph. At that stage, SED discovers one more spurious edge, $smoke \rightarrow dysp$, and prunes $smoke \rightarrow lung, lung \rightarrow either$ and $either \rightarrow xray$ from CSE since no other higher scoring reconstructed graphs can be found given $l = 2$. Then, l increases to 3 and SED prunes the last candidate spurious edge $either \rightarrow dysp$ from CSE, since no other reconstructed graph can further increase BIC. SED then terminates the search process since, at this point, all candidate spurious edges are pruned from CSE.

5. Empirical evaluation

We validate the effectiveness of the SED algorithm, which can be viewed as a structure learning addon, by applying it to five well-established structure learning algorithms spanning different classes of learning. These are the score-based HC, GES and ILP, the constraint-based PC-stable and the hybrid H2PC. We use the bnlearn R package (Scutari et al., 2010) to test the effect on HC and H2PC, the rcausal R package (Wongchokprasitti, 2019) for PC-stable and GES, and the pygobnilp python package (Cussens, 2011) for ILP.

We use the BIC score as the objective function for the three score-based HC, GES and ILP algorithms, and for the score-based phase of H2PC. For the constraint-based algorithm PC-stable, including the constraint-based phase in H2PC, we use the G-square test as the statistical test and set the threshold for rejecting the null hypothesis at 0.05. Lastly, ILP's

maximum in-degree is set to 3 (default hyperparameter). Because BIC is a score-equivalent objective function, HC, GES, ILP and H2PC produce a DAG from a Markov Equivalent Class of DAGs, and which we convert into the corresponding CPDAGs to be used as the input of the SED algorithm; i.e., input graph G in Algorithm 2. Recall that when the constraint-based PC-stable returns a PDAG that cannot be converted into a DAG, SED makes no modifications since the BIC score cannot score that PDAG and hence, Algorithm 1 returns 0 in this case. We employ two metrics to evaluate the learned CPDAGs. These are the F1 score which combines the *Precision* and *Recall* in the following form:

$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall} \tag{2}$$

and the Structural Hamming Distance (SHD) (Tsamardinos et al., 2006) which represents the number of edge additions, edge removals and arc reversals required to move from the learned graph to true graph. Since the graphs learned from larger networks generally have higher SHD scores, we divide the SHD score by the number of edges present in the true graph to adjust SHD relative to the edges that can be discovered.

The experiments are based on synthetic data generated from seven real-world BN models that are publicly available in the bnlearn repository (Scutari, 2020). These are the Asia, Alarm, Child, Insurance, Mildew, Water and Hailfinder networks. For each network, we generated seven error-free data sets with the sample sizes ranging from 100 to 100,000. Moreover, for each error-free data set we generate two noisy data sets by setting the error rate α_i for every variable V_i in a network to 0.1 and 0.2 respectively. Specifically, for each state v_i^l of an error-free variable V_i, we assign a randomised error rate α_i^l, with an upper bound of α_i, where the probability of the error for each state v_i^l follows a Dirichlet distribution. This process produces the corresponding noisy conditional probability distribution of each noisy variable V_i^* based on the following equation:

$$P\left(V_i^* \mid V_i = v_i^l\right) = \begin{cases} \alpha_{i1}^l, & V_i^* = v_i^1 \\ \alpha_{i2}^l, & V_i^* = v_i^2 \\ \vdots & \vdots \\ 1 - \alpha_i^l, & V_i^* = v_i^l \\ \vdots & \vdots \\ \alpha_{ir_i}^l, & V_i^* = v_i^{r_i} \end{cases} \tag{3}$$

where the parameters $\left(\alpha_{i1}^l, \alpha_{i2}^l, \cdots, \alpha_{ir_i}^l\right) \sim \alpha_i^l Dir \underbrace{(1, \ldots, 1)}_{r_i - 1}$ such that $\alpha_i^l = \sum_{j=1}^{r_i} \alpha_{ij}^l$, r_i represent the number of states in V_i.

5.1 Results

We explore the performance of the SED algorithm on both error-free and noisy data sets. Figure 10 presents the F1, precision and recall scores produced by the five algorithms averaged across all seven networks, on both the error-free and noisy data sets, with and without SED modifications. Note that in the case of error-free data sets, there is no visible difference

in the precision, recall and F1 scores between the learned graphs and the graphs modified by SED. From this, we can conclude that SED performs largely insignificant modifications to the graphs learned from error-free data sets. On the other hand, the modifications made on graphs learned from noisy data have led to noticeable improvements in terms of the precision and F1 metrics, and particularly in cases where data have higher sample size. This can be explained by the fact that the structure learning algorithms generally tend to produce more edges when the input data contain higher samples, such that more false positive 3-vertex cliques that could be detected and corrected by SED. We present the results of 3-vertex cliques in Appendix B. Another explanation is that the EM learning used by SED is less effective when the sample size of the input data is low.

