Bayesian network structure learning in the presence of latent variables

Kiattikun Chobtham

Bayesian Artificial Intelligence research lab,

Machine Intelligence and Decision Systems (MInDS) research group,

School of Electronic Engineering and Computer Science

Queen Mary
University of London

A Thesis submitted for the degree of

Doctor of Philosophy

DECLARATION

I, Kiattikun Chobtham, declare that the research included in this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, this is duly acknowledged, and my contribution indicated. Previously published material is also acknowledged. I further acknowledge any previously published material used in this work. It is affirmed that reasonable care has been exercised to ensure that the work is original and does not violate any UK law, infringe any third party's copyright or other Intellectual Property Rights, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Submission date: 12 July 2023

Examination date: 10 October 2023

Revised version: December 2023

The material presented in this thesis is based on the set of papers listed below.

LIST OF PUBLICATIONS

K. Chobtham and A. C. Constantinou. Bayesian network structure learning with causal effects in the presence of latent variables. In M. Jaeger and T. D. Nielsen, editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 101–112. PMLR, 23–25 Sep 2020.

K. Chobtham, A. C. Constantinou, and N. K. Kitson. Hybrid Bayesian network discovery with latent variables by scoring multiple interventions. *Data Mining and Knowledge Discovery*, volume, 37: pages 476–520, 2023. doi: 10.1007/s10618-022-00882-9.

K. Chobtham and A. C. Constantinou. Discovery and density estimation of latent confounders in Bayesian networks with evidence lower bound. In *Proceedings of the 11th International Conference on Probabilistic Graphical Models*, volume 186 of *Proceedings of Machine Learning Research*, pages 121–132. PMLR, 05–07 Oct 2022.

K. Chobtham and A. C. Constantinou. Tuning structure learning algorithms with out-of-sample and resampling strategies, *arXiv preprint* arXiv:2306.13932, 2023.

A. C. Constantinou, Y. Liu, K. Chobtham, Z. Guo, and N. K. Kitson. Large-scale empirical validation of Bayesian network structure learning algorithms with noisy data. *International Journal of Approximate Reasoning*, Vol. 131: pages 151–188, 2021. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2021.01.001.

A. C. Constantinou, Y. Liu, N. K. Kitson, K. Chobtham, and Z. Guo. Effective and efficient structure learning with pruning and model averaging strategies. *International Journal of Approximate Reasoning*, 151: pages 292–321, 2022.

N. K. Kitson, A. C. Constantinou, Z. Guo, Y. Liu, and K. Chobtham. A survey of Bayesian network structure learning. *Artificial Intelligence Review*, 2023. ISSN 1573-7462. Doi: 10.1007/s10462-022-10351-w.

A. C. Constantinou, N. K. Kitson, Y. Liu, K. Chobtham, A. Hashemzadeh, P. A. Nanavati, R. Mbuvha, and B. Petrungaro. Open problems in causal structure learning: A case study of COVID-19 in the UK, *Expert Systems with Applications*, Vol. 234, 2023, 121069, ISSN 0957-4174.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor Dr Anthony Constantinou, Head of Machine Intelligence and Decision Systems research group for guiding me throughout my four-year PhD journey. I am very grateful to him for his kindness, patience, and valuable advice in scientific thinking, effective communication, and my English writing, not only about the research but also about my professional career and life. Dr Constantinou is a talented researcher and lecturer, and he consistently provides prompt responses to my inquiries. I am very lucky to have had the opportunity to learn from him.

I am very grateful to my collaborator Dr Ken Kitson for his extremely helpful comments in our publications. Furthermore, I would like to extend my appreciation to my previous supervisor, Dr Athasit Surarerks, for providing a recommendation letter of support. A big thanks to my colleagues that made this journey more pleasant, Yang Liu and Buckley.

I would like to thank Thailand's Office of Civil Service Commission (OCSC) and Hydro-Informatics Institute for supporting me with a Royal Thai Government scholarship for my doctoral studies.

Finally, I would also like to express my gratitude to my parents, my sister and my best friends for their love and encouragement during my time in the UK.

ABSTRACT

A causal Bayesian Network (BN) is a probabilistic graphical model that captures causal or conditional relationships between variables, and enables causal reasoning under uncertainty. Causal reasoning via graphical representation in turn enables interpretability and full transparency in decision-making, and this makes causal BNs suitable for modelling critical real-world problems that require explainability, such as in healthcare, environmental sciences, government policy and economics.

Learning accurate causal structure from data represents a notoriously difficult task, and this difficulty increases with any imperfections present in the input data. For example, real data tend not to capture all relevant variables needed for causal representation, and these missing variables are referred to as hidden or latent variables. If some of the latent variables are latent confounders (i.e., missing common causes), they would confound the effect variables, thereby leading to spurious relationships in the learnt structure that could be misinterpreted as causal relationships. While the relevant literature includes structure learning algorithms that are capable of learning causal structure from data with latent variables, it is fair to say that accurate structural discovery from real data remains an open problem.

This thesis studies structure learning algorithms that recover graphical structure from data, and primarily focuses on the problem of latent variables. It investigates new solutions, including structure learning algorithms that learn from both observational and interventional data, approaches for density estimation that can be used to recover the underlying distribution of possible latent confounders, and techniques for hyperparameter optimisation of structure learning algorithms. The thesis explores this set of new approaches by applying them to a range of synthetic and real datasets of varying size, dimensionality, and data noise, and concludes by highlighting open problems and directions for future research.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACI	Ancestral Causal Inference
AIC	Akaike Information Criterion
ASP	Answer Set Programming
BDeu	Bayesian Dirichlet equivalent uniform
BDs	Bayesian Dirichlet sparse score
BIC	Bayesian Information Criterion
BN	Bayesian Network
BSF	Balanced Scoring Function
CCHM	Conservative rule and Causal effect Hill-climbing for MAG
cFCI	conservative rule Fast Causal Inference
CI	Conditional Independence
CIL	Constrained Incremental Learner
COmbINE	Causal discovery from Overlapping INtErventions
CPDAG	Completed Partially Directed Acyclic Graph
CPS	Candidate Parent Sets
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
EBIC	Extended BIC
ELBO	Evidence Lower Bound
FCI	Fast Causal Inference
FGS	Fast Greedy equivalence Search
FN	False Negative
FP	False Positive
GBN	Gaussian Bayesian Network
GES	Greedy Equivalence Search
GFCI	Greedy Fast Causal Inference
GIES	Greedy Interventional Equivalence Search
GLSL	Greedy Latent Structure Learner
HC	Hill-Climbing
HCLC-V	Hill-Climbing Latent Confounder search with VBEM
ILC-V	Incremental Latent Confounder search with VBEM
IMAPs	Independence MAPs
iss	imaginary sample size
JCI	Joint Causal Inference
KL	Kullback-Leiber divergence
LL	Log-Likelihood
MAG	Maximal Ancestral Graph
MAHC	Model Averaging Hill-Climbing
MAP	Maximum A Posteriori
MB	Markov Blanket
MCAR	Missing Completely At Random
MCMC	Markov Chain Monte Carlo
mFCI	majority rule Fast Causal Inference
mFGS-BS	majority rule and Fast Greedy equivalence Search with Bayesian Scoring
MI	Mutual Information
OCT	Out-of-sample Causal Tuning

OTSL	Out-of-sample Tuning for Structure Learning
P	Precision
PAG	Partial Ancestral Graph
PDAG	Partially Directed Acyclic Graph
poset	partial ordered set
R	Recall
RFCI	Really Fast Causal Inference
RFCI-BSC	Really Fast Causal Inference and Bayesian Scoring of Constraints
SAT	boolean SATisfiability instances
Sepsets	Separated sets
SEM	Structural Equation Model
SHD	Structural Hamming Distance
StARS	Stability Approach to Regularization Selection
TN	True Negative
TP	True Positive
VAE	Variational Autoencoder
VAEM	VAE for heterogeneous Mixed type data
VBEM	Variational Bayesian Expectation-Maximization

CONTENTS

Chapter 1

Introduction

Machine learning is rapidly evolving in both academic and industry settings. Black-box machine learning, and particularly deep learning, has proven to be effective in areas where causal representation may not be essential, such as in Natural Language Processing (NLP), machine vision, and sound information processing. However, the advancements in deep learning have also highlighted the limitations of black-box machine learning in providing interpretable solutions in critical real-world areas where decisions must be explained and justified. Causal models such as causal Bayesian Networks (BNs) proposed by Pearl (1988), do offer the interpretability that is necessary for transparency and explainability. A causal BN achieves this through graphical representation that enables quantitative reasoning under causal assumptions, where variables are represented by nodes and causal probabilistic dependencies between variables are represented by directed edges.

Pearl and Mackenzie (2018) argue that there are three progressive levels of reasoning needed for machines to achieve effective real-world decision making; namely predictive, interventional and counterfactual reasoning. They call this "*The ladder of causation*". Specifically, at Level 1, models are limited to establishing associational relationships and generating predictions that are based exclusively on those relationships. For instance, they can answer questions such as "*What symptoms are most likely to be observed in the presence of disease A*?". Moving up to Level 2, models at this stage incorporate a form of causal representation that enables them to address questions relating to interventions. For example, "*What would be the effect of taking drug A on symptoms B, considering that they are caused by disease C*?" Finally, at Level 3, this highest level of causal representation enables answering questions about causation that extend to counterfactual reasoning. For instance, they can respond to inquiries such as, "*Would the severity of my symptoms caused by disease C be reduced if I had opted for drug B instead of drug A*?". Pearl (2000) argues that only causal models, including causal BNs, can address inquiries related to all three levels of causation.

Over the past few decades, BNs have been widely used for decision making under uncertainty in diverse real-world applications. For instance,

- Thornley et al. (2012) utilised a causal BN to investigate the risk of cardiovascular disease;

- Constantinou et al. (2013) developed a causal BN to measure the performance of football teams and predict match outcomes;
- de Waal et al. (2016) used a causal BN to conduct a case study on rhino poaching;
- Constantinou and Fenton (2017) demonstrated the use of causal BNs in modelling investment decision making in the UK property market;
- de Zoete et al. (2019) employed a BN to illustrate the limitations of probability theory in legal reasoning, particularly in relation to legal paradoxes.

When BNs are applied in practice, the causal structure of these models is often determined by knowledge and expert judgments. However, knowledge elicitation requires access to expertise, which can be costly and time consuming. These limitations gave rise to algorithms that automatically recover the underlying causal structure from data. Nowadays, many applications of BN models rely on structure learning algorithms, some of which enable users to specify knowledge-based constraints that restrict or guide structure learning towards graphical structures that are consistent with prior expert knowledge (Castelo and Siebes, 2000).

1.1 Problem statement and Research hypothesis

The process of learning BNs from data is generally separated into two main tasks: structure learning and parameter learning. The former represents an unsupervised approach that determines the structure of a BN model, whereas the latter represents the process of learning the conditional distributions given the learnt structure. This thesis focuses on the problem of structure learning.

Learning causal structure from data is known to be an NP-hard problem where the number of possible graphs grows super-exponentially with the number of the variables. Moreover, large or dense networks tend to require large sample sizes to parameterise them with reasonable accuracy, and this adds further pressures to computational complexity which increases both with the number of the variables and the sample size. Moreover, these challenges are elevated when the input data are *imperfect* (Constantinou et al., 2021).

Structure learning algorithms rely on unrealistic assumptions that rarely, if ever, hold in practice. One such assumption is that the input data are noise-free and that they incorporate all possible causes. Because real data are rarely *perfect*, learning accurate causal models from real data remains a notoriously difficult task. This partly explains why BNs are yet to produce the level of real-world impact observed by some of the associational ML techniques, such as deep learning and reinforcement learning which excel in areas where causal representation is not necessary.

Not capturing all the relevant variables needed for accurate causal representation is an example of learning from *imperfect* data. This specific problem is often referred to as learning in the presence of unmeasured or latent variables, or under the assumption of causal insufficiency (Spirtes et al., 2001). A latent confounder, which is a special case of a latent variable, represents a variable missing from data that is a common cause of two or more observed variables, and tends to lead to spurious edges between observed variables that may be misinterpreted as causal relationships. This thesis investigates these challenges and focuses on improving structure learning accuracy, primarily in the presence of latent variables.

1.2 Structure of the thesis

The thesis is structured as follows:

a) **Chapter 2** provides preliminary information and covers past studies related to these preliminaries. It starts by covering the main features of BNs and then focuses on structure learning terminology, including descriptions of some of the main classes of learning and some of the algorithms underpinning them.
b) **Chapter 3** focuses on contributions that come from two published papers in which I am a co-author. The chapter starts by summarising the results published by Constantinou et al. (2021) where we investigate the impact of data noise on structure learning. The chapter includes the results of a model averaging structure learning algorithm we propose in (Constantinou et. al., 2022) for recovering graphical structures from noisy data. The material presented in this chapter provides a brief summary of these two publications.
c) **Chapter 4** focuses on a conference publication I led (Chobtham and Constantinou, 2020), where we propose a structure learning algorithm that recovers Gaussian BNs from data under the assumption of causal insufficiency. The material presented in this chapter primarily comes from the published paper.
d) **Chapter 5** describes another structure learning algorithm that focuses on learning discrete, rather than Gaussian, BNs from data under the assumption of causal insufficiency. Moreover, this algorithm is designed such that is capable of learning from multiple datasets that capture both observational and interventional data. The material presented in this chapter primarily comes from the relevant Journal publication I led (Chobtham et al., 2023).
e) **Chapter 6** presents a modified Variational Bayes method that can be combined with structure learning algorithms that predict latent confounders, to approximate the actual distributions of the predicted latent confounders. The material presented in this chapter comes from the relevant conference publication I led (Chobtham and Constantinou, 2022).
f) **Chapter 7** describes a tuning algorithm that can be paired with structure learning algorithms to optimise their hyperparameters. The material presented in this chapter comes from a paper I led that is currently under peer-review (Chobtham and Constantinou, 2023).
g) **Chapter 8**, the final chapter of this thesis, begins with a summary of concluding remarks and looks at possible directions for future research. This chapter ends by highlighting important open problems in causal structure learning as partly identified by this thesis and another study in which I am a co-author (Constantinou et al., 2023).

Credit co-authorship contribution statement

Kiattikun Chobtham	Constantinou et al., 2021	Constantinou et. al., 2022	Kitson et al., 2023	Constantinou et al., 2023
Conceptualization	✓			✓
Methodology	✓			✓
Data Curation				
Coding	✓			✓
Experiments	✓			✓
Analysis of results				
Visualization				
Software and repository				
Writing - Original Draft				
Writing - Review & Editing	✓	✓	✓	✓
Supervision				

Chapter 2

Background information and literature review

This chapter begins by providing necessary background information on BNs, and then moves to structure learning algorithms and methods for evaluating the discovery of causal or conditional dependency structures. The chapter ends with relevant literature review on these topics. Subsequent chapters provide additional, albeit more specialised, background information and literature review related to those chapters.

2.1 Bayesian networks

A causal BN is a generative model represented by a Directed Acyclic Graph (DAG) G, where nodes $\mathbf{X} = \{X_1, \dots, X_N\}$ represent random variables and directed edges represent dependencies or causal relationships between variables (Pearl, 1988). The dependencies between variables are described via conditional probabilities $P(X_i|\mathrm{parent}(X_i))$, where $\mathrm{parent}(X_i)$ is the set of parents of node X_i in the DAG. The joint distribution over all nodes is defined as the product of all conditional probabilities as follows:

$$P(X_1, \dots X_N) = \prod_{i=1}^{N} P(X_i|\mathrm{parent}(X_i))$$

Given a DAG G, we call *orientations* or *marks* at the ends of any edges which consist of arrowheads (>) and tails (-). Two nodes are *adjacent* if there is any type of edge between them. If $A \rightarrow B$ is present in G, A is called a *parent* of B and B is called a *child* of A. A *path* is a sequence of nodes X which nodes X_i and X_{i+1} are adjacent. For a *direct path*, X_i must be the parent of X_{i+1}. If there exists an indirect path from A to B, we classify A as an *ancestor* of B and B as a *descendant* of A. Node X is called a *collider* if two directed edges are entering X. If X is not a collider, X will be called a *non-collider*. A *v-structure* is an unshielded triple A, B and C where node C is the collider ($A \longrightarrow C \longleftarrow B$). Finally, the *Markov Blanket* of a node A, denoted as $\mathrm{MB}(A)$, encompasses the parents, children, and parents of children of A. It represents the smallest set of nodes that render A independent of all other nodes. In other words, $\mathrm{MB}(A)$ acts as a shield, protecting A from the influence of all other variables.

Definition 1 – d-separation: *Given a graph G, where A and B are two nodes in G and* $\boldsymbol{C}$ *is a set of nodes not containing A or B in G, A and B are d-connected given* $\boldsymbol{C}$ *($A \not\perp B \mid \boldsymbol{C}$) if every non-collider on paths between A and B is not a member of* $\boldsymbol{C}$ *or every collider on paths between A and B has a descendant in* $\boldsymbol{C}$. *If A and B are not d-connected given* $\boldsymbol{C}$, *then they are d-separated given* $\boldsymbol{C}$ *($A \perp B \mid \boldsymbol{C}$), or we say that there is no active path between A and B relative to* $\boldsymbol{C}$ *(Pearl, 1988).*

Inference in BNs is often described in terms of the following three causal classes:

a) **Serial connection (Causal chain):** As shown in Figure 2.1, in this scenario node A serves as the cause of node C, and node C serves as the cause of node B. Consequently, the influence of A on B is transmitted through C. However, if node C is observed, A has no longer influence on B. This prevents any active path between A and B given an observation on C. According to **Definition 1** of d-separation, we can deduce that A and B are d-separated given C (A ⊥ B | C).

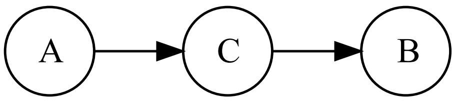


Figure 2.1 An example of a serial connection.

b) **Divergence connection (Common cause):** Figure 2.2 illustrates a scenario where node C acts as the cause of both A and B. If C is observed, then the path is blocked between the children of C. Consequently, we can infer that nodes A and B are d-separated given C (A ⊥ B | C).

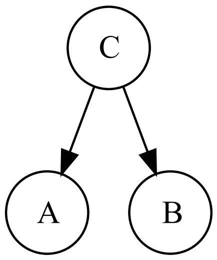


Figure 2.2 An example of a divergence connection.

c) **Converging connection (Common effect):** The relationship depicted in Figure 2.3 exhibits a notable characteristic known as "explaining away", which occurs when node C acts as the effect of both node A and node B. In this scenario, nodes A, B and C form a v-structure, with node C acting as the collider. According to **Definition 1** of d-separation, if node C is observed, then the path between its parents becomes active. Consequently, we can infer that A and B are conditionally dependent given C (A $\not\perp$ B | C).

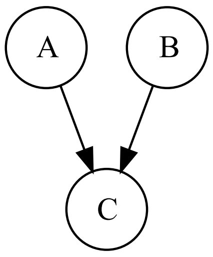


Figure 2.3 An example of a converging connection.

2.1.1 Markov property and Markov condition

BNs assume the Markov property and the Markov condition. It is through the Markov property that a DAG, representing a set of nodes, encodes a set of CI relationships on the joint probability distribution. We can formally define the Markov property as follows:

Definition 2: Markov property *Given a DAG G and a joint probability distribution P, we say that the distribution P satisfies:*

a) **The global Markov property** with respect to G if
$$\mathrm{A} \perp_{\mathrm{G}} \mathrm{B} \mid \mathbf{C} \Rightarrow \mathrm{A} \perp_{\mathrm{P}} \mathrm{B} \mid \mathbf{C}$$ for all disjoint nodes A, B and set of node $\mathbf{C}$.

b) **The local Markov property** if every node X in G is conditionally independent of its non-descendants given its parents.

c) **The Markov factorisation property** if P can be decomposed as follows:

$$\mathrm{P} = \prod_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{P}(\mathrm{X}_{\mathrm{i}} \mid \mathbf{Pa}(\mathrm{X}_{\mathrm{i}}));$$ where $\mathbf{Pa}(\mathrm{X}_{\mathrm{i}})$ are the parents of node X_{i}.

Using the Markov property, we can establish the relationship known as the **Markov condition** or **causal Markov condition**. This condition serves as a bridge principle that connects the causal interpretation of a DAG with probability distributions. The Markov condition can be described as follows:

Definition 3: Markov condition *Given a DAG G, every node is conditionally independent of its non-descendants given its parents in G.*

2.1.2 Intervention

As discussed in the introduction, BNs enable decision makers to model problems that go beyond prediction, such as by enabling the simulation of hypothetical interventions to estimate the effect of intervention. Pearl (2000) initially defined an intervention as an action that forces the state of a variable in a BN to a particular value. This action causes parts of the data generating process to change and induces an interventional distribution which might differ from the corresponding observational distribution. Pearl describes the difference between "*given that we see*" as observational data and "*given that we do*" as interventional data. Classic randomised controlled trials that capture treatments and their outcomes (Fisher, 1935) can be viewed as one kind of system suitable for generating interventional data. They typically involve randomly assigning patients into two groups, where the so-called treatment group is given the drug being tested, and the control group is given a placebo. If the outcome distribution differs significantly between the two groups, the difference is viewed as the effect of the drug.