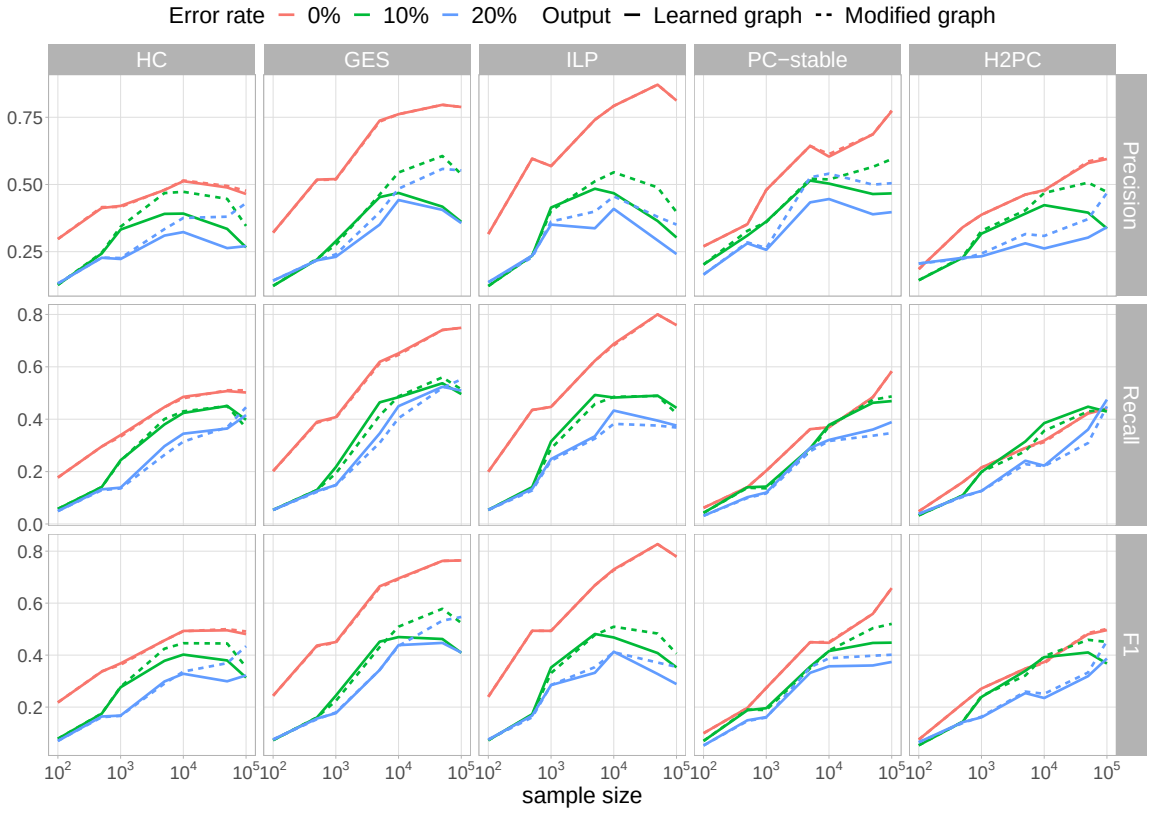


Figure 10: The average precision, recall and F1 scores of the graphs produced by the five algorithms where solid lines represent the results before SED modifications, dashed lines represent the results after SED modifications, red lines the results based on error-free data, green lines the results based on noisy data with 10% error rate, and blue lines the results based on noisy data with 20% error rate.

Specifically, the results show that SED improves the average precision and F1 scores across all the five algorithms, when the data contain measurement error (both 10% or 20%) and when the sample size is larger than 1,000. This suggests that the edges removed by SED are mainly false positives, even though SED would occasionally also remove some true positive edges which causes recall to drop slightly for some of the experiments. When

the sample size is less than 1,000, the learned graphs tend to contain a small number of 3-vertex cliques (refer to the Appendix B), and this gives little to no opportunity to SED to make modifications. The results also show that there is no apparent difference in the gain in score over the two different levels of measurement error tested. Interestingly, the improvements on graphs produced by score-based HC, GES and ILP are somewhat stronger compared to the improvements on graphs produced by PC-stable and H2PC, based on the hyperparameter defaults of these algorithms. These results are consistent with the empirical analysis presented in Figure 5, which shows that score-based learning is more sensitive to measurement error compared to constraint-based learning.

Figure 11 repeats these results for the SHD score. While the results are largely consistent with those based on the F1 score, the improvements appear to be major and more consistent in terms of the SHD score. Specifically, Figure 11 shows that after applying SED to the graphs learned from noisy data by all five structure learning algorithms, the number of removed and re-oriented edges required to convert the learned graphs to true graphs is significantly reduced, and this reinforces the observation that SED is effective at removing false positive edges. Figure 11 also shows that the number of edges produced by the algorithms increases slightly with measurement error as expected. This implies that SED is likely to have more opportunities to remove edges as the rate of measurement error increases. Overall, the SHD results suggest that the SED algorithm improves the graphs learned by the structure learning algorithms by successfully eliminating a greater number of false positive, in relation to true positive, edges.

Table 2 presents the number and percentage of modified graphs that are better, equal or worse than the learned graphs, under different experimental settings and evaluation metrics. The results show that when no measurement error exists in the input data (i.e., error-free case), the SED algorithm preserves the learned graph in around 90% of the cases, and improves or decreases the performance approximately in equal proportions for the remaining 9 % of cases. Specifically, without measurement error in the data, the modifications increased the F1 score in 11 (5%) graphs and decreased it in 10 (4%) graphs. Similarly, the modifications increased the SHD score in 10 (5%) graphs and decreased it in 9 (4%) graphs.

On the other hand, when measurement error exists in the input data (i.e., noisy cases), the SED modifications improve considerably more graphs than the graphs worsened. According to the SHD score, the SED modifications improve 120 (49%) and 130 (53%) learned graphs and only worsen just 9 (4%) and 10 (4%) learned graphs when the error rate is 10% and 20% respectively. In terms of the F1 score, the SED modifications improve 93 (38%) and 109 (44%) learned graphs and worsen 51 (21%) and 48 (20%) learned graphs when the error rate is 10% and 20% respectively. These percentages are generally consistent across all five algorithms irrespective of their class of learning.

Lastly, Figure 12 presents the average execution time needed to produce the learned and the modified graphs, across the different algorithms and over different error rates and sample sizes. The execution time for the modified graphs refers to the time it takes SED to modify the learned graphs. The five structure learning algorithms used to produce the learned graphs spent similar time learning the graphs from error-free data and noisy data with 10% error rate, and were slightly faster (in general) when learning from noisy data with 20% error rate. When it comes to SED, because the number of 3-vertex cliques tends to