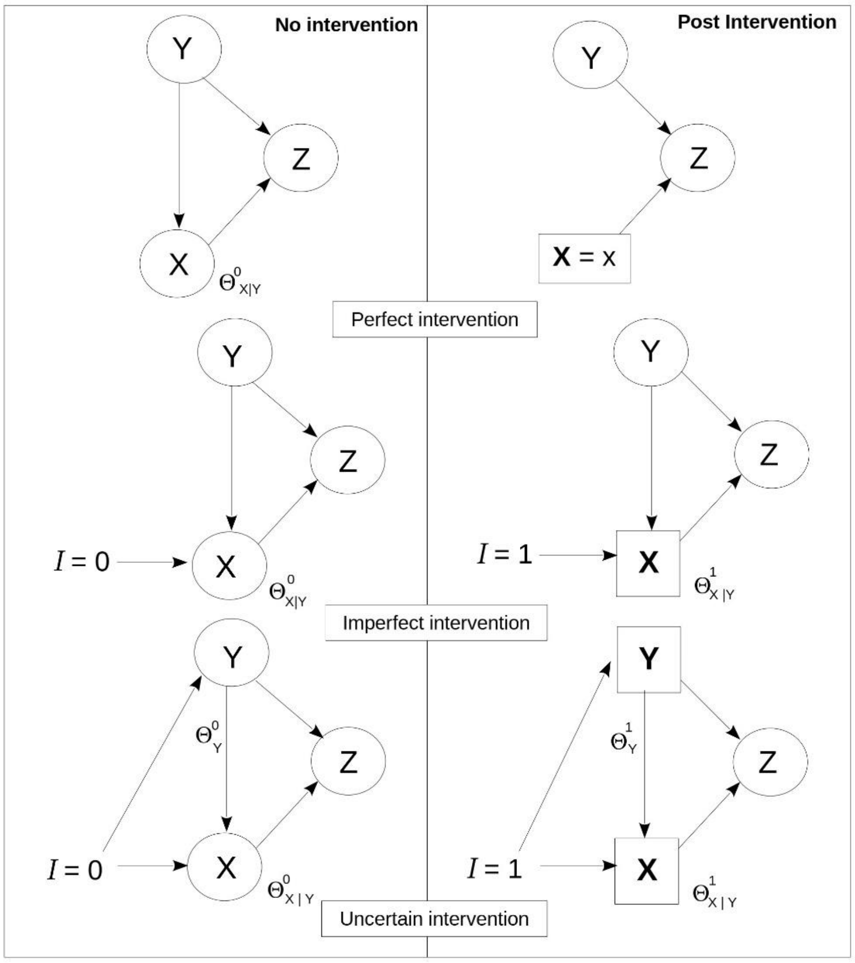


Figure 2.4 An illustration of the mechanisms of Perfect, Imperfect, and Uncertain interventions, where the square box represents the target node(s), $\Theta^0_{X|Y}, \Theta^0_Y$ are the parameters for nodes X and Y respectively when $I = 0$ (representing no intervention), and $\Theta^1_{X|Y}, \Theta^1_Y$ are the parameters for nodes X and Y respectively when $I = 1$ (representing an external imperfect or an uncertain intervention).

Figure 2.4 illustrates the three different intervention mechanisms by comparing the pre-intervention and post-intervention actions. Specifically, a **Perfect intervention** is what Pearl describes as *do-calculus* (do(X)) where the intervened variable is set to a given state with no uncertainty (Pearl, 2000). A perfect intervention modifies the original causal structure by rendering the intervened variable independent of its causes (also referred to as *graph surgery*). On the other hand, an **Imperfect intervention** or a mechanism change (Tian and Pearl, 2001) can be viewed as having external intervention nodes that act like switching parents I on an intervened variable X for each external intervention node. Specifically, $I = 1$ activates the intervention where the target node X is parameterised over Θ^1_X, whereas when $I = 0$ the intervention is deactivated and target node X is parameterised over Θ^0_X, which would imply no external influence on node X. Applications of imperfect intervention are often observed in healthcare studies, where medicine and therapeutic actions often have an imperfect effect in terms of treating symptoms or curing diseases (Rickles, 2009).

Lastly, an **Uncertain intervention** (Eaton and Murphy, 2007) represents the case where an external intervention I has multiple target nodes, or where the intervention on node X comes from more than one intervening route, as opposed to the imperfect intervention that assumes the relationship between intervention nodes and target nodes is one-to-one. Unlike perfect intervention, imperfect and uncertain interventions do not modify the graph and instead manipulate the node parameters.

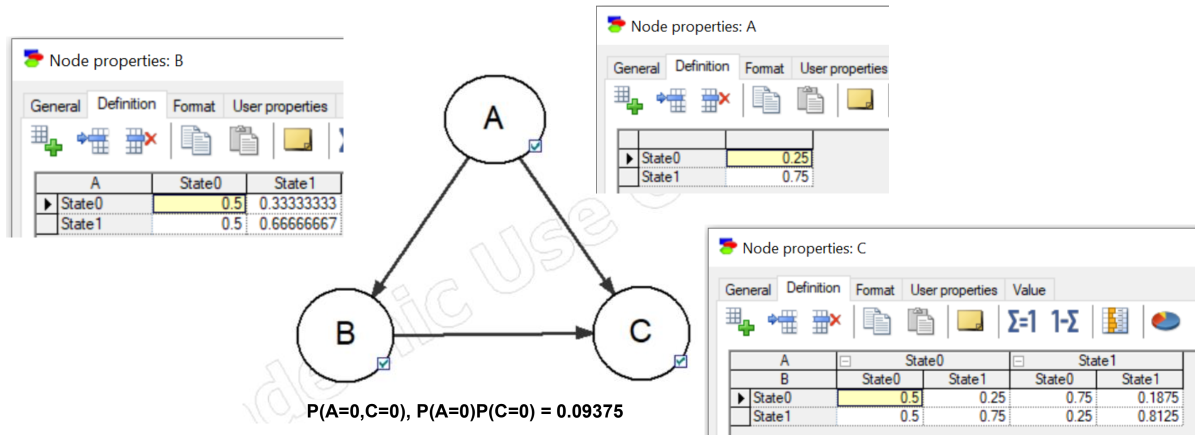
2.2 Structure learning

When BNs are applied in practice, their structure is determined by knowledge, structure learning, or a combination of both (Castelo and Siebes, 2000; Constantinou et al., 2016). This subsection focuses on algorithms that can be used to recover causal or conditional dependency structure from data. Structure learning methods tend to fall generally into two main categories: constraint-based learning and score-based learning. Additionally, there exist hybrid or other approaches, such as continuous rather than combinatorial optimisation, that are sometimes viewed as additional categories.

2.2.1 Constraint-based learning

The class of constraint-based learning primarily represents structure learning algorithms that rely on statistical CI tests to establish the CI relationships between variables and generate a graph skeleton based on observational data. Subsequently, the edge directions are determined by conditional dependence and other orientation rules to the constrained skeleton. The orientation phase depends on the accuracy of the skeleton and hence, any errors from the first phase are propagated to the orientation phase. Additionally, this class is often assumed to discover causal relationships under the assumption of causal faithfulness; an assumption that might not hold when working with real data. The faithfulness condition is defined as:

Definition 4: Faithfulness condition *Given a DAG G and a joint probability distribution P, we say that G and P satisfy the faithfulness condition if G entails all CI relationships in P, and all CI relationships in P are entailed by G, based on the Markov condition.*


Node properties: B
General Definition Format User properties
Node properties: A
General Definition Format User properties
Node properties: C
General Definition Format User properties Value
A
B
C

Node properties: B

A	State0	State1
State0	0.5	0.33333333
State1	0.5	0.66666667

Node properties: A

State0	0.25
State1	0.75

Node properties: C

A	State0		State1	
B	State0	State1	State0	State1
State0	0.5	0.25	0.75	0.1875
State1	0.5	0.75	0.25	0.8125

P(A=0,C=0), P(A=0)P(C=0) = 0.09375

P(A=0,C=1), P(A=0)P(C=1) = 0.15625

P(A=1,C=0), P(A=1)P(C=0) = 0.28125

P(A=1,C=1), P(A=1)P(C=1) = 0.46875

P(A,C)=P(A)P(C) for all values, and hence A $\perp$ C

Figure 2.5 An example where G is unfaithful to its joint probability distribution P.

When both the DAG G and the probability distribution P satisfy the faithfulness condition, we say that G and P are faithful to each other. In other words, G is a perfect map of

P. However, it is important to note that the faithfulness condition may not be intuitively determined at first glance. For instance, Figure 2.5 (Kitson et al., 2023) provides a simple illustration where the graph G exhibits two paths $A \rightarrow B$ and $B \rightarrow C$, which cancel each other's influence. Consequently, this creates an independence relationship between nodes A and C in the joint probability distribution P that is not implied by G. As a result, we can deduce that P and G are unfaithful in this case.

2.2.1.1 Conditional Independence (CI) tests

Learning causal structures from CI tests may lead to graphical outputs that are not a DAG, but which relate to a Markov equivalence class. We can formally define the Markov equivalence class as follows:

Definition 5: Markov equivalence class: *Two DAGs are Markov equivalent and belong to the same Markov equivalence class if they entail the same set of CI relationships.*

The DAGs depicted in Figures 2.1 and 2.2 entail the same CI relationship $A \perp B \mid C$. This implies that the DAGs cannot be distinguished by their CI relationships from observational data. As a result, these DAGs belong to the same Markov equivalence class. A Markov equivalence class of DAGs can be uniquely represented by a Completed Partially Directed Acyclic Graph (CPDAG). The CPDAG includes undirected edges that cannot be assigned a specific orientation based on observational data. For example, the CPDAG of both DAGs depicted in Figures 2.1 and 2.2 is $A - C - B$.

CI tests could be determined by different statistical functions suitable for measuring independence. Some of the commonly used such functions include:

a) Fisher’s z test

Fisher's z-test is a statistical test commonly used for continuous variables, specifically for linear Gaussian data (Fisher, 1921). It is defined as follows:

$$Z(\rho_{AB.\mathbf{C}}, n) = \frac{1}{2}\sqrt{n - |\mathbf{C}| - 3}\,\ln\left[\frac{|1 + \rho_{AB.\mathbf{C}}|}{|1 - \rho_{AB.\mathbf{C}}|}\right]$$

where $\rho_{AB.\mathbf{C}}$ represents the partial correlation coefficient between variables A and B given $\mathbf{C}$, n is denoted by the sample size, and $|\mathbf{C}|$ denotes the number of variables in set $\mathbf{C}$. The calculation of the partial correlation coefficient is given by:

$$\rho_{AB.\mathbf{C}} = \frac{\rho_{AB} - \rho_{A\mathbf{C}}\rho_{B\mathbf{C}}}{\sqrt{1 - \rho_{A\mathbf{C}}^2}\sqrt{1 - \rho_{B\mathbf{C}}^2}}$$

To determine whether a null hypothesis is rejected or accepted, a p-value is used as a test statistic or probability value, compared to a predefined significance threshold (α), typically set at 0.01, 0.05, or 0.1. If the p-value is less than α, the null hypothesis is rejected, indicating that variables A and B are conditionally dependent given $\mathbf{C}$. Conversely, if the p-value is greater than α, the null hypothesis is not rejected, and hence variables A and B are conditionally independent given $\mathbf{C}$.

b) Pearson's chi^2 test

The Pearson's chi^2 statistical test (Pearson, 1900) is a commonly used function for testing CI given discrete data. It assumes a null hypothesis that node A and node B are conditionally independent given the set of nodes **C**. The test produces a p-value of the test statistic, which is used to determine whether to reject or accept the null hypothesis. The significance threshold α serves as the hyperparameter of the Pearson's chi^2 test. The formula for the Pearson's chi^2 test is:

$$\chi^2 = 2\sum \frac{(\mathrm{n}_{abc} - \mathrm{m}_{abc})^2}{\mathrm{n}_{abc}}$$

where n_{abc} is the number of instances in the data D where A = a, B = b and **C** = c, $\mathrm{m}_{abc} = \frac{\mathrm{n}_{ac} \cdot \mathrm{n}_{bc}}{\mathrm{n}_c}$, and the calculation of the number of instances of n_{ac}, n_{bc} and n_c is analogous to that of n_{abc}. We will use Chi^2 interchangeably with the Pearson's chi^2 test for the rest of this thesis.

c) G^2 test

The G^2 statistical test (Sokal and Rohlf, 1981) is a likelihood ratio test commonly applied to assess CI with discrete variables. This test exhibits asymptotic equivalence to the Chi^2 test and is defined as follows:

$$G^2 = 2\sum \mathrm{n}_{abc} \ln \frac{\mathrm{n}_{abc}\mathrm{n}_c}{\mathrm{n}_{ac}\mathrm{n}_{bc}}$$

d) Mutual Information (MI) test

Shannon's Mutual Information (MI) was introduced as a measure of mutual dependence between two discrete variables (Cover and Thomas 2006). The mutual information between two nodes A and B is defined as:

$$\mathrm{MI}(A, B) = \sum_{a,b} \hat{p}(a,b) \ln \left[\frac{\hat{p}(a,b)}{\hat{p}(a)\hat{p}(b)}\right]$$

where $\hat{p}(a,b)$ refers to $\hat{p}(A = a, B = b)$ as the probability $p(a,b)$ derived from the maximum likelihood estimate. It is calculated as $\hat{p}(a,b) = \frac{\mathrm{n}_{ab}}{\mathrm{n}}$, where n is the total number of samples, and the process of calculating $\hat{p}(a)$ and $\hat{p}(b)$ is analogous to that of $\hat{p}(a,b)$. Consequently, conditional MI can be used for CI test, defined as:

$$\mathrm{MI}(A, B \mid \mathbf{C}) = \sum_{a,b,c} \hat{p}(a,b,c) \ln \left[\frac{\hat{p}(a,b,c)\hat{p}(c)}{\hat{p}(a,c) \cdot \hat{p}(b,c)}\right]$$

where $\hat{p}(a,b,c) = \frac{\mathrm{n}_{abc}}{\mathrm{n}}$ and the calculation of $\hat{p}(a,c)$, $\hat{p}(b,c)$ and $\hat{p}(c)$ is analogous to that of $\hat{p}(a,b,c)$. The significance threshold α serves the same purpose as in the Fisher's z-test, the Chi^2 test and the G^2 test, i.e., if $\mathrm{MI}(A, B \mid \mathbf{C})$ is greater than α, node A and node B are conditionally independent given **C**.

e) Shrinkage Mutual Information test (MI-sh)

James and Stein (1961) proposed a shrinkage estimate of MI for two random variables in the form of a regulariser, which they call the James-Stein-type shrinkage intensity λ. The conditional MI-sh test (Scutari and Brogini, 2012) between A and B given $\mathbf{C}$ is defined as the expectation of $\mathrm{MI}-\mathrm{sh}(A,B|\mathbf{C})$ with respect to the distribution of $\mathbf{C}$. As with the other CI functions, they use the significance threshold α to accept or reject the same null hypothesis. The MI-sh test is described as follows:

$$\mathrm{MI}-\mathrm{sh}(A,B|\,\mathbf{C}) = \sum_{a,b,c} p^{\mathrm{shrink}}(a,b,c)\log\left[\frac{p^{\mathrm{shrink}}(a,b,c)p^{\mathrm{shrink}}(c)}{p^{\mathrm{shrink}}(a,c)p^{\mathrm{shrink}}(b,c)}\right]$$

where:

$$p^{\mathrm{shrink}}(a,b,c) = \lambda\frac{1}{|A||B||\mathbf{C}|} + (1-\lambda)\hat{p}(a,b,c)$$

$$p^{\mathrm{shrink}}(a,c) = \lambda\frac{1}{|A||\mathbf{C}|} + (1-\lambda)\hat{p}(a,c)$$
$$p^{\mathrm{shrink}}(b,c) = \lambda\frac{1}{|B||\mathbf{C}|} + (1-\lambda)\hat{p}(b,c)$$
$$p^{\mathrm{shrink}}(c) = \lambda\frac{1}{|\mathbf{C}|} + (1-\lambda)\hat{p}(c)$$

where $|A|$, $|B|$ and $|\mathbf{C}|$ denote the number of states of variables A, B and the set of variables $\mathbf{C}$ respectively, and λ is the shrinkage intensity. Hausser and Strimmer (2009) proposed a closed-form estimator λ^* that employs James-Stein-type shrinkage making it highly efficient computationally. In the case of estimating a single parameter, λ^* is defined as:

$$\lambda^* = \frac{1-\sum_{k=1}^{V}(\hat{p}_k)^2}{(n-1)\sum_{k=1}^{V}(\frac{1}{V}-\hat{p}_k)^2}$$

where $\lambda^* = [0,1]$ is the shrinkage intensity, $\lambda^* = 0$ means no shrinkage and $\lambda^* = 1$ refers to full shrinkage, n is the sample size, and $\hat{p}_1 \dots, \hat{p}_V$ are the probabilities of a variable and $\sum_k \hat{p}_k$ =1.

2.2.1.2 Causally Insufficient Systems

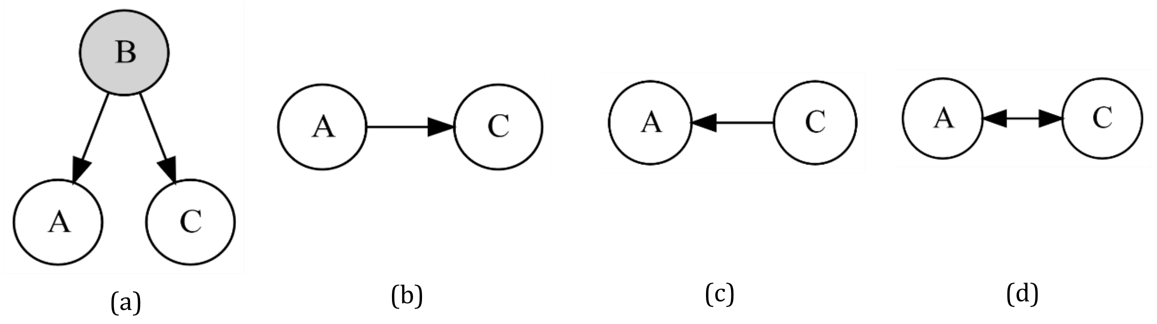


Figure 2.6 (a) Latent confounder B in grey causes A and C. **Definition 1** indicates that A and C are statistically dependent, leading to spurious directed edges in (b) and (c). The bidirected edge in the ancestral graph in (d) represents confounding.

In constraint-based learning, obtaining a DAG from data in the presence of latent variables can be problematic in the presence of latent confounders (Spirtes et al., 2001). Figure 2.6a presents a DAG where B is a latent confounder. Given **Definition 1**, A and C are statistically dependent. Therefore, in the absence of B, a spurious edge is typically discovered between A and C as illustrated in Figures 2.6b and 2.6c. Algorithms that account for latent confounders produce an ancestral graph that captures possible latent variables or latent confounders. Figure 2.6d presents an ancestral graph in which the bidirected edge indicates that the dependency between A and C may be due to confounding. Moreover, directed edges in ancestral graphs do not necessary represent direct causal relationships as in DAGs. Specifically, an ancestral graph is an extended version of a DAG under the assumption of causal insufficiency, in which the global Markov property offers the probabilistic interpretation that: if A and B are m-separated by $\mathbf{C}$, then A and B are conditionally independent given $\mathbf{C}$, where the m-separation definition in ancestral graphs aligns with the d-separation definition in DAGs.

A Maximal Ancestral Graph (MAG) (Richardson and Spirtes, 2000) is an ancestral graph where arcs indicate direct or ancestral relationships and bidirected edges represent confounding. A Partial Ancestral Graph (PAG) represents a set of Markov equivalent MAGs (Spirtes et al., 2001), in the same way a CPDAG represents a set of Markov equivalent DAGs, that entail the same set of CIs or m-separation criteria. A MAG can contain the following types of edges: —, →, and ↔. The undirected edge A— B indicates that A is an ancestor of B or a selection variable, and B is an ancestor of A or a selection variable. The selection variable indicates the presence of selection bias in the dataset. In this thesis, we do not explore selection bias and hence, the learnt MAGs or PAGs presented by the structure learning algorithms investigated will not include an undirected edge. Further, the directed edge A → B in a MAG or a PAG indicates parental or ancestral relationships, and the bidirected edge A ↔ B refers to the presence of a latent confounder where A and B are related but where neither A is an ancestor of B nor B is an ancestor of A. In a PAG, the variant mark (o) at the endpoint of edges indicates that the endpoint could be a tail (–) or an arrowhead (>) in the equivalence class of MAGs. For example, o→ in the PAG indicates that the edge can be either ↔ or → in the equivalent MAGs, whereas o—o indicates that the edge in the equivalent MAGs can be →, ← or ↔. Both MAGs and PAGs are acyclic graphs and do not allow the existence of almost directed cycles that may occur when A ↔ B is present and B is an ancestor of A (Richardson and Spirtes, 2000). Figure 2.7 illustrates an example of a DAG with latent variables L_1 and L_2, along with two examples of its Markov equivalent MAGs, and the PAG representing the Markov equivalence class of those MAGs (Chobtham and Constantinou, 2020).

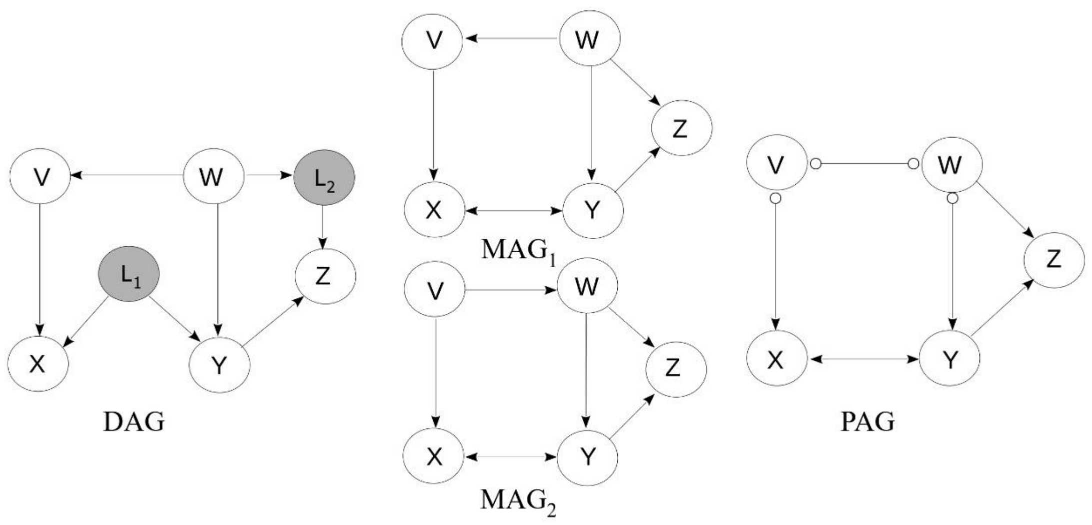


Figure 2.7 A causal DAG with observed variables $\{V, W, X, Y, Z\} \cup$ latent variables $\{L_1, L_2\}$ in grey, with two examples of Markov equivalent MAGs, and the Markov equivalent PAG of MAGs.

2.2.1.3 Constraint-based algorithms

Constraint-based algorithms can be divided into those which assume causal sufficiency and those which assume causal insufficiency. In describing these algorithms, we place a greater focus on algorithms that assume causal insufficiency, since they better fall within the scope of this thesis. We start with those that assume causal sufficiency, and some of the widely-used include:

a) **GS**: Grow-Shrink (GS) is a constraint-based algorithm that utilises the concept of Markov Blankets to reduce the number of CI tests required (Margaritis and Thrun, 1999). It identifies the Markov Blanket for each variable, determines the undirected skeleton of the graph, and orientates edges to produce a Partially Directed Acyclic Graph (PDAG) in which the only edges that are directed are those that are part of colliders.

b) **PC, CPC and PC-Stable:** The PC-Stable algorithm (Colombo and Maathuis, 2014) is a modified version of the classic constraint-based algorithm PC (Spirtes and Glymour, 1991) and produces a PDAG. PC-Stable addresses the variable order-dependency issue of PC by changing the order in which edge deletions are performed and by incorporating the v-structure phase from the CPC algorithm (Ramsey et al, 2006) (details are provided below). PC and its variant PC-Stable are considered the gold standard for benchmarking constraint-based learning algorithms.

c) **Inter-IAMB:** This is an enhanced variant of the IAMB algorithm (Tsamardinos, et al. 2003), employing an interleaved approach that combines the grown and shrink phases. The primary objective is to minimise the size of the Markov Blanket. By reducing the size of the Markov Blanket, Inter-IAMB achieves more precise results in the CI tests.

Many variants that are based on the above algorithms have been proposed under the assumption of causal sufficiency. However, some of the variants do assume causal insufficiency, and aim to recover a graph structure in the presence of latent variables. One of the most widely-used such constraint-based algorithms is Fast Causal Inference (FCI) (Spirtes et al., 2001). FCI modifies the PC algorithm (Spirtes and Glymour, 1991) such that it produces a PAG output, consistent with causal insufficiency considerations. Specifically, FCI first determines the adjacencies by employing CI tests to remove edges (dependencies) from a completed undirected graph in the adjacency phase, just like in PC. It then performs CI tests by checking all pairs of nodes A and B given an empty set to remove edges $A - B$ and progressively increasing the size of the conditioning sets or the separated sets (Sepsets) until a pre-defined Sepset size is reached. In this phase, the algorithm only considers Sepset members which are adjacent to A and B. Moreover, FCI makes use of the result of the Sepset obtained in this phase to be considered in the next v-structure phase. Next, the v-structure phase performs edge orientation given the graph skeleton. Specifically, if the Sepset for A and B does not contain C for an unshielded triple $A - C - B$, then the v-structure phase identifies it as the v-structure $A \longrightarrow C \longleftarrow B$. However, Sepsets in an ancestral graph can contain nodes which are not adjacent to A or B (Spirtes et al., 2001). The FCI algorithm uses complex strategies, such as Possible-D-Sep(A, B), to determine additional edges to remove. The v-structure orientation is subsequently repeated on this new skeleton. Finally, the FCI algorithm orientates some of the remaining undirected edges based on four orientation rules and by ensuring to that the creation of new v-structures is prevented.

Many modified versions of FCI have been published in the literature and include the augmented FCI (Zhang, 2008) which improves the orientation phase by extending the orientation rules of FCI from four to ten that are said to produce a sound and complete PAG. Others include the conservative rule FCI algorithm (cFCI) that incorporates CPC by Ramsey et al. (2006) to improve the edge orientation accuracy in the v-structure phase. Compared to FCI, cFCI performs additional CI tests on every pair of nodes A and B given on all subsets of all neighbours of A and B including C, for each unshielded triple $A - C - B$. The conservative rule in cFCI classifies each unshielded triple as either a definite v-structure, a definite non v-structure, or an ambiguous triple, e.g. if C is not in any Sepsets A and B, the conservative rule will classify the unshielded triple $A - C - B$ as a definite v-structure. Therefore, cFCI is more cautious about orientating edges than FCI; hence, the name "conservative".

Colombo and Maatthuis (2014) studied the impact of incorrect CI tests and found their outcome to be sensitive to the lexicographic ordering of the variable names – or on the order of the variables as read from data. To address this issue, they proposed PC-Stable and FCI-Stable by processing all the CI tests at each Sepset size, and removing edges at the end of – not during – the CI process. In the v-structure phase, FCI-Stable follows the approach adopted by cFCI by considering all the Sepsets of A and B in triple $A - C - B$ to decide whether it is a v-structure. Colombo and Maatthuis (2014) also found that the conservative rule was orientating only few of the v-structures and proposed the majority rule in the v-structure phase which can be viewed as a relaxed version of the conservative rule. They call this new variant mFCI. Specifically, in mFCI, the majority rule classifies each unshielded triple $A - C - B$ as:

a) A v-structure if C is in less than 50% of the Sepsets of A and B,
b) A non v-structure if C is in more than 50% of the Sepsets of A and B,
c) An ambiguous triple if C is in 50% of the Sepsets of A and B.

Lastly, the Really Fast Causal Inference algorithm (RFCI) was proposed by Colombo et al. (2011). This variant skips one adjacency phase and one v-structure phase in FCI, and therefore performs fewer CI tests. This modification makes the algorithm faster and more suitable to problems that involve thousands of variables, in exchange for a minor reduction in the accuracy of the learnt graph.

2.2.2 Score-based learning

Score-based methods can be viewed as a traditional machine learning approach that combines search with objective functions to identify the highest scoring graph. As previously mentioned in the introduction, the problem of structure learning is NP-hard, making an exhaustive search across all possible graphs impractical. To address this challenge, heuristic search techniques such as greedy search or hill-climbing search are often employed. However, these methods often get stuck in local optima. Exact search algorithms, such as Branch & Bound or Integer Linear Programming (ILP), guarantee the identification of the highest scoring graph within the search-space of graphs, but the search-space of these algorithms is generally restricted to a small maximum node in-degree to ensure reasonable computation times.

2.2.2.1 Objective scores

The objective scores used in score-based learning are usually score-equivalent, which means that they generate the same score for DAGs that are part of the same Markov equivalence class or CPDAG. The two most used score-equivalent objective functions are described below.

a) The Bayesian Dirichlet equivalent uniform (BDeu)

The BDeu score represents the Maximum A Posteriori (MAP) structure. It is a variant of BD and BDe scores, and assumes equivalent uniform priors. Importantly, these are decomposable scores where the total score of the graph represents the sum of the scores assigned to each of its nodes. A decomposable score is important for structure learning because most local scores can be reused, rather than recomputed, when exploring neighbouring graphs. The BD score was first introduced by Heckerman et al. (1994), under the assumption that the data follow a Dirichlet distribution. Pairing structure learning with BD as the objective function implies that the algorithm searches for a DAG G that maximises the posterior probability $P(G|D)$ given the data D. In this case, structure learning from data can be viewed as an optimisation problem to maximise $P(G|D) \propto P(G)\, P(D|G)$ where the highest posterior probability of a learnt graph G is approximated to the highest Log-Likelihood (LL) score:

$$\log P(G|D) = \log P(G) + \log P(D|G)$$

where $P(G)$ is the prior distribution over all DAGs. Because the search space of DAGs grows super-exponentially with the number of variables, it is impractical to specify informative priors for each DAG. For simplicity, the prior distribution is often taken to be uniform. The BD score can be computed as follows:

$$BD = \prod_{i=1}^{N} \prod_{j=1}^{q_i} \left[\frac{\Gamma(\Sigma_k \alpha_{ijk})}{\Gamma\left(\Sigma_k \alpha_{ijk} + \Sigma_k n_{ijk}\right)} \prod_{k=1}^{|X_i|} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})} \right]$$

where N is the number of variables, q_i is the number of possible combinations of values of the parents of node X_i (it is 1 if there is no parent), j is the index over the combinations of values of the parents of node X_i, $|X_i|$ is the number of states of node X_i, k is the index over the possible values of node X_i, Γ is the Gamma function, n_{ijk} is the total number of instances in data D where the parents of node X_i have the j^{th} combination of values, and α_{ijk} are the hyperparameters of the Dirichlet distribution. In BDeu, the hyperparameters are set to $\alpha_{ijk} = {}^{iss}/_{|X_i|q_i}$ where iss is the *imaginary sample size* that represents the user's prior belief about the impact of the prior distribution on the score. The study by Silander et al. (2007) suggests that reasonable hyperparameter values are $iss \in [1,20]$, where larger iss values tend to produce denser DAGs. Because the BDeu score produces a small value, it is computationally convenient to take its log value. Its closed-form expression is:

$$BDeu_{iss} = \sum_{i=1}^{N}\sum_{j=1}^{q_i}\left[\log\frac{\Gamma\left({}^{iss}/_{q_i}\right)}{\Gamma\left({}^{iss}/_{q_i} + \Sigma_k n_{ijk}\right)} + \sum_{k=1}^{|X_i|}\log\frac{\Gamma\left({}^{iss}/_{|X_i|q_i} + n_{ijk}\right)}{\Gamma\left({}^{iss}/_{|X_i|q_i}\right)}\right]$$

b) Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC)

Schwarz (1978) proposed BIC as a model-selection function to reduce the risk of model-overfitting by balancing the goodness-of-fit with model dimensionality. It is based on Occam's razor principle in that the simplest solution is usually the best solution. Like $BDeu_{iss}$, BIC is decomposable and score-equivalent, and is equally commonly used as the objective function in score-based structure learning. The general form of the score for discrete variables is expressed as:

$$BIC(G, D) = LL(G, D) - \frac{\log(n)}{2}F$$

where n is the sample size, $LL(G, D)$ denotes the LL of the data D given the graph G:

$$LL(G, D) = \log[\hat{p}(D|G)] = \sum_{i=1}^{V}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}}$$

and F is the complexity penalty represented by the number of free parameters of the model. It can be expressed as:

$$F = \sum_{i=1}^{V}(r_i - 1)q_i$$

In Akaike Information Criterion (AIC) (Akaike, 1974), the penalty term is just the number of free parameters in the score which is defined as:

$$AIC(G, D) = LL(G, D) - F$$

2.2.2.2 Score-based algorithms

Some of the state-of-the-art score-based algorithms that assume causal sufficiency include:

a) **FGS:** This is an efficient version, proposed by Ramsey (2015), of the score-based Greedy Equivalence Search (GES) algorithm proposed by Chickering (2003). It improves the efficiency of GES through parallelisation and caching scores. FGS and GES search for Markov equivalence classes of DAGs instead of the entire DAG space, leading to polynomial time complexity. They consist of two learning phases referred to as the forward and backward search phases. In the forward phase, the learning process begins with an empty graph. At each iteration, the graph is explored by adding the edge that maximises the objective score. In the backward phase, edges are removed until no further edge removals increase the objective score.

b) **GOBNILP**: The Integer Linear Programming (ILP) algorithm by Cussens (2011) offers exact learning by dividing structure learning into two phases. The first phase computes the scores for Candidate Parent Sets (CPSs), whereas the second phase optimally assigns parents to each node ensuring acyclicity. GOBNILP guarantees to return the graph with the highest score within the given search-space of graphs, but the search-space is often restricted to a low maximum in-degree due to computational complexity considerations.

c) **HC**: Hill-Climbing (HC) greedily searches the search-space of DAGs and returns the DAG that maximises a given objective score (Heckerman et al., 1994; Scutari et al., 2018). It starts from an empty graph and iteratively performs local moves such as arc additions, deletions, or reversals to improve the graph's score until a local maximum is reached.

d) **NOTEARS:** NOTEARS is a continuous optimisation algorithm that formulates a score-based algorithm to an equality-constrained problem with an acyclicity constraint (Zheng et al., 2018). Originally designed for continuous data, it also has applicability to ordinal discrete data.

e) **TABU**: This is extension of the HC algorithm (Bouckaert, 1995; Scutari et al., 2018) that allows the exploration of lower-scoring local moves that are likely to help the algorithm escape from some local maxima. It also avoids revisiting previously encountered DAGs, promoting exploration of new regions in the DAG space.

f) **WINASOBS**: An ordering-based algorithm by Scanagatta (2018) that is similar to ILP in terms of the two learning phases, but employs a simplified objective score called BIC*, and stronger pruning that does not guarantee exact learning. It is an approximate learning algorithm applicable to thousands of nodes.

As with constraint-based algorithms, some score-based algorithms also perform structure learning under the assumption of causal insufficiency – but these score-based variants are recently proposed and hence, few of them are available and are restricted to linear Gaussian distributions.

g) **GSMAG:** Possibly the first score-based variant that assumes causal insufficiency is the GSMAG algorithm that uses greedy search to discover structures from data under the assumption the input data are continuous and normally distributed (Triantafillou and Tsamardinos, 2016). GSMAG employs a variant of BIC as the objective function, suitable for discovering MAG structures (see subsection 4.1).

h) **BB:** Rantanen et. al (2021) recently proposed the Branch-and-Bound algorithm (BB) for an exact score-based algorithm in the search-space of MAGs. BB employs dynamic programming and the branch-and-bound technique in conjunction with the BIC score for MAGs as in GSMAG.

2.2.3 Hybrid learning

Hybrid learning algorithms combine techniques from both constraint-based learning and score-based learning. This methodology typically involves utilising constraint-based approaches in an initial restrictive phase to limit the search space. Subsequently, objective scores are employed in a maximisation phase to identify the highest scoring graph while considering pairs of nodes that are constrained based on the outcomes of the restrictive phase. Some commonly used hybrid algorithms that assume causal sufficiency include:

a) **H2PC**: Proposed by Gasse et al. (2014), H2PC combines the strengths of HPC and HC. HPC is an ensemble constraint-based algorithm that consists of three individual learners, each focusing on learning parents and children sets (PC learner). By integrating these learners, HPC aims to enhance the overall performance and reliability of the PC learner.

b) **MMHC**: The Max-Min Hill-Climbing (MMHC) algorithm integrates principles from local-learning, constraint-based learning, and score-based learning. It starts by building a skeleton graph and then applies greedy hill-climbing search to determine the edge orientations or that skeleton (Tsamardinos et al., 2006). Renowned for its effectiveness in high-dimensional data, MMHC often serves as baseline for evaluating other structure learning methods.

c) **SaiyanH**: This algorithm starts with CI tests that are used to produce a skeleton graph that can be viewed as a denser version of the maximum spanning tree. It then uses some of the results from CI, in conjunction with an objective score and the effect of intervention, to determine the orientation of those edges. The DAG is then given as an input to the TABU algorithm with the restriction not to remove edges that would lead to disjoint graphical fragments. This restriction ensures full propagation of evidence when the learnt structure is converted into a BN model (Constantinou, 2020).

Hybrid algorithms that perform structure learning under the assumption of causal insufficiency include:

d) **GFCI**: This algorithm by Ogarrio et al. (2016) combines the score-based FGS (Ramsey, 2015) with the orientation rules of the constraint-based FCI. GFCI starts by obtaining the dependencies from the learnt CPDAG returned by FGS, and performs CI tests on those dependencies to remove potential false positive edges. The result of this process is a skeleton. The orientation rules of FCI are then used to orientate some of those edges and to produce a PAG. Due to various choices of CI tests and objective scores, GFCI can work with both discrete and continuous variables.

e) **M^{3}HC**: This algorithm by Tsirlis et al. (2018) produces a MAG by incorporating a constraint-based learning as the restriction phase to the GSMAG algorithm. M^{3}HC assumes the data are continuous and normally distributed. Tsirlis et al. (2018) showed that hybrid algorithms such as M^{3}HC and GFCI demonstrate better performance over other relevant constraint-based algorithms.

f) **RFCI-BSC**: Jabbari et al. (2017) introduced RFCI-BSC as a model averaging variant of RFCI, that generates multiple potential models and returns the PAG with the highest probability as the preferred graph. However, the algorithm's process involves bootstrap sampling, where multiple datasets are created through resampling with replacement, which makes the algorithm non-deterministic. CI tests from RFCI are then applied to each of those datasets. This non-deterministic nature of RFCI-BSC necessitates running the algorithm multiple times and obtaining an average of the results to ensure reliable outcomes.

2.2.4 Evaluating structure learning algorithms

Structure learning algorithms can be evaluated in two different ways. Firstly, graphical accuracy metrics can be used to measure how close the learnt graph is to the true graph. This approach, however, assumes access to the ground truth graph, and therefore is only applicable to synthetic experiments. In real cases, the ground truth naturally remains unknown and hence, model-selection scores such as the BIC and BDeu (see subsection 2.2.2.1), amongst other domain-specific approaches, are often used to judge the validity of the learnt graph.

Three metrics are commonly used to assess the graphical accuracy of the learnt graphs. These are:

a) **F1:** It represents the harmonic mean of Precision (P) and Recall (R). F1 ranges from 0 to 1, where a higher F1 score represents better performance. The F1 score is measured as follows:

$$\mathrm{F1} = 2\frac{\mathrm{PR}}{\mathrm{P} + \mathrm{R}}$$

where $\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ and $\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$, and TP, FP and FN refer to the number of true positive, false positive, and false negative edges in a learnt graph compared to a true graph respectively.

b) **SHD**: The Structural Hamming Distance (SHD) metric is the most used metric in literature. It counts the number of steps needed, in terms of edge insertions, deletions, and reversals, to convert the learnt DAG to the true DAG (Tsamardinos, 2006). A lower SHD score represents better performance, and it is defined as:

$$\mathrm{SHD} = \mathrm{FN} + \mathrm{FP}$$

c) **BSF**: The Balanced Scoring Function (BSF) score (Constantinou, 2019) considers all four confusion matrix parameters (TP, TN, FP and FN) to balance the score relative to the density of the true graph. The BSF score is defined as:

$$\mathrm{BSF} = 0.5 \times \left(\frac{\mathrm{TP}}{\mathrm{a}} + \frac{\mathrm{TN}}{\mathrm{i}} - \frac{\mathrm{FP}}{\mathrm{i}} - \frac{\mathrm{FN}}{\mathrm{a}}\right)$$

where a is the number of edges in the true DAG, i is the number of independencies in the true DAG, $\mathrm{i} = \frac{\mathrm{N(N-1)}}{2} - \mathrm{a}$, TN is the number of true negative edges and N is the number of variables. The BSF score ranges from -1 to 1, where 1 corresponds to a perfect match between learnt and true graphs, 0 represents a score equivalent to that obtained from an empty or a fully connected graph, and -1 corresponds to the worst possible mismatch (i.e., the reverse of the true graph).

Chapter 3

Structure learning with imperfect data

This chapter discusses two papers that I co-authored. It begins by summarising the findings of Constantinou et al. (2021), which examine how data noise affects structure learning. The chapter also presents the results of our proposed model averaging structure learning algorithm, described in Constantinou et al. (2022), for recovering graphical structures from noisy data. The content of this chapter provides a concise overview of these two publications.

The structure learning algorithms introduced in the literature assume that the distributions of the input data reflect the true distributions of data generating system. Moreover, each algorithm relies on a set of assumptions about the input data, and tends to be evaluated with clean synthetic data (Scutari et al., 2019). However, it is widely acknowledged that the synthetic performance of structure learning algorithms tends to overestimate their real-world performance, although the extent of this overestimation remains unknown. This chapter examines how imperfect data influence structure learning performance, and how structure learning may be able to account for these imperfections in the data.

This chapter is organised as follows: subsection 3.1 describes the case studies, subsection 3.2 describes the methodology we followed to generate imperfect data, subsection 3.3 covers the structure learning algorithms evaluated, subsection 3.4 presents the results, subsection 3.5 describes and evaluates a new structure learning algorithm that assumes the presence of data noise, and we provide our concluding remarks in subsection 3.6.