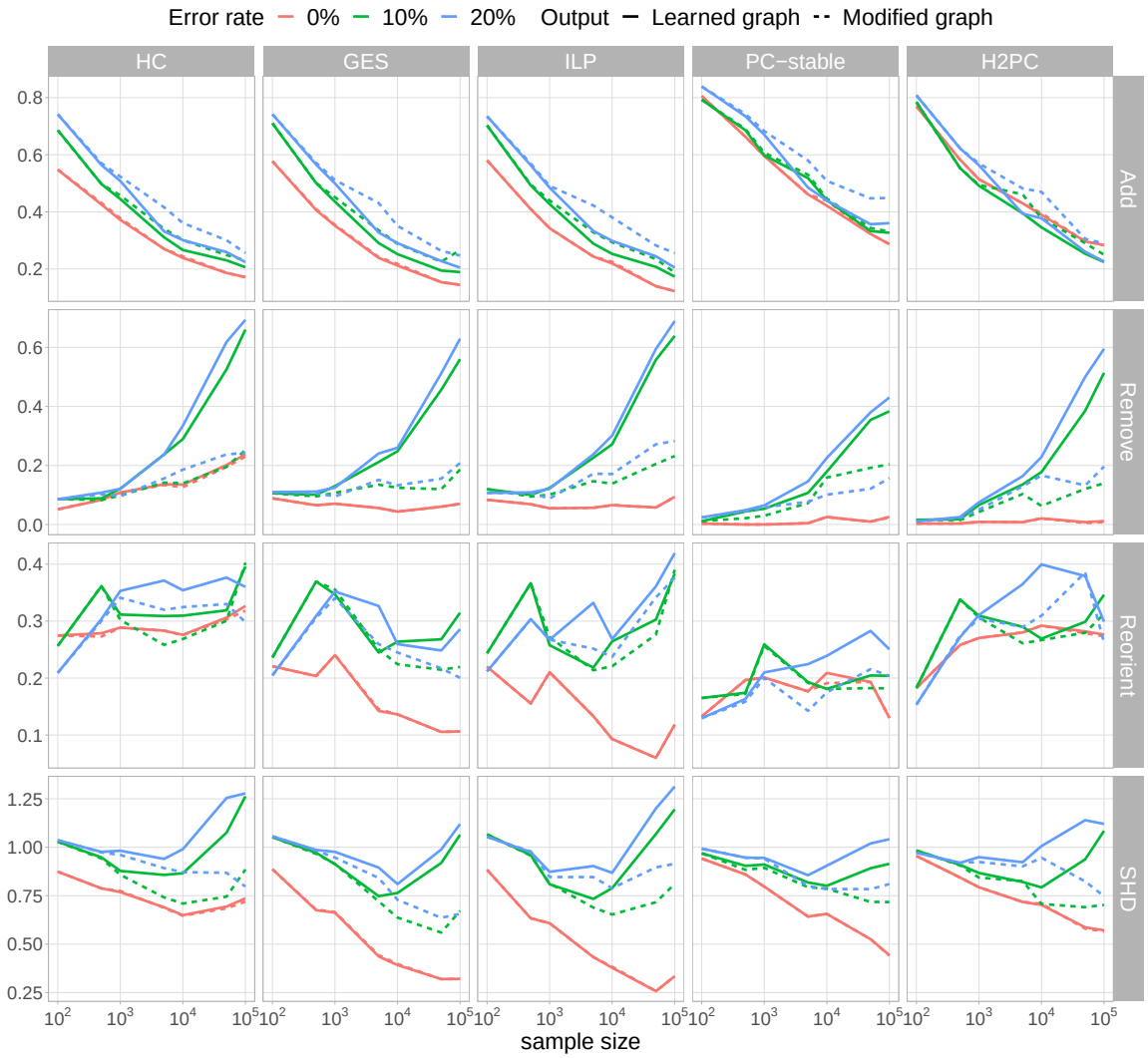


Figure 11: The average re-scaled SHD scores and its three components produced by the four algorithms, where solid lines represent the results before SED modifies the graphs, dashed lines represent the results after SED modifies the graphs, green lines the results based on noisy data with 10% error rate, and blue lines the results based on noisy data with 20% error rate.

increase with both the measurement and the sample size of the input data (see Appendix B), it naturally spends more time to process learned graphs produced with higher error rate and/or sample size. When the sample size is less than or equal to 10,000, the execution time spent by SED is generally less than the time spent by the structure learning algorithms, whereas the execution time is similar to that of the structure learning algorithms when the sample size is larger than 10,000.

Algorithm	Modified graph vs Learned graph	Error rate					
		0%		10%		20%	
		F1	SHD	F1	SHD	F1	SHD
HC	Better	7/14%	7/14%	25/51%	30/61%	25/51%	28/57%
	Equal	38/78%	39/80%	16/33%	18/37%	18/37%	19/39%
	Worse	4/8%	3/6%	8/16%	1/2%	6/12%	2/4%
GES	Better	0/0%	0/0%	20/41%	26/53%	21/43%	27/55%
	Equal	45/92%	45/92%	17/35%	21/43%	18/37%	20/41%
	Worse	4/8%	4/8%	12/24%	2/4%	10/20%	2/4%
ILP	Better	0/0%	0/0%	14/28%	23/47%	18/37%	25/51%
	Equal	48/98%	48/98%	19/39%	25/51%	19/39%	22/45%
	Worse	1/2%	1/2%	16/33%	1/2%	12/24%	2/4%
PC-stable	Better	2/4%	1/2%	13/27%	16/33%	25/51%	26/53%
	Equal	47/96%	48/98%	30/61%	29/59%	15/31%	21/43%
	Worse	0/0%	0/0%	6/12%	4/8%	9/18%	2/4%
H2PC	Better	2/4%	2/4%	21/43%	25/51%	20/41%	24/49%
	Equal	46/94%	46/94%	19/39%	23/47%	18/37%	23/47%
	Worse	1/2%	1/2%	9/18%	1/2%	11/22%	2/4%
Overall	Better	11/5%	10/4%	93/38%	120/49%	109/44%	130/53%
	Equal	224/91%	226/92%	101/41%	116/47%	88/36%	105/43%
	Worse	10/4%	9/4%	51/21%	9/4%	48/20%	10/4%

Table 2: The number and percentage of modified graphs that are better, equal or worse than the learned graphs in terms of graphical accuracy, for each algorithm, error-rate, and over different evaluation metrics.