3.1 Case studies

We consider the six discrete data case studies, available in the Bayesys repository (Constantinou et al., 2020), whose properties are provided in Table 3.1. Three of the case studies represent well-established examples from the BN structure learning literature, whereas the other three represent new cases and are based on recent BN real-world applications. These are:

a) **Asia:** a small network that captures the relationships between a visit to Asia, tuberculosis and lung cancer (Lauritzen and Spiegelhalter, 1988).
b) **Sports:** a small network that measures the effect of ball possession in football matches, on shots generated and goals scored (Constantinou et al., 2013).
c) **Property:** a medium-size network for investment decision making in the UK property market (Constantinou and Fenton, 2017).
d) **Alarm:** a medium-size network of an alarm notification system for patients in a hospital intensive care unit (Beinlich et al., 1989).
e) **ForMed:** a large network modelling the risk of violent reoffending in mentally ill prisoners before and after release or discharge (Constantinou et al., 2015).
f) **Pathfinder:** a very large network for diagnosis of lymph-node diseases (Heckerman et al., 1992).

Network size	Network	Variables	Edges	Max in-degree	Free parameters
Small	Asia	8	8	2	18
	Sports	9	15	2	1,049
Medium	Property	27	31	3	3,056
	Alarm	37	46	4	509
Large	ForMed	88	138	6	912
Very Large	Pathfinder	109	195	5	71,890

Table 3.1 The properties of the six real-world networks.

3.2 Generating imperfect data

Each of the six networks were used to generate synthetic data, with and without synthetic noise. We generated 16 different categories of input data per case study, and assume five different sample sizes for each of these 16 datasets (0.1k, 1k, 10k, 100k, and 1000k samples). The 16 categories are depicted in Table 3.2, and are generated as follows:

a) **No noise** (N): This represents the standard case of clean synthetic data.

b) **Missing values** (M): In this scenario, the datasets are modified to include missing data values that are Missing Completely At Random (MCAR). We explore two different scenarios: a) that each individual data value has a probability of 5% to become missing (dataset denoted as M5), and b) a probability of 10% to become missing (denoted as M10). Because the algorithms tested assume complete data as input, we subsequently replaced each missing data value with a new state called 'missing', indicating missingness.

c) **Incorrect values** (I): Where each data value has 5% (I5) or 10% (I10) risk to be replaced with an incorrect value, where the new value comes from the set of other possible values observed in each variable.

d) **Merged states** (S): Where 5% or 10% of the variables (both cases tested) have two of their states merged into one. For example, a variable with states $\{a, b, c\}$ would have two random states, such as a and b, both modified into a new state ab. This assumption aims to approximate the performance of the algorithms when applied to real datasets where some of the data variables have had their number of states decreased in an effort to reduce the dimensionality of the model.

e) **Latent variables** (L): Where approximately 5% or 10% of the variables (both cases tested) are randomly removed from the dataset. This assumption aims to approximate the performance of the algorithms when applied to datasets that incorporate latent variables.

f) **Combo** (c): This category represents dual combinations of the noisy categories described above (denoted as cMI, cMS, cML, cIS, cIL, and cSL), plus the combination of all four categories of data noise (cMISL). Because these experiments incorporate multiple types of noise, we chose the rate of 5% as the default rate of noise for each type of noise incorporated into a dataset. If 5% is not possible due to limited data, then the rate of 10% is chosen.

Experiment	**No noise**	**Missing values**		**Incorrect values**		**Merged states**		**Latent variables**	
		5%	**10%**	**5%**	**10%**	**5%**	**10%**	**5%**	**10%**
N	✓								
M5		✓							
M10			✓						
I5				✓					
I10					✓				
S5						✓			
S10							✓		
L5								✓	
L10									✓
cMI		✓		✓					
cMS		✓				✓			
cML		✓						✓	
cIS				✓		✓			
cIL				✓				✓	
cSL						✓		✓	
cMISL		✓		✓		✓		✓	

Table 3.2 The 16 experiment codes for different types of noise where N denotes no noise, M represents missing values, I represents incorrect values, S represents merged states, L represents latent variables, and c represents combo.

3.3 Structure learning algorithms and evaluation setup

We investigate the impact of data noise in terms of how the different imperfect datasets influence the structure learning performance of different algorithms. We consider 15 structure learning algorithms from all three classes of learning, all of which have already been described in subsections 2.2.1.3, 2.2.2.2, and 2.2.3. Each algorithm is tested with their default hyperparameter settings as implemented in structure learning software or packages listed in Table 3.3. The default hyperparameters are selected under the assumption that this is how most users would employ these algorithms in real-world settings, given that there is no guidance on how and when we should be changing the value of these hyperparameters.

Because of the large number of the experiments, we restrict runtime to six hours per experiment. Algorithms that exceed the runtime limit are assigned the lowest rank for that particular experiment. We evaluate structure learning performance by comparing the learnt graph to the ground truth, and we use the SHD, BSF and F1 metrics to do this (refer to subsection 2.2.4). While these algorithms are often compared in terms of how accurately they recover the true CPDAG, in this set of experiments we measure them in terms of how well they recover the true DAG. This is because the purpose of these experiments is to investigate the usefulness of these algorithms in real-world settings where we tend to require CBNs (i.e., a DAG) and hence, we would like the assessment to be driven by how well the algorithms achieve this objective, rather than driven by what some of the algorithms, or implementations of the algorithms, assume or can and cannot do. Moreover, when it comes to causally insufficient experiments (i.e., those which incorporate latent variables), we assess the learnt graphs with respect to the ground truth MAG. Table 3.4 presents the penalty weights assumed by the graphical metrics.

Algorithm	Learning class	Software	Programming Language	Reference
PC-Stable	Constraint-based	Tetrad	Java	(Wongchokprasitti, 2019)
FGS	Score-based			
FCI	Constraint-based			
GFCI	Hybrid			
RFCI-BSC	Hybrid			
Inter-IAMB	Constraint-based	bnlearn	R	(Scutari, 2019)
MMHC	Hybrid			
GS	Constraint-based			
HC	Score-based			
TABU	Score-based			
H2PC	Hybrid			
SaiyanH	Hybrid	Bayesys	Java	(Constantinou, 2020)
GOBNILP	Exact score-based	GOBNILP	C++	(Cussens, 2011)
NOTEARS	Score-based	Source code	Python	(Zheng et al., 2018)
WINASOBS	Score-based	BLIP	The BLIP software	(Scanagatta, 2017)

Table 3.3 The properties of the 15 structure learning algorithms considered for evaluation.

True edge	Learnt edge	Penalty	Reasoning
A → B	A → B, A o→ B	0	Complete match
A → B	A ↔ B, A − B , Ao−oB, A ← B, A←o B	0.5	Partial match
A → B	A B	1	No match
A ↔ B	Any edge/arc	0	Latent confounder
A B	A B	0	Complete match
A B	Any edge/arc	1	Incorrect dependency discovered

Table 3.4 The penalty weights used for evaluation, where o-o and o→ are learnt edges by structure learning algorithms under the assumption of causal insufficiency.

3.4 Results

Table 3.5 presents the average ranked performance, the overall ranked performance, and the relative performance of each structure learning algorithm in terms of both the F1 and SHD scores, with and without data noise, averaged over all six different cases, five sample sizes, and 15 noisy experiments (for the noisy case).

In this set of results, we note a few interesting observations that relate to the ranking of the algorithms and how that might be sensitive to the noise in the data. TABU, which tops all three rankings under clean data, loses significant ground against the other algorithms in the presence of data noise where it ranks 2nd overall by the F1 and BSF metrics, and 4th overall by SHD. In contrast, the HC algorithm, which all metrics ranked 2nd under clean data, ranks 1st by F1 and BSF, and 2nd by SHD in the presence of data noise. This is an interesting observation because TABU is an improved search version of HC that escapes some of the suboptimal search regions in which HC has the tendency to get stuck in. This result can only suggest that data noise has misled TABU into performing escapes from a local maximum into regions that may better fit the noisy input data but which further deviate from the true graph.

Algorithm	**Average rank**			**Overall rank**			**Average rank**			**Overall rank**		
	Clean	**Noisy**	**Δ**	**Clean**	**Noisy**	**Δ**	**Clean**	**Noisy**	**Δ**	**Clean**	**Noisy**	**Δ**
			F1						**SHD**			
FCI	7.7	8.67	-1	9	11	-2	6.57	8.67	-2.1	7	12	-5
FGS	7.5	7.15	0.35	8	8	0	7.83	7.12	0.71	10	8	2
GFCI	6.87	7.26	-0.4	7	9	-2	6.87	6.91	-0	9	7	2
GS	11.87	11.7	0.12	14	15	-1	10.43	9.54	0.89	13	13	0
H2PC	6.13	5.66	0.47	5	5	0	5.1	4.96	0.14	3	3	0
HC	3.63	3.6	0.03	2	1	1	4.77	4.92	-0.2	2	2	0
GOBNILP	4.8	5.17	-0.4	3	3	0	6.43	6.72	-0.3	5	6	-1
Inter-IAMB	10	9.79	0.21	12	12	0	8.6	7.82	0.78	12	9	3
MMHC	7.77	6.51	1.26	10	6	4	6.47	4.66	1.81	6	1	5
NOTEARS	12	11.7	0.35	15	14	1	13	12.83	0.17	15	15	0
PC-Stable	8.1	7.59	0.51	11	10	1	6.83	7.87	-1	8	10	-2
RFCI-BSC	11.5	11.5	0	13	13	0	10.9	11.05	-0.2	14	14	0
SaiyanH	5.33	5.27	0.06	4	4	0	8	7.87	0.13	11	11	0
TABU	3.27	3.62	-0.4	1	2	-1	4.43	4.99	-0.6	1	4	-3
WINASOBS	6.3	6.54	-0.2	6	7	-1	5.87	5.49	0.38	4	5	-1
			BSF									
FCI	7.67	8.23	-0.6	9	11	-2						
FGS	7.1	7.37	-0.3	8	8	0						
GFCI	6.97	7.6	-0.6	6	10	-4						
GS	11.9	11.68	0.22	14	14	0						
H2PC	6.97	6.26	0.71	6	5	1						
HC	3.17	3.03	0.13	2	1	1						
GOBNILP	4.13	4.35	-0.2	3	3	0						
Inter-IAMB	10.43	9.98	0.45	12	12	0						
MMHC	8.6	7.59	1.01	11	9	2						
NOTEARS	12	12.51	-0.5	15	15	0						
PC-Stable	8	7.15	0.85	10	7	3						
RFCI-BSC	11.47	11.54	-0.1	13	13	0						
SaiyanH	4.77	5.16	-0.4	4	4	0						
TABU	3.1	3.13	-0	1	2	-1						
WINASOBS	6.17	6.77	-0.6	5	6	-1						

Table 3.5 The average and overall ranked performance for each algorithm over all case studies and sample sizes, and over all the 15 noisy-based experiments, as determined by each of the three metrics, where Δ represents the relative difference in performance compared to noise-free experiments N. Green and red rankings indicate improved and decreased structural accuracy in the presence of data noise, relative to the corresponding noise-free experiments.

The GOBNILP algorithm, which is the only exact learning algorithm tested in this set of experiments, has lost some ground in relative performance but not enough to alter its ranking. This result is consistent with that of TABU on the basis that data noise appears to distort model fitting which in turn has a negative effect on algorithms that seek the highest, or close to the highest, model-selection scores across the search-space of graphs.

Another interesting observation involves MMHC, which is the only algorithm that shows significant gains in performance across all the three metrics. Specifically, MMHC ranks 6th, 1st, and 9th in terms of F1, SHD, and BSF metrics respectively in the presence of data noise, up from 10th, 6th, and 11th with clean data.

On the other hand, FCI is the algorithm with the highest loss in relative performance. Conversely, FCI experiences the most substantial decline in relative performance compared to other algorithms. On the other hand, and rather surprisingly, the algorithms designed to account for latent variables during structure learning, such as the FCI, GFCI and RFCI-BSC, did not improve their performance relative to other algorithms under experiments which involve the reconstruction of the true MAG (experiments which incorporate code 'L').

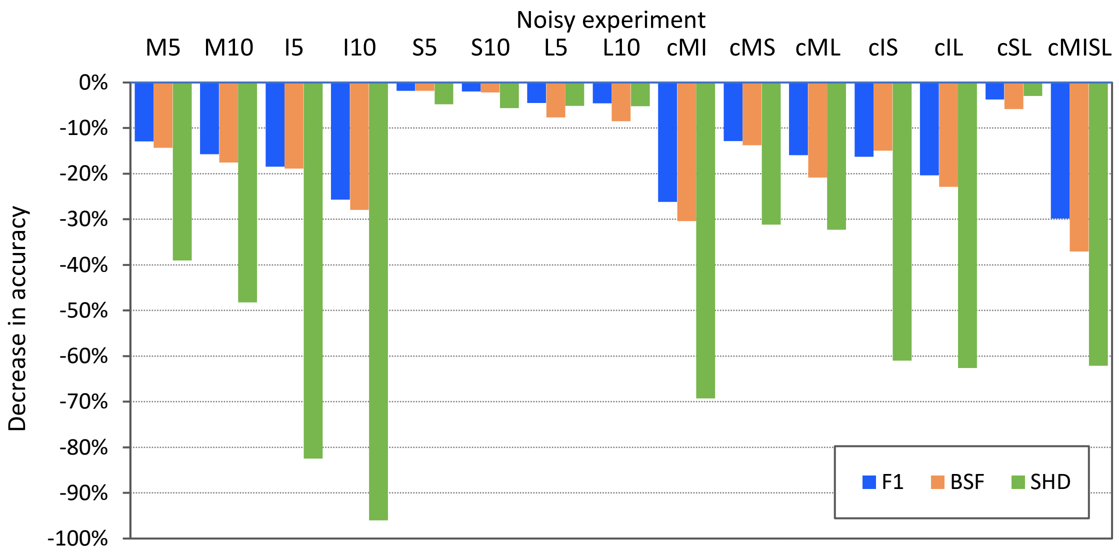


Figure 3.1 The overall decrease in accuracy of F1 and BSF and the corresponding increase in SHD are observed across all algorithms in each noisy experiment. These observations are made in comparison to the results of the experiment N conducted with clean data.

Figure 3.1 illustrates the overall decline in accuracy for all algorithms across each noisy experiment, and relative to noise-free experiments N. The findings highlight some inconsistencies in the conclusions drawn from the F1 and BSF metrics compared to the SHD metric. For example, the SHD score leads to the counterintuitive conclusion that experiments I5, I10, cMI, and cIL, have decreased structure learning performance more than experiment cMISL which incorporates all types of data noise as well as a higher total rate of data noise. On the other hand, the F1 and BSF metrics correctly identify that cMISL has had the largest negative impact on structure learning performance, as might be expected.

According to the F1 and BSF metrics, the overall results suggest that data noise of types S and L have had a relatively minor impact on structure learning performance. However, it should be noted that the results from experiments that incorporate L are based on the reconstruction of the true MAGs which incorporate a lower number of variables compared to the true DAGs used in experiments that do not incorporate L, and the difference in the number of variables in the data influences the result generated by the metrics. Conversely, data noise of types M and I have had a much stronger impact on decreasing the performance of the algorithms. Combining all four types of noise into a single dataset (experiment cMISL), which might better approximate real data, leads to the highest negative impact on structure learning performance.

3.5 Model averaging and pruning strategies for structure learning with imperfect data

In the previous subsections we studied the impact of data noise on structure learning. We show that some algorithms are less sensitive, and sometimes react differently, to a given type of data noise than others. Importantly, we found that non-exact or simple learners are more resilient to data noise, compared to exact or more sophisticated non-exact learners. For example, less sophisticated non-exact learners such as HC perform better in the presence of data noise compared to more sophisticated non-exact learners such as TABU which, in theory, TABU is an improved search technique over HC.

Motivated by these results, this subsection describes a novel approximate BN structure learning algorithm, which we call Model Averaging Hill-Climbing (MAHC), that combines two novel strategies, pruning and model averaging, with hill-climbing search (Constantinou et. al., 2022). This set of strategies produces an algorithm that searches a considerably smaller search-space of graphs and maximises the score over a set of graphs, rather than exploring individual graphs which might be generating a higher score due to data noise.

Pruning the search space of graphs represents a particularly important strategy in exact learning algorithms. This is because, in the absence of pruning, an exact algorithm would need to perform exhaustive search to guarantee the discovery of the optimal graph. Because exact learning is known to be computationally intractable, pruning becomes necessary. An important distinction between pruning strategies involves whether the pruning is *sound* or not, and sound pruning ensures the pruned search space of graphs contains the optimal graph. Research into sound pruning has been important in the development of exact search. Well-established exact learning solutions include integer programming approaches such as GOBNILP by Cussens (2011), and combinatorial optimisation approaches such as Branch-and-Bound by de Campos (2009). Both these approaches employ effective versions of sound pruning and allow exact learning to scale to tens of variables.

However, because pruning for approximate learning algorithms need not to be sound, it can be more aggressive. This is especially useful in the presence of data noise given that the highest scoring graph would be the one that best fits the noisy – not the true – data. On this basis, there is no incentive to preserve the highest scoring graph in the search-space of graphs. Moreover, even in the case of noise-free input data, Guo and Constantinou (2020) show that pruning CPSs by removing those with relatively low local scores leads to marginal reductions in structure learning accuracy in exchange for considerable increase in efficiency, thereby easing the application of structure learning to large datasets.

Model averaging, on the other hand, aims to reduce inconsistencies in the learnt output. It may involve returning the average output over multiple outputs produced by different ML algorithms, or the average output over multiple candidate outputs as determined by a single ML algorithm. In the context of structure learning, unlike model selection which involves returning the single best graph discovered, model averaging typically involves returning an output that represents a weighted average across a set of high-scoring graphs. One of the earliest papers that discuss the difference between model selection and model averaging in this context of structure learning is the work by Madigan et al. (1996) who average the output over a set of CPDAGs. Recent related works include those by Chen and Tian (2014) who implemented an algorithm to return the k-best equivalence classes of BN structure for model averaging, by Goudie and Mukherjee (2016) who describe a Gibbs sampler for learning DAGs that involves averaging across a set of DAGs that satisfy a set of conditions, and by Kuipers et al. (2022) who propose a hybrid learning algorithm that samples DAGs from the posterior distribution to reduce the complexity of MCMC and enable full Bayesian model averaging for large networks.

3.5.1 The MAHC algorithm

The MAHC algorithm can be viewed as a variant of the classic HC algorithm with two extensions. The first extension involves pre-processing some of the local objective scores and applying pruning to the search space of DAGs. The outcome of the pre-processing step will be a set of arcs pruned off, in addition to a set of local scores pre-processed that can be reused during the structure learning phase.

We denote the set of discrete variables by uppercase letter V, the CPS j of variable V_i by $CPS_{i,j}$ where i iterates over all variables and j iterates over the CPSs of V_i, and $S_{i,j}$ corresponds to the objective score of $CPS_{i,j}$. MAHC employs the following pruning rules by exploring CPS up to a node in-degree of 3:

Pruning rule 1 – for all empty and single-parent CPS: Assuming $CPS_{1,1}$ and $CPS_{1,2}$ have corresponding scores $S_{1,1}$ and $S_{1,2}$, if $CPS_{1,1} \subset CPS_{1,2}$ and $S_{1,1} \geq S_{1,2}$, then the parents resulting from set subtraction $CPS_{1,2} - CPS_{1,1}$ are pruned off. Note that any edges pruned off apply to CPSs of all sizes.

Pruning rule 2 – with constraints for parent-sets of size 2 and 3: Each CPS corresponds to a node i that is part of V, denoted as V_i, and each V_i has $|V| - 1$ possible parents ranked by lth highest score. Consider that the first and second highest valid[1] scoring parents of V_i are $CPS_{i,l=1}$ and $CPS_{i,l=2}$ respectively; e.g., $CPS_{i,l=1}$ has the highest score as a CPS of size one (single parents) of V_i. When **Pruning rule 1** is executed on CPSs of size two for node V_i, it is only applied to CPSs that contain $\{CPS_{i,l=1}, V_k\}$ and iterate over k, where $V_k \notin \{CPS_{i,l=1}, V_i\}$. Similarly, when executed on CPS of size three, it will be restricted to CPSs that contain $\{CPS_{i,l=1}, CPS_{i,l=2}, V_k\}$ iterating over k, where $V_k \notin \{CPS_{i,l=1}, CPS_{i,l=2}, V_i\}$. In other words, for CPS sizes greater than 1, pruning is

[1] It is possible for one of the highest scoring parents to be pruned off during pre-processing. This can happen when pre-processing CPSs of at least size 2. When this happens, the next available highest scoring parent takes the place, in the ladder of highest scores for a given node, of the parent that has been pruned off.

only applied to the CPSs that include the $p-1$ highest scoring valid parents, where p denotes the number of parents.

Once the pre-processing phase is completed and the set of edges that can be considered for structure learning is determined, the algorithm moves to the structure learning phase which involves the second extension where model averaging is applied over the hill-climbing search space. Unlike traditional model averaging which involves averaging over a set of graphs, the model averaging approach employed in MAHC involves averaging a set of model-selection scores (i.e., BIC scores), where each average score is assigned to a single graph explored. The formal description of this modification is provided by **Modification 1** and **Modification 2**, for search and score respectively.

Modification 1 (Search): Given a candidate DAG G, traditional hill-climbing involves visiting each neighbouring graph G_n of G at each hill-climbing iteration. In the extended version, we modify search such that each hill-climbing iteration involves not only visiting each neighbouring graph G_n of G, but also each neighbouring graph G_{nn} of G_n (i.e., G_{nn} is the neighbouring graph of the neighbouring graph of G).