6. Concluding remarks

This paper described the SED algorithm that can be viewed as a structure learning addon which can be incorporated as an additional learning phase to discrete BN structure learning algorithms. The purpose of SED is to discover and eliminate potential false positive edges that structure learning algorithms tend to produce when learning graphs from data that contain measurement error, irrespective of their class of learning.

We have applied SED modifications to graphs produced by algorithms spanning different classes of learning (i.e., score-based, constraint-based and hybrid learning). The results are based on both error-free and noisy synthetic data that vary in sample size, and which have been generated from real-world BN models that also greatly vary in terms of the size of network. SED is a heuristic algorithm that may lack the theoretical guarantees of asymptotic correctness of its results. On this basis, we derive our conclusions solely on the basis of the empirical investigation, which shows that SED generally maintains, or slightly

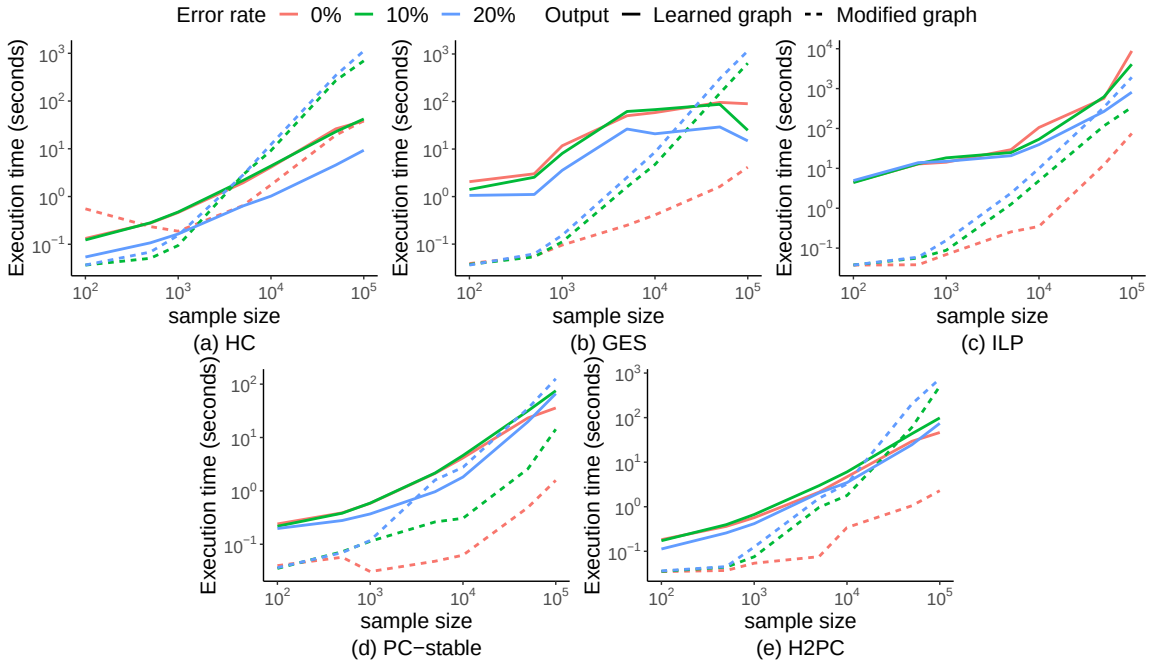


Figure 12: Average execution time needed to produce the learned and modified graphs for the specified algorithms, across different error rate and sample size combinations. The execution time of SED is based on the time it takes to modify graphs.

improves, the graphs produced by other algorithms when these graphs are learned from error-free data, and effectively improves the graphs learned from noisy-data.

A limitation of our work is that SED is restricted to discrete data. While extending SED to continuous data might be a sensible direction for future work, it should be noted that working with continuous data is likely to introduce further challenges, and this is because the computational complexity of EM learning tends to increase substantially when applied to continuous data. Another limitation is that the proposed algorithm relies on the assumption that a noisy variable is independent of other variables in the network conditional on its error-free version, and this assumption is often considered to be too strong in some fields (Hu, 2008). For example, a survey on unemployment data by Bound et al. (2001) shows that unemployment rate is underestimated, and the underestimation error appears to be dependent on the demographic characteristics of the respondent, such as age and sex. Moreover, since the problem of measurement error can be viewed as a special case of a hidden variable problem, future work could extend the application of this approach to structure learning algorithms designed to learn graphical structures under the assumptions of causal insufficiency (Zhang, 2008; Ogarrio et al., 2016).