Modification 2 (Score): Given a candidate DAG G, traditional hill-climbing moves to the neighbouring graph that maximises $S(G_n)$ given a set of scores S_n that consists of multiple $S(G_n)$. In other words, traditional hill-climbing searches for the maximum objective score $S(G_n)$across neighbouring scores. In the extended version, hill-climbing moves to the neighbouring graph that returns $\max\left(\overline{S(G_n, G_{nn})}\right)$ given a set of scores S_n that consists of multiple $\overline{S(G_n, G_{nn})}$. In other words, the extended version searches for the highest average objective score $\overline{S(G_n, G_{nn})}$, each of which corresponds to a neighbouring graph G_n and the scores of all its valid neighbouring graphs G_{nn}.

3.5.2 Evaluation and results

We evaluate MAHC by considering the same evaluation setup as described in subsections 3.1, 3.2 and 3.3. However, we focus on the case which contains multiple types of data noise (i.e., cMISL), under the assumption that multiple types of data noise represent a more realistic scenario, and data sample sizes of 0.1k, 1k, 10k, and 100k. Moreover, we compare MAHC against six structure learning algorithms, spanning all three classes of structure learning. These are HC, TABU, GOBNILP, PC-Stable, FCI, MMHC and SaiyanH.

Learning class	Relative to algorithm:	F1			BSF			SHD		
		Better	Same	Worse	Better	Same	Worse	Better	Same	Worse
Constraint-based	FCI	**83.3%**	0.0%	16.7%	**66.7%**	0.0%	33.3%	**62.5%**	4.2%	33.3%
	PC-Stable	**79.2%**	0.0%	20.8%	**54.2%**	0.0%	45.8%	**54.2%**	0.0%	45.8%
Score-based (exact)	GOBNILP	**45.8%**	20.8%	33.3%	29.2%	20.8%	**50.0%**	37.5%	16.7%	**45.8%**
Score-based (approximate)	HC	**54.2%**	37.5%	8.3%	**45.8%**	29.2%	25.0%	**58.3%**	25.0%	16.7%
	TABU	**50.0%**	25.0%	25.0%	33.3%	20.8%	**45.8%**	**41.7%**	20.8%	37.5%
Hybrid	MMHC	**41.7%**	20.8%	37.5%	**66.7%**	16.7%	16.7%	**66.7%**	16.7%	16.7%
	SaiyanH	**79.2%**	4.2%	16.7%	41.7%	4.2%	**54.2%**	**50.0%**	0.0%	**50.0%**

Table 3.6 The graphical accuracy of MAHC relative to the other seven algorithms. The percentages represent the number of times the score of MAHC was better, worse, or the same relative to each of the other algorithms, across all case studies, sample sizes, and noisy experiments. The best performance is shown in bold.

The results that follow are discussed in terms of “better score”, indicating higher learning accuracy. For F1 and BSF, a better score refers to a higher F1 or BSF value. In the case of the SHD, however, a better score refers to a lower SHD value. Table 3.6 presents the graphical accuracy of MAHC relative to the other algorithms in the presence of data noise, over all experiments. While MAHC is shown to perform similar to other algorithms in noise-free experiments (Constantinou et al., 2022), the results presented in Table 3.6 that focus on structure learning with noisy data show that MAHC consistently outperforms most of the other algorithms, suggesting that the model averaging process make MAHC considerably less sensitive to data noise compared to other algorithms.

Table 3.7 summarises the BIC results in terms of normalised average percentage score. For example, the average score of GOBNILP is 100% for clean data at 0.1k sample size, because GOBNILP produced the highest score in all of the experiments that involved clean data with a sample size of 0.1k. The results show that the score-based algorithms produce the highest model-selection scores, as expected (since they maximise BIC by design/implementation). The hybrid algorithms which include a phase that relies on constraint-based learning, as well as the constraint-based algorithms, generate considerably lower BIC scores since they are not designed with the sole purpose to maximise BIC. The BIC scores of MAHC are closer to those of HC, and considerably stronger than those produced by the hybrid and constraint-based learning algorithms. This outcome suggests that model averaging has a relatively small negative impact on the model-selection score, especially in the presence of data noise, relative to the gain in graphical accuracy over HC as illustrated in Table 3.6.

What could be classified as surprising, however, is the BIC score of the ground truth graphs. The results show that the lower the sample size, the lower the chance the true graph will be part of the higher scoring graphs. However, this is true only for the clean data cases. The data noise drops the relative performance of the ground truth under all sample sizes, and decreases the BIC score of the ground truth graph considerably. This supports our initial hypothesis that the incentive to search for the highest scoring graph diminishes considerably in the presence of data noise.

Sample size	Score-based			Exact score-based	Constraint-based		Hybrid		True graph
	MAHC	TABU	HC	GOBNILP	PC-Stable	FCI	MMHC	SaiyanH	
100 (Clean)	94.63%	99.16%	99.13%	**100%**	76.21%	77.65%	69.67%	88.88%	11.44%
1k (Clean)	88.19%	97.08%	93.77%	**100%**	35.15%	43.91%	40.68%	91.19%	42.70%
10k (Clean)	92.50%	**98.66%**	97.42%	90.21%	20.77%	28.45%	18.14%	90.22%	80.96%
100k (Clean)	83.94%	86.66%	85.84%	80.00%	57.27%	56.94%	9.99%	**93.56%**	98.15%
100 (Noisy)	99.96%	99.97%	99.97%	**99.98%**	95.16%	94.76%	99.73%	86.53%	0.00%
1k (Noisy)	99.54%	99.92%	99.86%	**100.00%**	69.09%	66.42%	99.32%	91.06%	30.81%
10k (Noisy)	97.83%	99.68%	98.89%	**100.00%**	19.45%	16.54%	95.70%	97.45%	72.34%
100k (Noisy)	94.37%	98.69%	94.98%	**100.00%**	59.25%	41.89%	47.22%	84.24%	34.51%

Table 3.7 The percentages represent the average normalised BIC scores, where higher percentages correspond to a better score. An average of 100% indicates that the algorithm obtained the highest average BIC score across all experiments.

3.6 Conclusions

This study focused on evaluating the performance of BN structure learning algorithms, with the objective to assess their ability to reconstruct the true causal graphs under various hypotheses of data noise. The investigation focused on 15 different algorithms from different classes of learning, and 16 different data noise scenarios, over six case studies and five sample sizes.

The results suggest that data noise can have a considerable impact on the accuracy of the learnt graph. Specifically, incorporating all four types of noise in a single dataset decreases structure learning accuracy in the range of 30% to 37% (i.e., accuracy increases by 43% to 59% without data noise). These results have major implications since they suggest that BN structure learning accuracy presented in the literature, on the basis of traditional synthetic data, overestimates real-world performance to higher degree than maybe was previously assumed. Still, traditional noise-free synthetic experiments remain important in evaluating BN structure learning algorithms under various hypothetical assumptions.

With regards to the novel model averaging MAHC structure learning algorithm, the results suggest that the performance of MAHC is competitive when the input data are clean, and often superior when the input data are noisy. The results suggest that model averaging strategies could be better suited for learning from real data, under the assumption that real observations never satisfy the ideal conditions assumed in clean synthetic experiments and that they often incorporate different kinds of data noise, many of which might be similar to those assumed in this work. Additionally, the results show that the ground truth graph will not only not have the highest objective score, but will also often deviate considerably from the highest scoring graph, either due to data noise or limited data, both of which distort model fitting. This decreases the incentive to search for the highest scoring graph, and at the same time increases the importance of approximate learning.

Practitioners who work with real data should priorotise structure learning solutions with weaker assumptions that are less sensitive to data noise. The results presented in this chapter suggest that, in general, score-based solutions tend to be more resilient to noise compared to constraint-base methods, and this is perhaps explained by constraint-based learning being sensitive to early errors made during the structure learning process, which in turn affect subsequent results of conditional independence tests. This is an observations that would benefit from further investigation. The results also support model averaging strategies which are found to successfully reduce sensitivity to data noise in the context of structure learning. Finally, because it is impossible for algorithms that assume noise-free data to successfully generalise to noisy experiments, the incorporation of knowledge-based constraints remains a desirable feature for the application of these algorithms to real-world problems, warranting further exploration in future investigations.

Chapter 4

Structure learning with causal effects in the presence of continuous data and latent variables

Latent variables may lead to spurious relationships that can be misinterpreted as causal relationships. As discussed in subsection 2.2.1.2, this challenge is known as learning under causal insufficiency. Structure learning algorithms that assume causal insufficiency tend to reconstruct an ancestral graph where bidirected edges represent confounding and directed edges represent direct or ancestral relationships.

This chapter presents a hybrid structure learning algorithm called Conservative rule and Causal effect Hill-climbing for MAG (CCHM), which can be used to recover ancestral graphs (covered in subsection 2.2.1.2) from data with latent variables. CCHM combines the constraint-based part of cFCI with hill-climbing score-based learning. The score-based process incorporates Pearl's do-calculus to measure causal effects, which are used to orientate edges that would otherwise remain undirected.

We focus on Gaussian Bayesian Networks (GBNs), where the data follows a multivariate Gaussian distribution. In general, GBNs (Geiger and Heckerman, 1994) consist of a random variable X_i where:

$$P\big(X_i \big| \mathrm{parent}(X_i)\big) \sim \mathcal{N}(\theta_{X_i} | \mathrm{parent}(X_i))$$

and X_i can be written in the form of a linear regression model:

$$X_i = \mu x_i + \beta\, \mathrm{parent}(X_i) + \epsilon$$

where μ is the mean of the random variable X_i, β is the coefficient for the directed edge j to i $\{\beta_{ij}\}$ and ϵ is a positive random error vector which follows a Gaussian distribution $\epsilon \sim \mathcal{N}(0, \sigma_{x_i}^2)$ with covariance $\sigma_{x_i}^2$.

This chapter describes the hybrid CCHM algorithm that is designed to learn GBNs from causally insufficient data. Relevant literature review, including descriptions of the relevant algorithms considered in this chapter, can be found in subsections 2.2.1.3, 2.2.2.2 and 2.2.3. The chapter is organised as follows: subsection 4.1 describes the CCHM algorithm, subsection 4.2 describes random networks and real networks as the case studies and evaluation, subsection 4.3 illustrate results, and subsection 4.4 provides concluding remarks and future research directions.

4.1 Conservative rule and Causal effect Hill-climbing for MAG (CCHM)

The process of CCHM can be divided into two phases. The first phase adopts the CI tests of cFCI (covered in subsection 2.2.1.3) to construct the skeleton of the graph and to further classify definite colliders as whitelist and definite non-colliders as blacklist. The second phase involves score-based learning that uses the BIC score as the objective function, adjusted for MAGs, where edge orientation is augmented with causal effect measures. These steps are described as follows:

a) Definite colliders (whitelist) and definite non-colliders (blacklist)

CI tests are used to determine the edges between variables and to produce the skeleton graph. A p-value associates with each statistical test result, which is used to sort conditional independencies in ascending order. An α hyperparameter is then used as the cut-off threshold in establishing independence. For each conditional independency $A \perp B \mid \mathbf{Z}$, $\mathbf{Z}$ is recorded as Sepset of nodes A and B. The orientation of edges is determined by a method inherited from cFCI, where extra CI tests over all unshielded triples determine the classification for each of those triples as either a definite collider or a definite non-collider:

- Given unshielded triple $A - C - B$, perform CI tests on A and B over all neighbours of A and B.
- If C is **NOT** in all Sepsets of A and B, add $A - C - B$ to the whitelist as a definite collider.
- If C is in **ALL** Sepsets of A and B, add $A - C - B$ to the blacklist as a definite non-collider.

b) BIC for MAGs

The score-based learning part of CCHM involves hill-climbing greedy search that minimises the BIC score, which balances the goodness-of-fit scores against a penalty term for model dimensionality based on Occam's razor principle. CCHM adopts the BIC function used in the M^{3}HC (Tsirlis et al., 2018) and GSMAG algorithms (Triantafillou and Tsamardinos, 2016), which is adjusted for MAGs. Formally, given a dataset over variables V with a distribution $\mathcal{N}(0, \Sigma)$ where Σ is a covariance matrix calculated from the dataset, a unique solution Y is found where $\hat{\Sigma} = (I - \mathcal{B})^{-1}\Omega(I - \mathcal{B})^{-t}$. MAG $\mathcal{G}$ is constructed from linear equations $Y = \mathcal{B} \cdot Y + \epsilon$, where $Y = \{Y_i | i \in V\}$, $\mathcal{B}$ is a $V \times V$ coefficient matrix for the directed edge j to i $\{\beta_{ij}\}$, I is an identity matrix, ϵ is a positive random error vector for the bidirected edge j to i $\{\omega_{ij}\}$, and the error covariance matrix $\Omega = \mathrm{Cov}(\epsilon) = \{\omega_{ii}\}$. The BIC score is then calculated as follows (Richardson, Spirtes, 2000):

$$\mathrm{BIC}(\hat{\Sigma}|\mathcal{G}) = -2\ln\left(\mathrm{LL}_{\mathcal{G}}(\hat{\Sigma}|\mathcal{G})\right) + \ln(n)(2|V| + |E|)$$

where $\mathrm{LL}_{\mathcal{G}}$ is likelihood function, $|V|$ and $|E|$ are the size of nodes and edges that are part of the complexity penalty term, and n is the sample size. Similar to the factorisation property of DAGs, the score $\mathrm{LL}_{\mathcal{G}}(\hat{\Sigma}|\mathcal{G})$ can be decomposed into c-components (S_k) of $\mathcal{G}$ which refer to the connected components that are partitioned by removing all directed edges (Nowzohour et al., 2015):

$$LL_{\mathcal{G}}(\hat{\Sigma}|\mathcal{G}) = -\frac{N}{2}\sum_{k} S_k$$

$$\text{where } S_k = |C_k| \cdot \ln(2\pi) + \ln\left(\frac{|\hat{\Sigma}_{\mathcal{G}_k}|}{\prod_{j\in Pa_{\mathcal{G}_k}} \sigma_{kj}^2}\right) + \frac{n-1}{n} \cdot \mathrm{tr}[\hat{\Sigma}_{\mathcal{G}_k}^{-1} S_{\mathcal{G}_k} - |Pa_{\mathcal{G}}(C_k)\backslash\{C_k\}|]$$

and where C_k denotes the set of nodes for each c-component k, $\mathcal{G}_k$ is the marginalisation from C_k, with all their parent nodes defined as $Pa_{\mathcal{G}}(C_k)$ in C_k, and σ_{kj}^2 represents the diagonal $\hat{\Sigma}_{\mathcal{G}_k}$ of the parent node k. The likelihood $\hat{\Sigma}$ is determined by the RICF algorithm (Drton et al., 2006).

c) Direct causal effect criteria

Because the BIC for MAGs is a Markov equivalent score, it is incapable of orientating all edges from statistical observations. Optimising for BIC under causal insufficiency returns a PAG, or one of the MAGs that are part of the equivalence class of the optimal PAG. In this work, we are interested in orientating all edges that would enable us to generate a MAG, rather than a PAG, output. We achieve this using Pearl's do-calculus (Pearl, 2000) to measure the direct causal effect on edges that the BIC score fails to orientate. The direct causal effect is estimated by intervention that renders the intervening variable independent of its parents.

Theorem: Single-door criterion for direct effect

Single-Door Criterion for direct effect (Pearl, 2000): Let G be any DAG in which β is the path coefficient associated with X→Y, the path coefficient β is identifiable and equal to the regression coefficient if there exists a set of variables that (i) contains no descendant of Y and (ii) is a set d-separated of X and Y in subgraph after removing X→Y from G.

The interpretation of the path coefficient β in the regression of the single-door criterion theorem can be expressed as the direct causal effect determined by the rate of change of E[Y] given intervention X (Maathuis et al., 2009) as follows:

$$\beta = \frac{\partial}{\partial x} E[Y|\, do(x)] = E[Y|do(X = x + 1)] - E[Y|do(X = x)] \text{ for any value of x}$$

This assumes that all causal effect parameters are identifiable by RICF (Drton et al., 2006), and that the path coefficient or the direct causal effect is the regression coefficient estimated from the likelihood function. Let A→B be the edge in the true graph, the Structural Equation Model (SEM) $B = \beta_A A + \epsilon_B$, if we assume that we have $A \sim \mathcal{N}(\mu_A, \sigma^2{}_A)$, $\epsilon_B \sim \mathcal{N}(0, \sigma_{\epsilon_B}^2)$, and ϵ_B and A are independent. Thus, $E[B] = \beta_A E[A], \sigma^2{}_B = \beta_A{}^2 \sigma^2{}_A + \sigma_{\epsilon_B}^2$. For every pair A and B in the learned graph, two causal graphs where A→B and A←B need to be constructed to measure the direct causal effects. Specifically,

- For graphs A→B, do the intervention on A; i.e., $do(a)$ (Pearl, 2000, page 161)

$$\beta_A = \frac{E[BA]}{E[A^2]} \qquad (4.1)$$

- For graphs B→A, do the intervention on B; i.e., $do(b)$.

$$\beta_B = \frac{E[AB]}{E[B^2]} \qquad (4.2)$$

From Equations (4.1), (4.2) and the variance of a random variable X $(\sigma^2{}_X) = E[X^2] - E[X]^2$:

$$\frac{\beta_A}{\beta_B} = \frac{E[B^2]}{E[A^2]} = \frac{E[B]^2 + \sigma^2{}_B}{E[A]^2 + \sigma^2{}_A}$$

Substitute $E[B] = \beta_A E[A],\ \sigma^2{}_B = \beta_A{}^2 \sigma^2{}_A + \sigma^2_{\epsilon_B}$ from the graph,

$$= \frac{\beta_A^2 E[A]^2 + \beta_A{}^2\sigma^2{}_A + \sigma^2_{\epsilon_B}}{E[A]^2 + \sigma^2{}_A} = \beta_A^2 + \frac{\sigma^2_{\epsilon_B}}{E[A]^2 + \sigma^2{}_A} \qquad (4.3)$$

If $E[A] = \mu_A = 0, \sigma^2{}_A = 1$ and $\sigma^2{}_{\epsilon_B} = 1$ in the normal distribution in (4.3)

$$\frac{\beta_A}{\beta_B} = \beta_A{}^2 + 1 \text{ ; we have the probability } (|\beta_A| > |\beta_B|) = 1$$

Note that this aligns with the study by Peters and B¨uhlmann (2013) where a causal graph can be identifiable from observational data in linear Gaussian models with equal error variances. However, in their experiments the authors assume causally sufficient with the results restricted to DAG discovery, whereas here we assume causal insufficiency aiming for MAG discovery. Algorithm 1 describes the steps of CCHM in detail.

Algorithm 1: CCHM (Conservative rule and Causal effect Hill-climbing for MAG)

```
Input: significance threshold α, maximum Sepset size k, CI test
Output: MAG
// Search for a skeleton (Step 1 and 2 are the first and second steps of the cFCI Algorithm)
Step 1    Set up a complete undirected graph and initialise Sepset Z with size =0
          Repeat
              remove edges between each pair of nodes A and B that become independent conditional
              on Sepset Z, as determined by α
          Until all Sepset Z size = k have been tested
Step 2    Given unshielded triple A − C − B, perform CI tests on A and B given all neighbours of A and
          B as determined by α
              a) If C is NOT in all Sepsets of A and B, add A − C − B to the whitelist as a definite
                 collider
              b) If C is in ALL Sepsets of A and B, add A − C − B to the blacklist as a definite non-
                 collider
Step 3    Orientate as many edges as possible in the skeleton graph given the whitelist, and retrieve the
          BIC score for MAGs of the resulting graph
// Score-based learning with do-calculus
Step 4    Repeat
            For each pair(A,B), in ascending order by p-value
              Calculate the BIC scores for MAGs for each edge A→B, A←B and A↔B.
              If i) BIC decreases, ii) the result graph remains acyclic, and iii) the result triple is not in
                 blacklist
                If edges A→B, A←B and A↔B produce unequal BIC scores for MAGs
                  Add the edge A→B, A←B or A↔B that minimises the BIC score for MAGs
                Else
                  Calculate the direct causal effect β for edges A→B and A←B
                        $\beta_A = E(B|do(A = a + 1)) - E(B|do(A = a))$
                        $\beta_B = E(A|do(B = b + 1)) - E(A|do(B = b))$
                  Orientate A→B or A←B that maximises the direct causal effect from
                  Equation (4.3)
          Until no undirected edges remain
```

4.2 Evaluation

The graphs produced by the CCHM algorithm are compared to the outputs of the M^{3}HC, GSPo, GFCI, RFCI, FCI, and cFCI algorithms, when applied to the same data. GSPo is an order-based search algorithm that performs greedy search over the space of independence maps (IMAPs) to determine the minimal IMAP (Bernstein et al., 2019). This is achieved by defining a partial ordered set (poset) that is linked to the IMAP, expressed as a discrete optimisation problem. However, GSPo uses a random starting point for a poset, and this makes the algorithm non-deterministic since each run is likely to produce a different MAG. Experimental results show that GSPo may converge to a different solution each time it is executed, and this instability makes such algorithms more difficult to evaluate and less desirable in practice. The GSPo algorithm was tested using the *causaldag* Python package by Squires (2018), the M^{3}HC algorithm was tested using the MATLAB implementation by Triantafillou (2019), and the GFCI and RFCI algorithms were tested using the Tetrad-based *rcausal* package in R (Wongchokprasitti, 2019). The computational time of CCHM is compared to the M^{3}HC, FCI and cFCI, which are based on the same MATLAB package. The CCHM implementation is available online at https://github.com/kiattikunc/CCHM.