Acknowledgments

This research was supported by the EPSRC Fellowship project EP/S001646/1 on Bayesian Artificial Intelligence for Decision Making under Uncertainty (Constantinou, 2018), and by The Alan Turing Institute in the UK under the EPSRC grant EP/N510129/1.

Appendix A. EM algorithm and BIC score

The EM algorithm (Lauritzen, 1995) is an iterative process that computes the Maximum Likelihood Estimation (MLE) of the parameters θ for a given structure and from incomplete data. Generally, The EM algorithm can be decomposed in two steps, known as the Expectation step (E step) and the Maximization step (M step). In the E step, the EM algorithm computes the expected log-likelihood function $Q\left(\theta \mid \theta^t\right)$ based on θ^t obtained with each sample (data row) in the data. Assuming $\boldsymbol{X}$ represents a set of variables with missing values in data set D with sample size N, the expectation of the LL function is:

$$Q\left(\theta \mid \theta^t\right) = \sum_{m=1}^{N} \sum_{\boldsymbol{x} \in \Omega_{\boldsymbol{X}}} \mathbb{P}\left(\boldsymbol{X} = \boldsymbol{x} \mid D_m, \theta^t\right) \log\mathbb{P}\left(\boldsymbol{X} = \boldsymbol{x}, D_m \mid \theta\right) \tag{4}$$

At the M step, the EM algorithm revises θ by maximising the expected LL:

$$\theta^{t+1} = \arg\max_{\theta} Q\left(\theta \mid \theta^t\right) \tag{5}$$

The EM algorithm starts from a random initialisation of θ and terminates when the LL converges over a given threshold ϵ:

$$\log\mathbb{P}\left(D \mid \theta^t\right) - \log\mathbb{P}\left(D \mid \theta^{t-1}\right) < \epsilon \tag{6}$$

where ϵ is a threshold for judging whether the process is converged.

Applying EM learning on a BN requires that we compute:

$$\widetilde{N}_{ijk}^t = \sum_m \mathbb{P}\left(V_i = k, pa\left(V_i\right) = j \mid D_m, \theta^t\right) \tag{7}$$

for the E step, where $\widetilde{N}_{ijk}$ represents the expected count of number of records where the value of variable $V_i = k$ and its parents $pa\left(V_i\right) = j$. For the M step, the solution of equation 5 has the following form:

$$\theta^{t+1} = \frac{\widetilde{N}_{ijk}^t}{\sum_k \widetilde{N}_{ijk}^t} \tag{8}$$

Once the parameters of the model are estimated, the LL obtained by EM is used as the LL input in the BIC equation to measure the goodness-of-fit of a given reconstructed graph G_r with respect to the observed data. Specifically, the BIC score of a BN model M and corresponding data set D is defined as:

$$BIC\left(M \mid D\right) = \log\mathbb{P}\left(D \mid M\right) - \frac{1}{2}\log\left(N\right) d \tag{9}$$

where N is the sample size of data set D and $d = \sum_i \left(r_i - 1\right) q_i$ is the number of free parameters in M, where r_i represents the number of states in variable V_i and q_i represents

the number of configuration of the parents of V_i. When computing the BIC score on Bayesian Network without hidden variables, due to the decomposability of the LL function, the equation 9 can be simplified as:

$$BIC\left(M \mid D\right) = \sum_{ijk} N_{ijk} \frac{N_{ijk}}{N_{ij}} - \frac{1}{2} \log\left(N\right) d \tag{10}$$

where N_{ijk} is the number of counts when $V_i = k$ and $pa\left(V_i\right) = k$ and $N_{ij} = \sum_k N_{ijk}$.