All experiments are based on synthetic data. However, we divide them into experiments based on data generated from BNs which had their structure and dependencies randomised, and data generated from real-world BNs. Randomised BNs were generated using Triantafillou's (2019) MATLAB package. We created a total of 600 random Gaussian DAGs that varied in variable size, max in-degree, and sample size. Specifically, 50 DAGs were generated for each combination of variables V and max in-degree settings $\mathcal{D}$, where V = {10, 20, 50, 70, 100, 200} and $\mathcal{D}$ = {3, 5}. Each of those 600 graphs was then used to generate two datasets of sample sizes 1k and 10k, for a total of 1,200 datasets. Data were generated assuming linear Gaussian parameters $\mu = 0$ and $\sigma^2 = 1$ and uniformly random coefficients $\beta \pm[0.1,0.9]$ for each parent set to avoid very weak or very strong edges. Approximately 10% of the variables in the data are made latent in each of the 600 datasets.

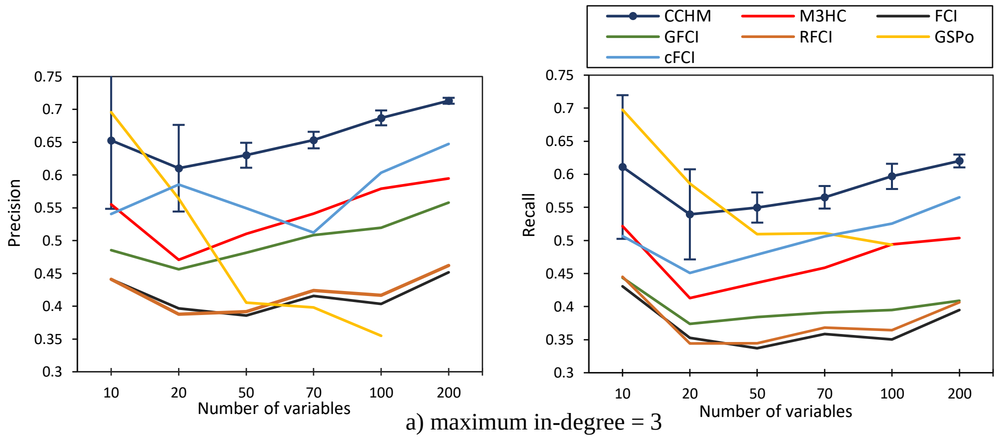
In addition to the randomised networks, we made use of four real-world Gaussian BNs taken from the *bnlearn* repository (Scutari, 2019). These are the a) *MAGIC-NIAB* (44 nodes) which captures genetic effects and phenotypic interactions for Multiparent Advanced Generation Inter-Cross (MAGIC) winter wheat population, b) *MAGIC-IRRI* (64 nodes) which captures genetic effects and phenotypic interactions for MAGIC indica rice population, c) *ECOLI70* (46 nodes) which captures the protein-coding genes of *E. coli*, and d) *ARTH150* (107 nodes) which captures the gene expressions and proteomics data of *Arabidopsis Thaliana*. Each of these four BNs was used to generate data, with the sample size set to 10k. For each of the four datasets, we introduced four different rates of latent variable: 0%, 10%, 20% and 50%. This made the total number of real-world datasets 16; four datasets per BN.

The following hyperparameter settings are used for all algorithms: a) $\alpha = 0.01$ for the Fisher's z CI test for datasets sampled from the randomised BNs, b) $\alpha = 0.05, 0.01, 0.001$, which are the same settings as those used by Tsirlis et al. (2018), for datasets generated by the real-world BNs, and c) the max Sepset size of the conditioning set is set to '4' so that runtime is maintained at reasonable levels. The maximum length of discriminating paths is also set to '4' for the four FCI-based algorithms (this is the same as the max Sepset size). For GSPo, the depth of depth-first search is set to '4' and the randomised points of posets to '5' (these are the default settings). Because GSPo is a non-deterministic algorithm that generates a different output each time it is executed, we report the average scores obtained over five runs. Lastly, all algorithms were restricted to a four-hour runtime limit. Further, because the algorithms will output either a PAG or a MAG, we convert all MAG outputs into the corresponding PAGs. The accuracy of the learnt graphs is then assessed with respect to the true PAG. The results are evaluated using the traditional measures of Precision and Recall, SHD, and BSF, described in subsection 2.2.4.

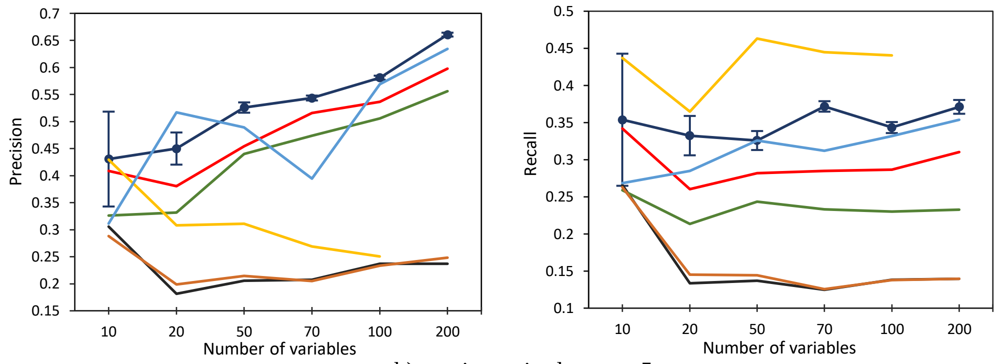
4.3 Empirical results

4.3.1 Results based on random Gaussian Bayesian Networks

Figure 4.1 presents the Precision and Recall scores achieved by each of the algorithms on the datasets generated by the randomised BNs. The scores are averaged across the different settings of variable size and max in-degree. Note that because there was no noteworthy difference between the overall results obtained from the two different data sample sizes, we only report the results based on sample size 10k. Therefore, the results and conclusions based on the datasets with sample size 10k also hold for the datasets with sample size 1k.



a) maximum in-degree = 3



b) maximum in-degree = 5

Figure 4.1 Average Precision and Recall scores of the algorithms (variances for CCHM) for each combination of variable size and max in-degree settings (50 graphs per combination). The results are based on synthetic data with sample size 10k and assume that 10% of the variables are latent.

Overall, the results show that CCHM outperforms all other algorithms in terms of both Precision and Recall, and across all settings excluding Recall under max in-degree 5 where GSPo ranks highest (Figure 4.1b). While GSPo appears to perform best when the number of variables is lowest, its performance decreases sharply with the number of variables, and fails to produce a result within the four-hour time limit when the number of variables is highest.

The results show no noticeable difference between FCI and its variant RFCI, whereas the cFCI and GFCI show strong improvements over FCI, with cFCI outperforming all the other FCI-based algorithms. Moreover, the performance of cFCI is on par with that of M^{3}HC. Note that while CCHM employs the BIC objective function of M^{3}HC, CCHM outperforms M^{3}HC in both sparse (Figure 4.1a) and dense (Figure 4.1b) graphs. This result provides empirical evidence that the conservative rules used in the constraint-based phase of CCHM and the do-calculus used in the score-based phase of CCHM have indeed improved structure learning performance.

Figure 4.2 compares the average runtime of CCHM to the runtimes of the other algorithms. The runtime comparison is restricted to algorithms that are based on the same MATLAB implementation on which CCHM is based. The results show that CCHM is marginally faster than cFCI and slower than the other algorithms, with the worst case scenario observed when the number of variables is highest, where CCHM is approximately two times slower than FCI.

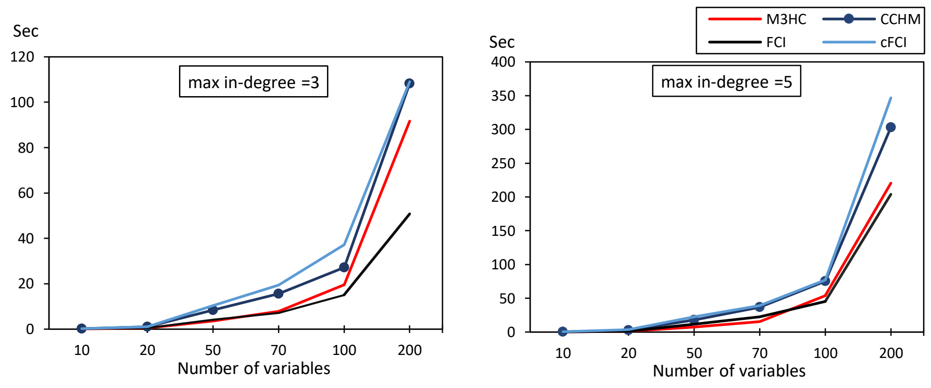
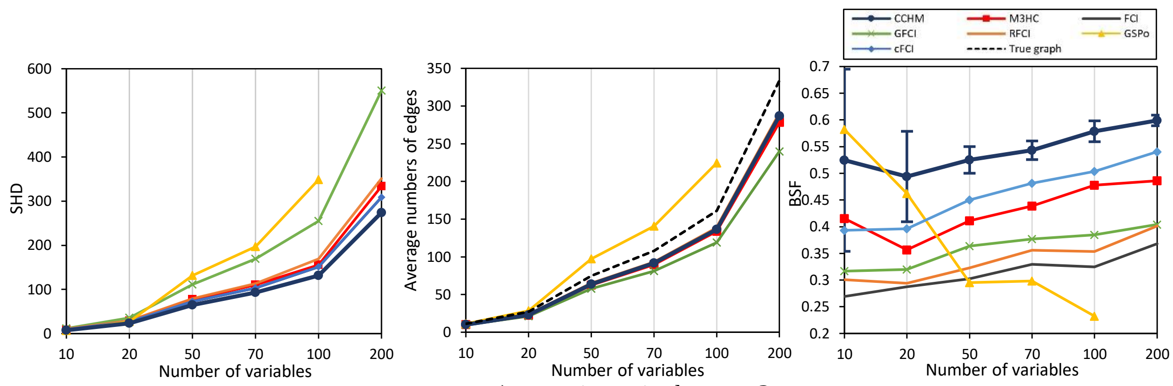
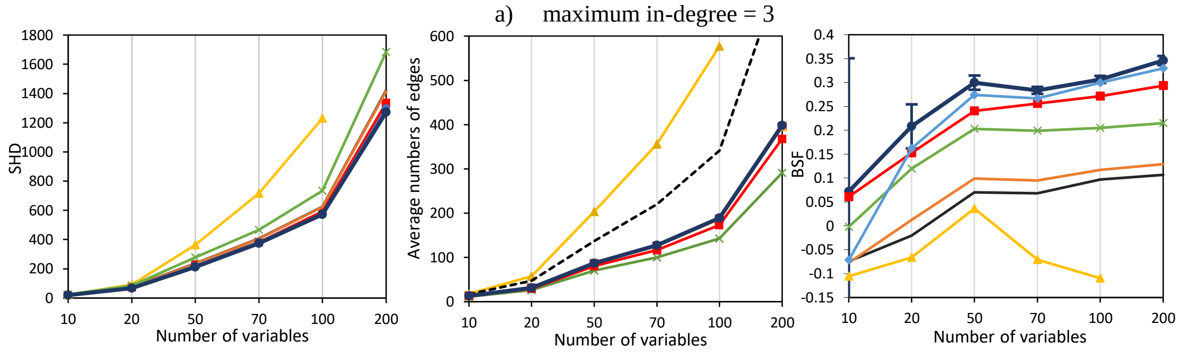


Figure 4.2 Average computation time of the algorithms for each combination of variable size and max in-degree settings (50 graphs per combination). The results are based on synthetic data with sample size 10k and assume that 10% of the variables are latent.



a) maximum in-degree = 3



b) maximum in-degree = 5

Figure 4.3 Average number of edges, SHD and BSF scores of the algorithms (variances of BSF for CCHM) for each combination of variable size and max in-degree settings (50 graphs per combination). The results are based on synthetic data with sample size 10k and assume that 10% of the variables are latent.

Figure 4.3 presents the SHD and BSF scores, along with the corresponding numbers of edges generated by each algorithm. Both the SHD and BSF metrics rank CCHM highest when the number of variables is more than 10, and these results are consistent with the Precision and Recall results previously depicted in Figure 4.1 where GSPo performs best when the number of variables is lowest in sparse graphs. The number of edges produced by CCHM is in line with the number of edges produced by the other algorithms, and this observation provides confidence that CCHM achieves the highest scores due to accuracy rather than due to the number of edges, which may sometimes bias the result of a metric (Constantinou et. al., 2021). One inconsistency between the SHD and other metrics involves the GFCI algorithm, where SHD ranks lower than all the other FCI-based algorithms, something which contradicts the results of Precision, Recall, and BSF. Interestingly, while GSPo produces the highest BSF scores for graphs that incorporate just 10 variables, its performance diminishes drastically with the number of variables and quickly becomes the worst performer (refer to the BFS scores in Figure 4.3a); an observation that is largely consistent with the results in Figure 4.1.

4.3.2 Results based on real-world Gaussian Bayesian Networks

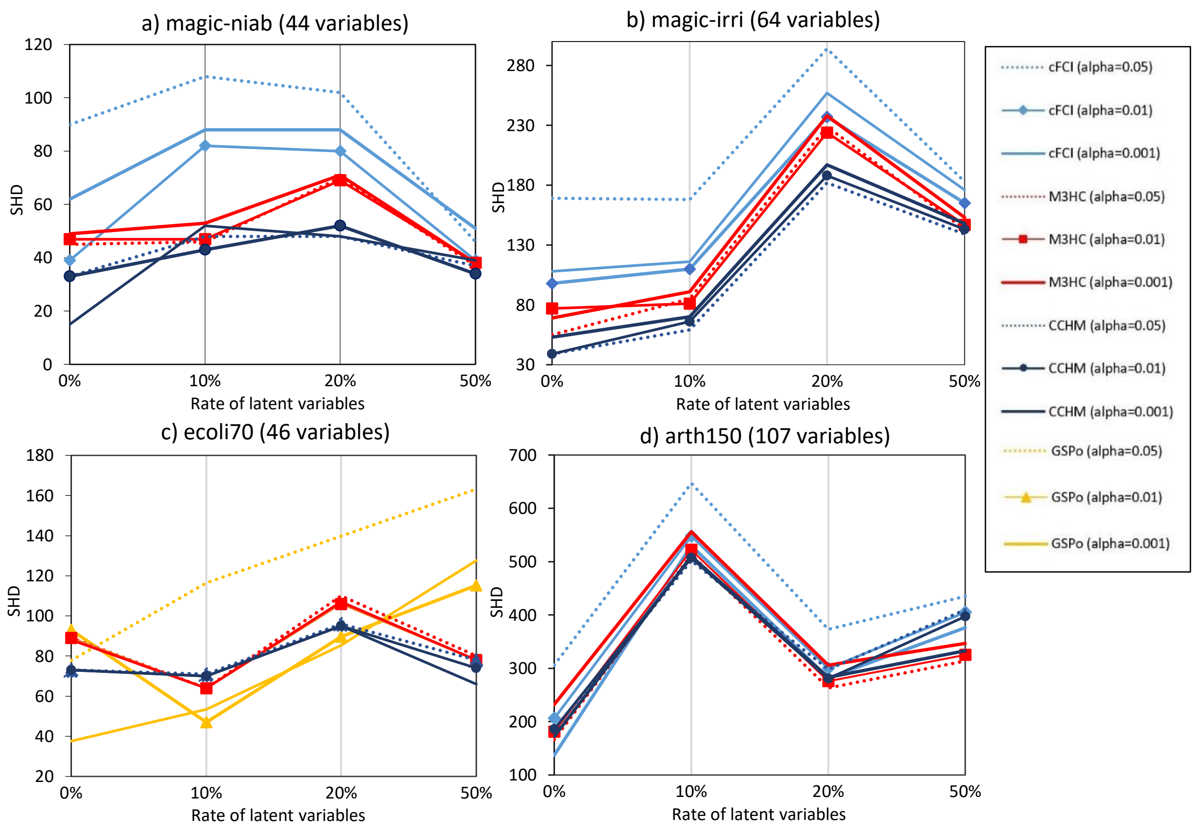


Figure 4.4 The SHD scores of the top three algorithms in each of the four Gaussian BNs, over three different input settings for hyperparameter α. The results are based on synthetic data with sample size 10k.

The reduced number of experiments that associate with the real-world GBNs (i.e., 16 instead of 600 randomised experiments) enabled us to also test the sensitivity of the algorithms on the α hyperparameter, which reflects the significance cut-off point in establishing independence. Figure 4.4 presents the SHD scores for each of the four real-world GBNs, and over different rates of latent variables. The results are restricted to the top three algorithms for each case study, and this is because we report three different results for each of the top three algorithms based on the three different hyperparameter inputs α specified in Figure 4.4.

Only four algorithms (CCHM, M^{3}HC, cFCI and GSPo) achieved a top-three performance in any of the four networks, and this suggests that the relative performance between algorithms is rather consistent across the different case studies. While there is no clear relationship between the rate of latent variables and SHD score, the results do suggest that the accuracy of the algorithms decreases with the rate of latent variables in the data. This is because while we would expect the SHD score to decrease with less variables in the data, since less variables lead to potentially fewer differences between the learned and the true graphs (refer to Figure 4.3), the results in Figure 4.4 reveal a weak increasing trend in SHD score with the rate of latent variables in the data.

Overall, the CCHM algorithm was part of the top three algorithms in all the four case studies. Specifically, CCHM generated the lowest SHD error in networks (a) and (b). The results in network (c) were less consistent, with GSPo ranked 1st at latent variable rates of 10% and 20%, and CCHM ranked 1st at latent variable rates of 0% and 50%. In contrast, the results based on network (d) show no noteworthy differences in the performance between the three top algorithms. Overall, the results suggest that cFCI and GSPo are much more sensitive to the α hyperparameter compared to the CCHM and M^3HC algorithms, and that CCHM generally performs best when $\alpha = 0.01$.

4.4 Conclusions

This work describes a novel structure learning algorithm, called CCHM, which builds on recent developments in BN structure learning under causal insufficiency. CCHM combines constraint-based and score-based learning with causal effects to learn GBNs. The constraint-based part of CCHM adopts features from the state-of-the-art cFCI algorithm, whereas the score-based part is based on traditional hill-climbing greedy search that minimises the BIC score for MAGs. CCHM applies Pearl's do-calculus as a method to orientate the edges that both constraint-based and score-based learning fail to do so from observational data. The results show that CCHM outperforms the state-of-the-art algorithms in the majority of the experiments, which include both randomised and real-world GBNs.

A limitation of this work is that the algorithm assumes linear GBNs and that the data are continuous. A possible direction for future work would be to extend this approach to discrete BNs, where causal insufficiency remains an important open problem (Jabbari et al., 2017). Other directions include investigating different strategies in the way the do-calculus effect is applied to the process of structure learning; e.g., it can be applied directly to the calculation of the BIC score during score-based learning, or computed as the total causal effect of the graph using do-calculus rules or via back-door adjustment with graph surgery. Lastly, causal insufficiency represents just one type of data noise that exist in real-world datasets, and future work could also investigate the effects of causal insufficiency when combined with other types of noise in the data.

Chapter 5

Hybrid structure learning from multiple discrete datasets by scoring multiple interventions

In BNs, the direction of edges is crucial for causal reasoning and decision making via intervention. However, as discussed in subsection 2.2.1.1 regarding Markov equivalence class considerations, many structure learning algorithms cannot orientate all edges purely from observational data. That is, the causal chain and common cause relationships cannot be distinguished from observational data, irrespective of sample size. Figure 5.1a presents an example where three nodes A, B and C, are $A \perp B \mid C$, and there are three DAGs that support this CI statement.

Interventional data, however, may help us orientate some of those undirected edges. As illustrated in Figure 5.1b, a perfect intervention on C would force its state to c (C = c) independent of its parents. Simulating hypothetical interventions by rendering the intervention independent of its causes enables us to measure the effect of intervention. Importantly, interventional analysis may help us distinguish over different DAGs that fall within the same Markov equivalence class.

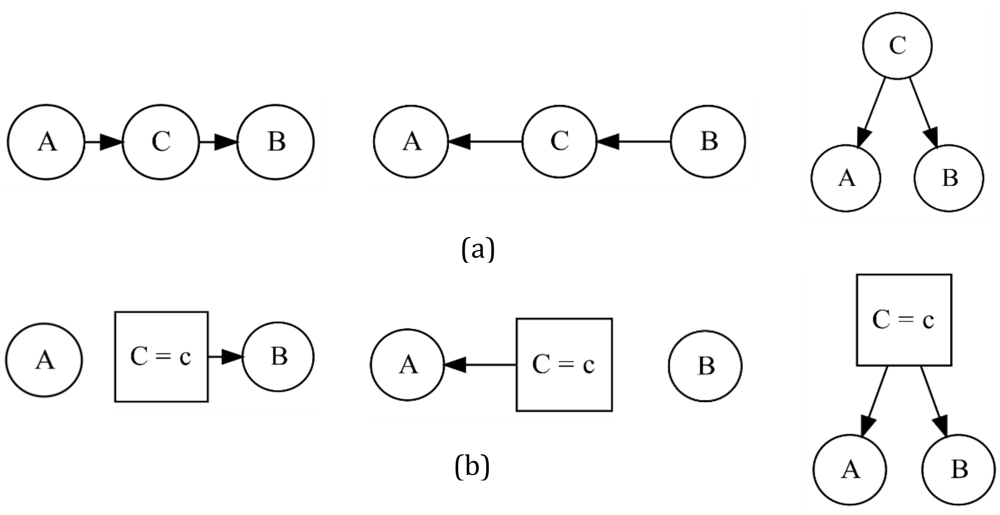


Figure 5.1 (a) Three Markov equivalent DAGs that entail the CI statement $A \perp B \mid C$, and (b) the corresponding modified DAGs when assuming a perfect intervention on C, where the square box represents the target node for intervention.

This chapter describes the hybrid mFGS-BS (majority rule and Fast Greedy equivalence Search with Bayesian Scoring) algorithm for structure learning from discrete data that involves an observational data set and one or more interventional data sets. The aim of the proposed algorithm is to orientate as many edges as possible from both observational and interventional data. The algorithm assumes causal insufficiency in the presence of latent variables and produces a PAG output. The proposed algorithm relies on a hybrid approach and a novel Bayesian scoring paradigm that calculates the posterior probability of each directed edge being added to the learnt graph.

This chapter is organised as follows: subsection 5.1 discusses related works, subsection 5.2 describes the mFGS-BS algorithm, subsection 5.3 describes evaluation setup along with the case studies, subsection 5.4 presents the results, and we provide our concluding remarks and future research directions in subsection 5.5.

5.1 Related works

Structure learning algorithms that learn from both observational and interventional data tend to do so from pooled data, which is a method that pools all datasets together with intervened variables specified. These algorithms aim to generate a graph that is consistent, as much as possible, with all input data. Examples include IGSP (Wang et al., 2017) and GIES (Hauser and B¨uhlmann, 2012) that return a DAG from pooled causally sufficient data.

Other methods involve determining the results of CI tests from each dataset separately and constructing a single graph using conflict resolution strategies. For causally insufficient data, the Causal discovery from Overlapping INtErventions (COmbINE) algorithm by Triantafillou and Tsamardinos (2015) implements the cFCI approach to learn the common characteristics and the results of CI tests from different datasets, which it then converts into Boolean Satisfiability (SAT) instances in a MINISAT application to resolve any conflicts. Other algorithms that operate on such results of CI tests include HEJ (Hyttinen et al., 2014) which uses Clingo (Gebser et al., 2011) – an Answer Set Programming (ASP) rule-based declarative programming language that solves various representations of NP-hard optimisation tasks (Gelfond and Lifschitz, 1988; Niemela, 1999) – for conflict resolution. It produces cyclic directed mixed graphs encoding results of CI tests from conditioning and marginalisation operations, and the graphs may contain directed, bidirected or undirected edges. The ACI algorithm (Magliacane et al., 2016) also relies on Clingo and can be viewed as a computationally less expensive variant of HEJ that operates in the search space of ancestral graphs, but which does not support bidirected edges for latent confounder representation. Lastly, JCI (Mooij et al., 2020) is a constraint-based algorithm that uses auxiliary context variables and system variables, which the authors define as variables of interest (presumably observed variables) and intervention targets respectively. JCI learns from a pooled dataset including knowledge about the relationship between context variables and generates a directed mixed graph, but which does not fall under the ancestral graph family. Table 5.1 summarises the main features of these relevant algorithms.

Algorithm	Class	Discrete /Continuous data	Output	Intervention type	Dataset
COmbINE	Constraint-based	Both	PAG	Perfect	Separate
HEJ	Constraint-based	Both	Cyclic Directed Mixed Graph	Perfect	Separate
JCI	Constraint-based	Both	Acyclic Directed Mixed Graph	Perfect/ Imperfect/ Uncertain	Pooled
ACI	Constraint-based	Both	Ancestral graph	Perfect/ Imperfect	Separate

Table 5.1 Overview of the relevant structure discovery algorithms that assume causal insufficiency, and learn graphs from multiple interventions.

Previous works that assumed prior probabilities for the existence of directed edges, as opposed to a binary outcome, include those by Castelo and Siebes (2000) who introduced the idea of assigning subjective prior probabilities (specified by experts) to directed edges, and by Scutari (2016) who assumed the marginal uniform prior probabilities of directed edges $A \rightarrow B$ and $A \leftarrow B$ to be ¼, while the prior probability of the independency between A and B to be ½ in a variant of the BD score called the Bayesian Dirichlet sparse score (BDs).

Hyttinen et al. (2014) proposed a Bayesian scoring method that applies prior probabilistic weights to the results obtained from CI tests. These prior probabilities are subjective and obtained from knowledge. In this work, we modify this method so that the prior probabilities are objectively calculated from data, and are assigned to directed edges rather than to the results obtained from CI tests. These details are discussed in subsections 5.2.1 and 5.2.2. With reference to the method by Hyttinen et al. (2014), the posterior probability of CI ($P(r|D_{OBS})$), given observational data, is:

$$P(r|D_{OBS}) = \frac{\text{prior} \times P(D_{OBS}|r)}{\text{prior} \times P(D_{OBS}|r) + (1 - \text{prior}) \times P(D_{OBS}|\bar{r})} \quad (5.1)$$

where r is an arbitrary CI that A and B are independent given $\mathbf{Z}$ ($A \perp B \mid \mathbf{Z}$), $\bar{r}$ is an arbitrary conditional dependence that A and B are dependent given $\mathbf{Z}$ ($A \not\perp B \mid \mathbf{Z}$), $\mathbf{Z}$ is the set of variables that is the Sepset of variables A and B, prior is an informative or uninformative probability from knowledge that $A \perp B \mid \mathbf{Z}$ is true, $P(D_{OBS}|r)$ is the network score of $A \perp B \mid \mathbf{Z}$ (marginal likelihood), and $P(D_{OBS}|\bar{r})$ is the network score of $A \not\perp B \mid \mathbf{Z}$ ($A \rightarrow B$ or $A \leftarrow B$).

Similarly, Jabarri et al. (2017) used the BDeu score, which we describe in subsection 2.2.2.1, to obtain a posterior probability for CI in the hybrid RFCI-BSC algorithm, and assumed a uniform prior as the uninformative probability for each result obtained from CI tests as follows:

$$P(r|D_{OBS}) = \frac{P(D_{OBS}|r)}{P(D_{OBS}|r) + P(D_{OBS}|\bar{r})}$$

where $P(D_{OBS}|r)$ is the BDeu score (marginal likelihood) of structure $A \leftarrow \mathbf{Z} \rightarrow B$ ($A \perp B \mid \mathbf{Z}$), and $P(D_{OBS}|\bar{r})$ is the BDeu score of structure $A \leftarrow \mathbf{Z} \rightarrow B$ and $A \rightarrow B$ ($A \not\perp B \mid \mathbf{Z}$), and all variables in $\mathbf{Z}$ are parents of both A and B. These structures are proposed by Jabarri et al. (2017; 2020) to be the representation of all possible structures that correspond to the relevant CI tests. Since the marginal likelihoods can be found in the objective scores computed by score-based learning (Margaritis, 2005), the BDeu score of these structures can be used to derive the marginal likelihoods for discrete variables.

5.2 The mFGS-BS algorithm

The mFGS-BS algorithm described in this subsection learns a PAG from both observational and interventional data, under the assumption of causal insufficiency and that the intervened variables are subject to perfect intervention. The novelty of mFGS-BS involves assigning probabilities to each possible directed edge. If the two opposing directions between a pair of variables both have probabilities that are higher than a given threshold, then a bidirected edge is assumed.

We first describe in subsection 5.2.1 how the probabilities of directed edges from a single observational dataset can be obtained, and then describe in subsection 5.2.2 how we extend this concept to cases in which we want to learn a structure from both observational and interventional data. Subsection 5.2.3 provides the overall description of mFGS-BS.

5.2.1 Determining the probabilities of directed edges from a single observational dataset

We devise a new method to determine directed edges that is largely based on the methods of Hyttinen et al. (2014) and Jabbari et al. (2017) that focus on assigning probabilities to each result obtained from CI tests. In this work, we label observational data as D_{OBS} and interventional data as D_{INT}. When assuming the unconditional independence between two nodes A and B, we modify Equation (5.1) to consider the possibility of edges A B (i.e. no edge between A and B), A → B and A ← B in a DAG as follows:

$$P(A\ B|D_{OBS}) = \frac{P(A\ B) \times P(D_{OBS}|A\ B)}{P(A\ B) \times P(D_{OBS}|A\ B) + P(A \rightarrow B) \times P(D_{OBS}|A \rightarrow B) + P(A \leftarrow B) \times P(D_{OBS}|A \leftarrow B)}$$

Since $P(A\ B|D_{OBS}) + P(A \rightarrow B|D_{OBS}) + P(A \leftarrow B|D_{OBS}) = 1$ and $P(A\ B) + P(A \rightarrow B) + P(A \leftarrow B) = 1$, then:

$$1 - (P(A \rightarrow B|D_{OBS}) + P(A \leftarrow B|D_{OBS})) =$$
$$\frac{(1 - (P(A \rightarrow B) + P(A \leftarrow B))) \times P(D_{OBS}|A\ B)}{(1 - (P(A \rightarrow B) + P(A \leftarrow B))) \times P(D_{OBS}|A\ B) + P(A \rightarrow B) \times P(D_{OBS}|A \rightarrow B) + P(A \leftarrow B) \times P(D_{OBS}|A \leftarrow B)} \quad (5.2)$$

where $P(A \rightarrow B)$ is the prior probability of directed edge $A \rightarrow B$, $P(A \leftarrow B)$ is the prior probability of directed edge $A \leftarrow B$ that we later describe in subsection 5.2.2, $P(D_{OBS}|A \rightarrow B)$ is the BDeu score of structure $A \rightarrow B$, and $P(D_{OBS}|A \leftarrow B)$ is the BDeu score of structure $A \leftarrow B$.

Because we assume that the learnt ancestral graph is a PAG that may contain bidirected edges, the bidirected edge $A \leftrightarrow B$ corresponds to the dependency between A and B from the assumed true structure $A \leftarrow L \rightarrow B$ ($A \not\perp B$) where L is a latent confounder. The dependency between A and B in a PAG can be $A \rightarrow B$, $A \leftarrow B$ or $A \leftrightarrow B$. Because Equation (5.2) is not suitable to calculate the posterior probabilities of these types of edges, we devise two equations: (1) calculating $P(A \rightarrow B|D_{OBS})$ by ignoring $A \leftarrow B$, as described in Case 1 below, and (2) calculating $P(A \leftarrow B|D_{OBS})$ by ignoring $A \rightarrow B$, as described in Case 2 below. These enable us to calculate the probabilities of each of these directed edges independently. If the posterior probabilities of both directed edges $A \rightarrow B$ and $A \leftarrow B$ are higher than a given threshold, then mFGS-BS will not be able to orientate the given directed edges and will produce the bidirected edge $A \leftrightarrow B$.

Case 1: Calculate $P(A \rightarrow B|D_{OBS})$ given the assumption that $P(A \leftarrow B|D_{OBS}) = 0$, $P(D_{OBS}|A \leftarrow B) = 0$ and $P(A \leftarrow B) = 0$ from Equation (5.2), then:

$$1 - P(A \rightarrow B|D_{OBS}) = \frac{(1 - P(A \rightarrow B)) \times P(D_{OBS}|A\ B)}{(1 - P(A \rightarrow B)) \times P(D_{OBS}|A\ B) + P(A \rightarrow B) \times P(D_{OBS}|A \rightarrow B)}$$

Case 2: Calculate $P(A \leftarrow B|D_{OBS})$ given the assumption that $P(A \rightarrow B|D_{OBS}) = 0$, $P(D_{OBS}|A \rightarrow B) = 0$ and $P(A \rightarrow B) = 0$ from Equation (5.2), then:

$$1 - P(A \leftarrow B|D_{OBS}) = \frac{(1 - P(A \leftarrow B)) \times P(D_{OBS}|A\ B)}{(1 - P(A \leftarrow B)) \times P(D_{OBS}|A\ B) + P(A \leftarrow B) \times P(D_{OBS}|A \leftarrow B)}$$

From this, we define the posterior probabilities of directed edges as specified by **Definition 6**.

Definition 6: Assuming the learnt graph is a PAG, we define a bidirected edge $A \leftrightarrow B$ as the dependency between A and B derived from the possibility of both $A \rightarrow B$ and $A \leftarrow B$, where the posterior probabilities $P(A \rightarrow B|D_{OBS})$ and $P(A \leftarrow B|D_{OBS})$ are:

$$P(A \rightarrow B|D_{OBS}) = 1 - \frac{(1 - P(A \rightarrow B)) \times P(D_{OBS}|A\ B)}{(1 - P(A \rightarrow B)) \times P(D_{OBS}|A\ B) + P(A \rightarrow B) \times P(D_{OBS}|A \rightarrow B)}$$

$$P(A \leftarrow B|D_{OBS}) = 1 - \frac{(1 - P(A \leftarrow B)) \times P(D_{OBS}|A\ B)}{(1 - P(A \leftarrow B)) \times P(D_{OBS}|A\ B) + P(A \leftarrow B) \times P(D_{OBS}|A \leftarrow B)}$$

5.2.2 Determining the probabilities of directed edges from both observational and interventional datasets

Cooper and Yoo (1999) propose a Bayesian score for DAG structures that assumes a mixture of causally sufficient observational and experimental data simultaneously. However, the proposed closed-form solution assumes that exhaustive enumeration of DAG structures is possible, and this renders the process computationally expensive or intractable when applied to today's larger datasets. In this work, we extend the approach described in subsection 5.2.1 to learn from an observational dataset and one or more interventional datasets, which the algorithm processes in turn to improve computational efficiency. The proposed mFGS-BS algorithm computes the posterior probabilities of directed edges derived from previously processed data, where each posterior probability of a given edge serves as the prior probability of that edge in the next iteration. For each interventional dataset, INT_i, the algorithm uses Equations (5.3) and (5.4) to determine the posterior probability of each directed edge. We use the term "posterior" here to reflect the fact that this probability, denoted for example, $P(A \rightarrow B|D_{INT_i})$, is based ***both*** on the current interventional dataset being processed ***and*** all previous datasets processed.

$$P(A \rightarrow B|D_{INT_i}) = 1 - \frac{(1-P(A \rightarrow B)) \times P(D_{INT_i}|A\ B)}{(1-P(A \rightarrow B)) \times P(D_{INT_i}|\ A\ B) + P(A \rightarrow B) \times P(D_{INT_i}|A \rightarrow B)} \quad (5.3)$$

$$P(A \leftarrow B|D_{INT_i}) = 1 - \frac{(1-P(A\leftarrow B))\times P(D_{INT_i}|\,A\,B)}{(1-P(A\leftarrow B))\times P(D_{INT_i}|\,A\,B)+P(A\leftarrow B)\times P(D_{INT_i}|A\leftarrow B)} \quad (5.4)$$

The term $P(A \rightarrow B)$ on the right hand side of Equation (5.3) represents the objective prior probability of directed edge $A \rightarrow B$ based on the previously processed datasets. The term $P(A \leftarrow B)$ plays an analogous role as the objective prior for $A \leftarrow B$ in Equation (5.4). The prior for $A \rightarrow B$ is taken to be ***either*** the posterior for that directed edge computed in the previous iteration, that is, $P(A \rightarrow B|D_{INT_{i-1}})$, ***or*** a prior derived using Equation (5.5) ***whichever is the larger***.

$$P(A \rightarrow B) = \max\{P_{FGS}(A \rightarrow B)|D_{OBS,INT_{1:i-1}}, P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}\} + \sum_{k=1}^{i-1} P(A - B)_{\text{local BDeu of B,target =A}}|D_{OBS,INT_k} \quad (5.5)$$

where $P(A \rightarrow B)$ is computed from three factors on the right hand side of Equation (5.5):

a) Factor 1: $P_{FGS}(A \rightarrow B)|D_{OBS,INT_{1:i-1}}$ is the probability of directed edge $A \rightarrow B$ over all previously learnt CPDAGs from FGS across $D_{OBS,INT_{1:i-1}}$ (further details are provided in subsection 5.2.2.1).

b) Factor 2: $P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$ is the probability of directed edge $A \rightarrow B$ calculated from the ratio of Sepsets determining v-structure $A \rightarrow B \leftarrow C$ using the majority rule from D_{OBS} (further details are provided in subsection 5.2.2.2).

c) Factor 3: $\sum_{k=1}^{i-1} P(A - B)_{\text{local BDeu of B,target =A}}|D_{OBS,INT_k}$ is the summation of all relative changes in the local BDeu scores of node B compared to D_{OBS}, when the intervened variable is A across all previously learnt D_{INT}. The relative changes in the local BDeu scores are described in subsection 5.2.2.3).

5.2.2.1 Factor 1: Determining the probabilities of directed edges given the occurrence rates of each directed edge over all learnt CPDAGs

The first, out of the three, factors used to calculate the prior probability of a directed edge is based on the occurrence rate of each directed edge derived from the probability of directed edge $A \rightarrow B$ ($P_{FGS}(A \rightarrow B)|D_{OBS,INT_{1:i-1}}$) over all learnt CPDAGs obtained by applying FGS to each input dataset. Specifically,:

$$P_{FGS}(A \rightarrow B)|D_{OBS,INT_{1:i-1}} = \frac{\#\text{directed edge}(A \rightarrow B)}{\#\text{total directed edge}(A \rightarrow B) + \#\text{total directed edge}(A \leftarrow B)}$$

where:

$$\text{directed edge}(A \rightarrow B) = \begin{cases} 1 & : \text{if } A \rightarrow B \text{ is in a learnt CPDAG} \\ 0.5 & : \text{if } A - B \text{ is in a learnt CPDAG and the intervened variable} = A \\ 0 & : \text{otherwise} \end{cases}$$

and:

$$\text{total directed edge}(A \rightarrow B) = \begin{cases} 1: & \text{if } A \rightarrow B \text{ is in a learnt CPDAG} \\ 1: & \text{if } A - B \text{ is in a learnt CPDAG and the intervened variable} = A \\ 0: & \text{otherwise} \end{cases}$$

The total number of directed edges $A \rightarrow B$ represents the number of directed edges $A \rightarrow B$ present in each of the learnt CPDAGs. These formulas rely on a simple counting method to calculate probabilities, adopted by Hyttinen et al. (2014) who estimate prior probabilities of directed edges specified by experts. Note that CPDAGs learnt from interventional data should not produce directed edges entering the intervened variable due to the graph surgery mechanisms illustrated in Figure 2.4 of Chapter 2 (i.e., interventions are rendered independent of their parents). For example, if the undirected edge $A - B$ is present in the learnt CPDAG when we intervene on node A, the algorithm assigns probability 0 for directed edge $A \leftarrow B$ and probability 0.5 for directed edge $A \rightarrow B$ to account for the risk of false positive edges learnt by FGS, since it does not produce bidirected edges in the presence of latent confounders (Ogarrio et al., 2016).

It is important to clarify that in the absence of intervention, an undirected edge in the learnt CPDAG does not imply equal probability for either direction (Kummerfeld, 2021). The correct probability for each directed edge can be obtained by enumerating all possible DAGs from the learnt CPDAG. However, this tends to increase the computational complexity of the algorithm substantially, especially in the case of mFGS-BS which is designed to produce a CPDAG for each input dataset. For simplicity and reasons of efficiency, when an undirected edge is present in a learnt CPDAG, mFGS-BS assumes a probability of 0.5 for either direction.

5.2.2.2 Factor 2: Determining the probabilities of directed edges given the ratios of Sepsets determining v-structures

Because the joint probability distribution from interventional data will not capture all dependencies, we consider the v-structures as determined by observational data. Therefore, interventional data is not used by this factor. In mFCI, the v-structures are obtained from unshielded triples that are part of an initial undirected graph determined by statistical CI tests. Then, the majority rule in mFCI is used to definitively orientate the edges of unshielded triples $A - B - C$ into v-structures $A \rightarrow B \leftarrow C$, determined by the ratio of Sepsets (Colombo and Maathuis, 2014). In this work, we use a novel method to instead calculate the probabilities of these directed edges, where $P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$ and $P(C \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$ correspond to the individual probabilities of directed edges $A \rightarrow B$ and $C \rightarrow B$ in producing v-structure $A \rightarrow B \leftarrow C$ given the observational data. In order to assign a probability to directed edges in an unshielded triple $A - B - C$, mFGS-BS considers how many of the Sepsets of A and C contain B. If B is in less than 50% of the Sepsets of A and C (i.e., the ratio of Sepsets < 0.5) then we assume that B does not block an active path between A and C. Hence, the likelihood of v-structure $A \rightarrow B \leftarrow C$ will be higher than 0.5, and from this we deduce that $P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS} > 0.5$ and $P(C \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS} > 0.5$. Conversely, if B is in ≥ 50% of the Sepsets of A and C, we deduce that the unshielded triple $A - B - C$ is unlikely to be a v-structure and that instead is likely to be either $A \rightarrow B \rightarrow C$, $A \leftarrow B \rightarrow C$ or $A \leftarrow B \leftarrow C$. These assumptions lead to Equations (5.6) and (5.7) which are calculated independently as follows:

$$P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS} = P(C \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS} = \begin{cases} \text{1-the ratio of Sepset} & : \text{the ratio of Sepset } < 0.5 \\ 0.5 & : \text{the ratio of Sepset } \geq 0.5 \end{cases} \quad (5.6)$$

$$P(A \leftarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS} = P(C \leftarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS} = 0.5 \quad (5.7)$$

where the ratio of Sepsets $= \frac{|\text{Sepsets of A and C which contain B}|}{|\text{all Sepsets of A and C}|}$, $|\text{Sepsets of A and C which contain B}|$ and $|\text{all Sepsets of A and C}|$ represent the number of Sepsets in D_{OBS}. $P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$, $P(C \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$ from Equation (5.6), $P(A \leftarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$ and $P(C \leftarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$ from Equation (5.7) are assigned the value of 0.5 for the reasons covered in subsection 5.2.2.1.