However, when computing the BIC score on a reconstructed graph, the LL function is not decomposable due to the presence of hidden variable, which means that we cannot directly use equation 10. Instead, we use the LL converged at the final M-step as described in equation 6. We use the formula $d = \sum_i \left(r_i - 1\right) q_i$ to determine the number of free parameters d in the reconstructed graph, considering both the hidden and observed variables. This equation represents the upper bound of the number of free parameters of graphs with hidden variables (Geiger et al., 1998, 2001). Because there is no closed form solution to compute the number of free parameters of the reconstructed graphs that contain hidden variables, we assume the highest theoretical upper bound of the number of free parameters, and this implies high dimensionality penalty which in turn decreases the likelihood that the reconstructed graphs will outperform the learned graphs in terms of the BIC score. Moreover, if the learned graph or a reconstructed graph is a CPDAG, we will randomly select a DAG from the Markov equivalence class of the CPDAG and retrieve the BIC score of that DAG to represent the BIC score of the CPDAG, since the BIC score is equivalent for Markov equivalent DAGs.

Appendix B. The number of 3-vertex cliques produced under different scenarios

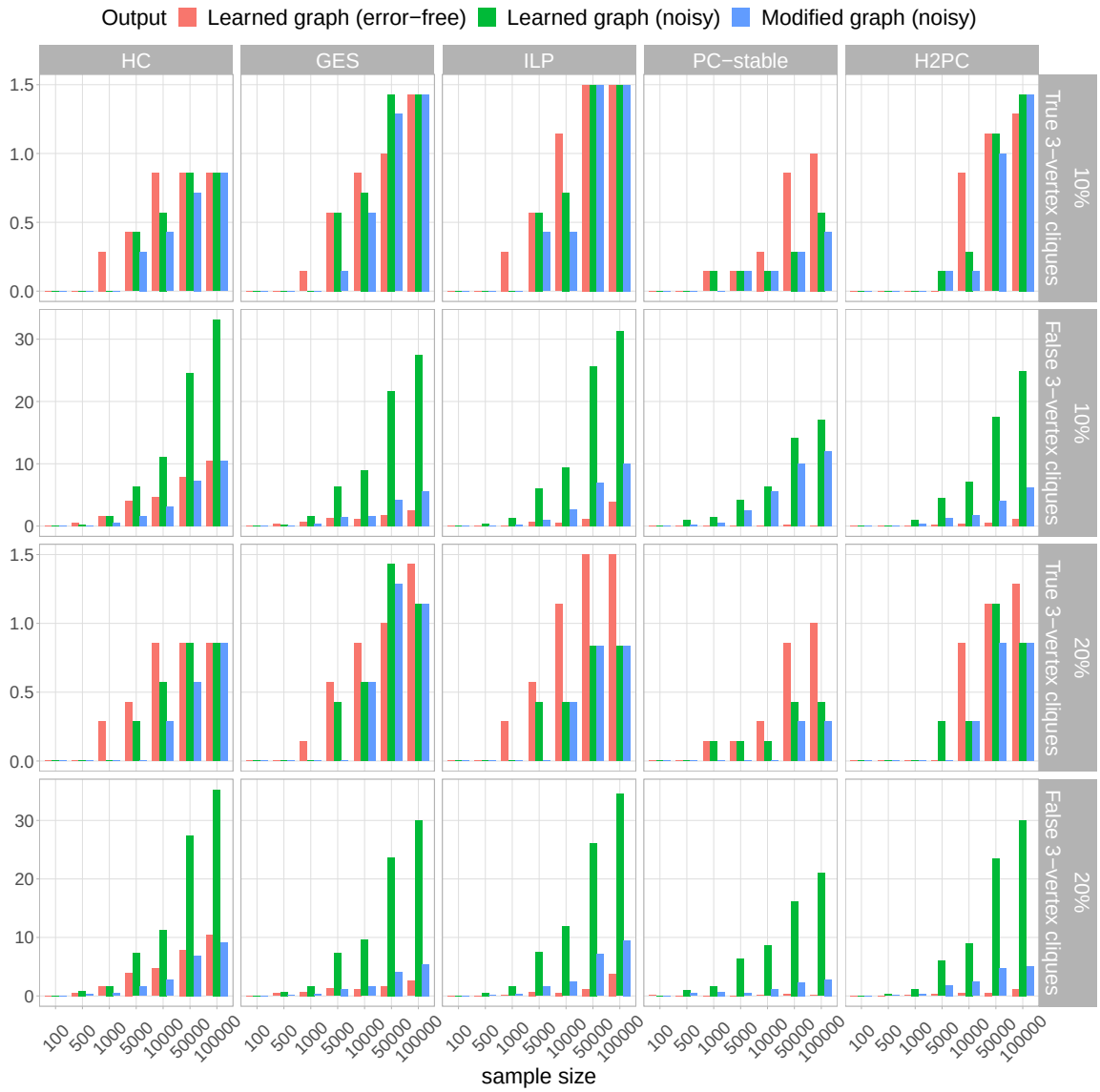


Figure 13: The average number of false and true 3-vertex cliques produced by the learned graphs learned from error-free data sets, the learned graphs learned from noisy data sets and the modified graphs learned from noisy data sets.

References

Tineke Blom, Anna Klimovskaia, Sara Magliacane, and Joris M Mooij. An upper bound for random measurement error in causal discovery. pages 570–579, 2018.

Christopher R Bollinger and Martijn van Hasselt. Bayesian moment-based inference in a regression model with misclassification error. *Journal of Econometrics*, 200(2):282–294, 2017.

Remco R Bouckaert. Properties of Bayesian belief network learning algorithms. In *Uncertainty Proceedings 1994*, pages 102–109. Elsevier, 1994.

John Bound, Charles Brown, and Nancy Mathiowetz. Measurement error in survey data. In *Handbook of econometrics*, volume 5, pages 3705–3843. Elsevier, 2001.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

Gerda Claeskens, Nils Lid Hjort, et al. Model selection and model averaging. *Cambridge Books*, 2008.

Diego Colombo and Marloes H Maathuis. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

Anthony C Constantinou. Bayesian artificial intelligence for decision making under uncertainty. *Engineering and Physical Sciences Research Council (EPSRC)*, 2018.

Anthony C Constantinou, Yang Liu, Kiattikun Chobtham, Zhigao Guo, and Neville K Kitson. Large-scale empirical validation of Bayesian Network structure learning algorithms with noisy data. *International Journal of Approximate Reasoning*, 131:151–188, 2021.

James Cussens. Bayesian network learning with cutting planes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 153–160, 2011.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Dorit Dor and Michael Tarsi. A simple algorithm to construct a consistent extension of a partially oriented graph. *Technicial Report R-185, Cognitive Systems Laboratory, UCLA*, 1992.

Maxime Gasse, Alex Aussem, and Haytham Elghazel. A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications*, 41(15):6755–6772, 2014.

Dan Geiger, David Heckerman, and Christopher Meek. Asymptotic model selection for directed networks with hidden variables. In *Learning in Graphical Models*, pages 461–477. Springer, 1998.

Dan Geiger, David Heckerman, Henry King, and Christopher Meek. Stratified exponential families: graphical models and model selection. *The Annals of statistics*, 29(2):505–529, 2001.

Yingyao Hu. Identification and estimation of nonlinear models with misclassification error using instrumental variables: A general solution. *Journal of Econometrics*, 144(1):27–61, 2008.

Jaime S Ide and Fabio G Cozman. Random generation of Bayesian networks. In *Brazilian symposium on artificial intelligence*, pages 366–376. Springer, 2002.

Steffen L Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, 19(2):191–201, 1995.

Junning Li and Z Jane Wang. Controlling the false discovery rate of the association/causality structure learned with the PC algorithm. *Journal of Machine Learning Research*, 10(2), 2009.

Andrew A Neath and Joseph E Cavanaugh. The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203, 2012.

Juan Miguel Ogarrio, Peter Spirtes, and Joe Ramsey. A hybrid causal search algorithm for latent variable models. In *Conference on Probabilistic Graphical Models*, pages 368–379, 2016.

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(2):121–129, 2017.

Richard Scheines and Joseph Ramsey. Measurement error and causal discovery. In *CEUR workshop proceedings*, volume 1792, page 1. NIH Public Access, 2016.

Marco Scutari. Bayesian network repository, 2020. `https://www.bnlearn.com/bnrepository/`.

Marco Scutari et al. Learning Bayesian Networks with the bnlearn R package. *Journal of Statistical Software*, 35(i03), 2010.

Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

Joe Suzuki. A construction of bayesian networks from databases based on an mdl principle. In *Uncertainty in Artificial Intelligence*, pages 266–273. Elsevier, 1993.

Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 673–678, 2003.

Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

Chirayu Wongchokprasitti. R-causal r wrapper for tetrad library, v1.2.1. `https://github.com/bd2kccd/r-causal/`, 2019.

Jiji Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896, 2008.

Kun Zhang, Mingming Gong, Joseph Ramsey, Kayhan Batmanghelich, Peter Spirtes, and Clark Glymour. Causal discovery with linear non-Gaussian models under measurement error: Structural identifiability results. In *UAI*, pages 1063–1072, 2018.