5.2.2.3 Factor 3: Determining the probability of directed edges given the relative changes in local BDeu scores

BDeu is a score-equivalent function where the BDeu score of a graph represents the summation of all local BDeu scores assigned to each node within that graph. The local BDeu score for node i (Z_i) (Cussens, 2012) is denoted as:

$$Z_i = \sum_{j=1}^{q_i}\left[\log\frac{\Gamma(iss/q_i)}{\Gamma(iss/q_i + \Sigma_k n_{ijk})} + \sum_{k=1}^{|X_i|}\log\frac{\Gamma(iss/|X_i|q_i + n_{ijk})}{\Gamma(iss/|X_i|q_i)}\right]$$

The effect of an intervention represents the difference between pre and post-intervention distributions of the children of a target node (Zhang, 2006). We consider the difference in their local BDeu scores to represent the effect of the intervention, assuming the sample size of the input observational data is the same with the sample size of the interventional data when computing this difference. From this, we obtain the relative change in the local BDeu scores as described by **Definition 7**.

Definition 7: Assuming equal sample size for both observational and interventional data, the relative change in the local BDeu scores between pre-intervention ($Z_i|D_{OBS}$) and post-intervention ($Z_i|D_{INT}$) of node i is:

$$\left|\frac{Z_i|D_{OBS} - Z_i|D_{INT}}{Z_i|D_{OBS}}\right| \quad (5.8)$$

For example, when we intervene on node A when $A \rightarrow B$ is present in the graph, then we would expect the effect of this intervention to be reflected in the probability distribution of B. When A is the intervened variable and the undirected edge $A - B$ is learnt by FGS given D_{INT}, we are interested in the likelihood of the directed edge $A \rightarrow B$ being present in the true graph. In this case, the probability of directed edge $A \rightarrow B$ is measured by Factor 3 in terms of the relative change in the local BDeu score of node B, given D_{INT} and D_{OBS}, as defined by Equation (5.8).

Example 1. This example is described with reference to Figure 5.2, and assumes that the true DAG is the one shown in Figure 2.7 of Chapter 2. Figure 5.2a shows the undirected graph as constructed by the CI tests given D_{OBS}, to determine unshielded triples. Figures 5.2b, 5.2c and 5.2d present the three hypothetical CPDAGs learnt by FGS from three different datasets. We first illustrate how to derive Factor 2 in Table 5.2, where the first column shows that the CI tests over V and Y, given the unshielded triple $V - X - Y$ in Figure 5.3a, return 3 Sepsets with p-values greater than the significant threshold α of 0.05. The only Sepset of node V and Y that contains X is $\{W, X, Z\}$. This means that the ratio of Sepsets in determining the given v-structure will be 0.333, as shown in the second column in Table 5.2. The third and fourth columns show how we arrive at the calculation of Factor 2, given Equations (5.6) and (5.7) respectively, each of which corresponds to a probability of the directed edge being present in the true graph.

Sepsets of V and Y	the ratio of Sepsets containing X	Factor 2: $P(V \rightarrow X)_{V\rightarrow X\leftarrow Y} \mid D_{OBS}$ $P(Y \rightarrow X)_{V\rightarrow X\leftarrow Y} \mid D_{OBS}$ given Equation (5.6)	Factor 2: $P(V \leftarrow X)_{V\rightarrow X\leftarrow Y} \mid D_{OBS}$ $P(Y \leftarrow X)_{V\rightarrow X\leftarrow Y} \mid D_{OBS}$ given Equation (5.7)
$\{W\}$ $\{W, X, Z\}$ $\{Z\}$	$\frac{1}{3} = 0.333$	1-0.333 $= 0.667$	0.5

Table 5.2 How the probabilities of directed edges of Factor 2 are calculated, given the unshielded triple $V - X - Y$ in Example 1 and with reference to Figure 5.2a.

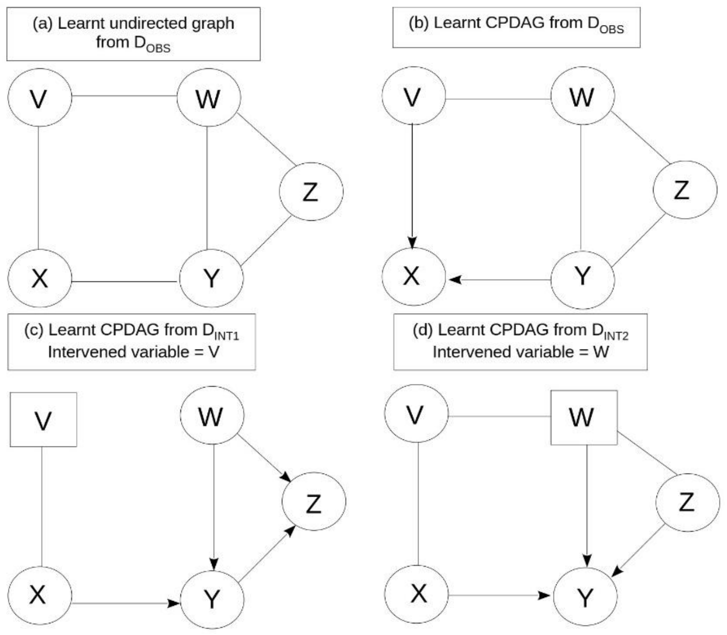

(a) Learnt undirected graph from D_{OBS}
(b) Learnt CPDAG from D_{OBS}
(c) Learnt CPDAG from D_{INT1} Intervened variable = V
(d) Learnt CPDAG from D_{INT2} Intervened variable = W
V
W
Z
X
Y

Figure 5.2 (a) The undirected graph produced by the CI tests given D_{OBS}, (b)-(d) and the three CPDAGs learnt by FGS from observational and interventional data (D_{OBS} , D_{INT_1} and D_{INT_2}) generated based on the DAG shown in Figure 2.7 of Chapter 2, with variables targeted for intervention T_1={V}, T_2={W} shown in the square boxes.

Table 5.3 illustrates how Factor 3 is calculated, that produces the relative change in the local BDeu scores as described in subsection 5.2.2.3. The example is based on one observational dataset, two interventional datasets, and one intervened variable per interventional dataset as shown in Figure 5.2c and Figure 5.2d. Figure 5.2c shows that the undirected edge $V - X$ is learnt by FGS given D_{INT_1}. When V is the intervened variable, we observe that the relative change in the local BDeu score of node X is 0.0119 from the effect of this intervention, so this increases the probability of directed edge $V \rightarrow X$ being present in the true graph. Table 5.3 also shows the relative changes in the local BDeu score of V and Z are 0.0174 and 0.0001 respectively when W is the intervened variable in Figure 5.2d.

Directed edges	Interventional datasets	Intervened variables	Local BDeu score (pre-intervention)	Local BDeu score (post-intervention)	Relative change in local BDeu scores given Equation (5.8)
$V \rightarrow X$	D_{INT_1}	V	X = -11,507	X = -11,370	0.0119
$W \rightarrow V$	D_{INT_2}	W	V = -14,274	V = -14,026	0.0174
$W \rightarrow Z$	D_{INT_2}	W	Z = -6,936	Z = -6,935	0.0001

Table 5.3 An example of calculating the relative change in the local BDeu scores as described in Example 1 and with reference to Figure 5.2c and Figure 5.2d.

Finally, Table 5.4 presents the outputs produced by each of the three factors, and with reference to the directed edges presented in the first column. The calculations in the second, third and fourth columns correspond to the outputs of Factors 1, 2 and 3 respectively. In calculating Factor 1 for directed edge $X \rightarrow Y$, Figures 5b, 5c and 5d show that $X \leftarrow Y$ appears once and $X \rightarrow Y$ appears twice across the three CPDAGs, thus $P_{FGS}(X \rightarrow Y)|D_{OBS,INT_{1:2}} = 0.67$. For directed edge $W \rightarrow V$, Figure 5.2b shows $W - V$, Figure 5.2c shows no edge, and Figure 5.2d shows $W - V$ given D_{INT_2} and hence, $P_{FGS}(V \rightarrow W)|D_{OBS,INT_{1:2}}$ is set to 0 and $P_{FGS}(W \rightarrow V)|D_{OBS,INT_{1:2}}$ to 0.5. This is because W is the intervened variable in Figure 5.2d, and from this we can conclude that if an edge is discovered between V and W, then the direction of that edge can only be entering V. Note that $P_{FGS}(W \rightarrow V)|D_{OBS,INT_{1:2}}$ is set to 0.5 and not to 1 because FGS suggests $W - V$ instead of $W \rightarrow V$. Finally, the fifth column of Table 5.4 shows the overall calculation for the prior probability of each directed edge, that takes into consideration all three factors, given Equation (5.5).

Directed edges	**Factor 1:** $P_{FGS}(A \rightarrow B)\|D_{OBS,INT_{1:2}}$	**Factor 2:** $P(A \rightarrow B)_{A \rightarrow B \leftarrow C}\|D_{OBS}$	**Factor 3:** $\sum_{k=1}^{2} P(A - B)_{\text{local BDeu of B,target} = A}\|D_{OBS,INT_k}$	$P(A \rightarrow B)$ **given Equation (5.5)**
$X \rightarrow Y$	0.67	0.5	-	0.67
$Y \rightarrow X$	0.34	0.67	-	0.67
$V \rightarrow X$	0.75	0.67	0.0119	0.7619
$W \rightarrow V$	0.5	0	0.0174	0.5174
$V \rightarrow W$	0	0	-	0
$W \rightarrow Y$	1	0	-	1
$W \rightarrow Z$	0.75	0	0.0001	0.7501

Table 5.4 Examples of the calculation of the prior probability of directed edges with reference to Example 1, Figure 5.2, Table 5.2 and Table 5.3.

5.2.3 Algorithm mFGS-BS

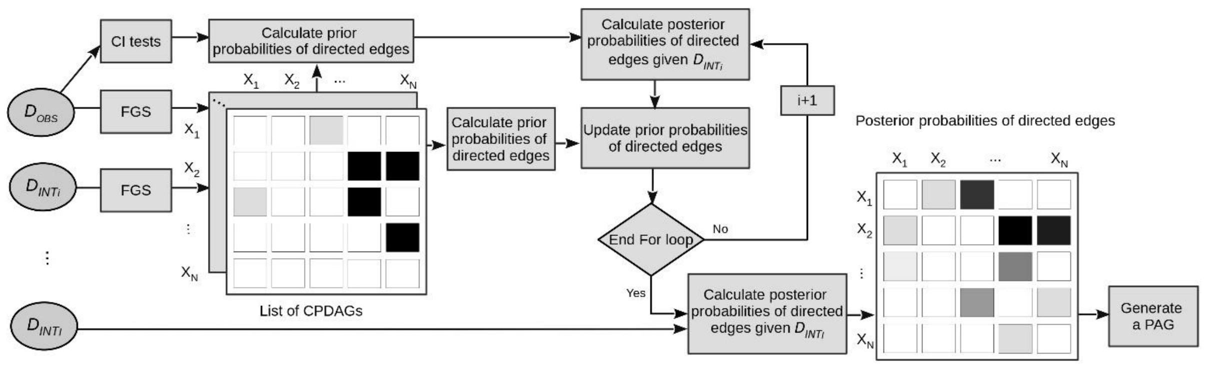

CI tests
Calculate prior probabilities of directed edges
Calculate posterior probabilities of directed edges given D_{INT_i}
D_{OBS}
FGS
D_{INT_i}
FGS
X_1 X_2 ... X_N
Calculate prior probabilities of directed edges
Update prior probabilities of directed edges
i+1
Posterior probabilities of directed edges
End For loop
No
Yes
List of CPDAGs
D_{INT_i}
Calculate posterior probabilities of directed edges given D_{INT_i}
Generate a PAG

Figure 5.3 The overall process of the mFGS-BS algorithm that iteratively processes datasets and calculates posterior probabilities of directed edges to generate a PAG.

We now use the concepts described in subsections 5.2.1 and 5.2.2 to formulate the mFGS-BS algorithm. The pseudocode of mFGS-BS is provided in Algorithm 2. The algorithm takes as an input an observational dataset and one or more interventional datasets, the set of variables targeted for intervention for each interventional data, and the hyperparameters specified in Algorithm 2. The overall process of mFGS-BS is shown in Figure 5.3. The first step in Algorithm 2 performs CI tests given an observational dataset. Steps 2 to 4 derive the initial prior probabilities of directed edges forming v-structures and the probabilities of directed edges learnt by FGS given an observational dataset. Step 5 then iteratively calculates the posterior probabilities of directed edges derived from each interventional dataset ($D_{INT_1} - D_{INT_{I-1}}$), as described in subsection 5.2.2. In the last steps, a PAG is constructed from the posterior probabilities of directed edges obtained after processing the last interventional dataset (D_{INT_I}), based on a hyperparameter cut-off threshold used to determine the existence of a directed edge or bidirected edge.

Algorithm 2: mFGS-BS (majority rule and Fast Greedy equivalent Search with Bayesian Scoring)

```
Input: interventional datasets $D_{INT_I}$, an observational data $D_{OBS}$, intervened variable sets $T_I$, significance threshold $\alpha$,
posterior probability cut-off threshold c, maximum Sepset size k, CI test
Output: a PAG

Step 1  Set up a complete undirected graph $\mathcal{U}$ and Sepset Z size = 0
        Repeat
            Remove the dependencies for each pair $(A, B)$ in $\mathcal{U}$ if they become independent given subsets of
            Sepset Z, determined by significance threshold $\alpha$ in $D_{OBS}$
            Sepset Z size = Sepset Z size +1
        Until Sepset Z size k has been tested
Step 2  Given unshielded triple $A - B - C$ from $\mathcal{U}$ resulting from Step 1, perform CI tests, with significance
        threshold $\alpha$, on A and C given all neighbours of A and C including B, given $D_{OBS}$ and calculate Factor 2
        $P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$, $P(A \leftarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$, $P(C \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$ and $P(C \leftarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}$
        according to Equations (5.6) and (5.7)
Step 3  Run FGS on $D_{OBS}$ and add the learnt CPDAG to the list $\mathcal{L}_G$
Step 4  For each pair $(A, B)$ over all variables
            Calculate the prior probabilities $P(A \rightarrow B), P(A \leftarrow B)$ for each possible directed edge $A \rightarrow B$, $A \leftarrow B$
                where $P(A \rightarrow B) = \max\{P_{FGS}(A \rightarrow B)|D_{OBS}, P(A \rightarrow B)_{A\rightarrow B\leftarrow C}|D_{OBS}\}$
            Calculate the posterior probabilities $P(A \rightarrow B|D_{INT_1}), P(A \leftarrow B|D_{INT_1})$ for each possible directed edge
                $A \rightarrow B$, $A \leftarrow B$ according to Equations (5.3) and (5.4)
Step 5  For i=1 to I-1
            Run FGS on $D_{INT_i}$ and add the learnt CPDAG to the list $\mathcal{L}_G$
            For each pair $(A, B)$ over all variables
                If A is the intervened variable
                    Calculate Factor 3 $P(A - B)_{\text{local BDeu of B,target=A}}|D_{OBS,INT_i}$ given Equation (5.8) and add it to
                        the list $\mathcal{L}_S$

                Calculate the prior probabilities $P(A \rightarrow B), P(A \leftarrow B)$ for each possible directed edge $A \rightarrow B$ and
                    $A \leftarrow B$ given Equation (5.5), where Factor 1 is calculated given $\mathcal{L}_G$, Factor 2 is calculated
                    given Equations (5.6) and (5.7) in Step 2, and Factor 3 is calculated given $\mathcal{L}_S$
                $P(A \rightarrow B) \longleftarrow \max\{P(A \rightarrow B), P(A \rightarrow B|D_{INT_i})\}$
                $P(A \leftarrow B) \longleftarrow \max\{P(A \leftarrow B), P(A \leftarrow B|D_{INT_i})\}$
                Calculate the posterior probabilities $P(A \rightarrow B|D_{INT_{i+1}}), P(A \leftarrow B|D_{INT_{i+1}})$ for each possible
                    directed edge $A \rightarrow B$, $A \leftarrow B$ according to Equations (5.3) and (5.4)
Step 6  Repeat until no cycles or an almost cyclic are present in the output graph
            For each pair $(A, B)$ in all variables
                Select edge $A \rightarrow B$ if the posterior probability $P(A \rightarrow B|D_{INT_I})$ is higher than threshold c;
                Select edge $A \leftrightarrow B$ if the posterior probabilities $P(A \rightarrow B|D_{INT_I})$ and $P(A \leftarrow B|D_{INT_I})$ are both
                    higher than threshold c;
                Select edge $A$ o— o $B$ if the posterior probabilities $P(A \rightarrow B|D_{INT_I})$ and $P(A \leftarrow B|D_{INT_I})$ are both
                    lower or equal to threshold c, but $A - B$ exists in $\mathcal{U}$.
            If the graph contains a cycle or an almost cyclic
                Remove an edge that causes a cycle, or an almost cycle, where the edge selected is the one that has
                    the lowest posterior probability.
Step 7  Output a PAG by combining the set of edges learnt from Step 6
```

5.3 Evaluation

We consider the same six networks previously presented in Table 3.1 of Chapter 3 that vary in dimensionality. Because the experiments assume multiple interventional datasets, we restrict the evaluation to synthetic experiments that can accurately represent such scenarios since we had no access to suitable real data. We use the networks to generate one synthetic observational and up to 10 synthetic interventional datasets. The true MAGs and true DAGs for each of the networks are available in the Bayesys repository (Constantinou et al., 2020). For each true DAG, we consider observational and interventional datasets over two sample sizes (n = 1k and n = 10k). Interventional data are generated using the *bnlearn* R package (Scutari, 2019). For each dataset, we randomly choose one or five variables to be targeted for intervention. This means it is possible for the same variable to be targeted for intervention in more than one interventional dataset. We remove all incoming edges entering intervened variables, and we assume a uniform distribution for each state of variables targeted for intervention, before the intervention is set, as in (Korb et al., 2004). Finally, 10% of the variables in the smaller networks (Asia and Sports) and 5% of the variables in the larger networks (Property, Alarm, Formed and Pathfinder) are made latent. To minimise uncertainty, we repeat the experiments five times per algorithm and obtain the average scores.

The structure learning performance is evaluated using the graphical measures of Precision, Recall, F1 and BSF as described in subsection 2.2.4. We compare the graphical scores obtained by mFGS-BS to those obtained by COmbINE, RFCI-BSC and GFCI as described in subsections 5.1 and 2.2.3, which are three similar algorithms that also produce a PAG. RFCI-BSC assigns probabilities to CIs that are used to learn a PAG, which is the most similar approach to mFGS-BS, whilst the well-establish GFCI supports latent variables, and has been shown to more accurate than FCI and RFCI (Ogarrio et al., 2016; Constantinou et. al., 2021). An important difference amongst these algorithms is that COmbINE enables learning from multiple interventional datasets while RFCI-BSC and GFCI do not. RFCI-BSC and GFCI are hybrid algorithms which assume the input data are observational. We therefore combine the observational and interventional datasets into a single dataset which we use as input to these algorithms. This serves as a baseline experiment where the RFCI-BSC and GFCI algorithms produce a result given all data but without taking advantage of interventional information.

True edges	Predicted edges	Penalty	Result
A ↔ B	A → B, A ⇢ B, A ← B, A ⇠ B, A B	1	True Positive = 0
A → B	A ↔ B, A ← B, A ⇠ B, A B	1	True Positive = 0
A ← B	A ↔ B, A → B, A ⇢ B, A B	1	True Positive = 0
A B	A → B, A ⇢ B, A ← B, A ⇠ B, A ↔ B	1	True Negative = 0
A → B	Ao— oB, Ao---oB	0.5	True Positive = 0.5
A ← B	Ao— oB, Ao---oB	0.5	True Positive = 0.5
A → B	A o→ B, A o⇢ B	0.25	True Positive = 0.75
A ← B	A ←o B, A ⇠o B	0.25	True Positive = 0.75

Table 5.5 The edge and orientation penalty scores used by the scoring metrics, where ⇢ represents one of the output edges of COmbINE.

COmbINE was tested using the MATLAB implementation by Triantafillou (2019) while RFCI-BSC and GFCI were tested using the *rcausal* package, which is the R wrapper for Tetrad Library (Wongchokprasitti, 2019). The mFGS-BS implementation is available online at https://github.com/kiattikunc/mFGS-BS. Note the output of COmbINE represents a special type of PAG that contains dashed edges (---) indicating uncertainty about the existence of an edge learnt from each interventional dataset. Since we are interested in the direction of causation, all output PAGs are measured against the true MAG using the penalty scores described in Table 5.5. Regarding the hyperparameter inputs of the algorithms, the significant threshold α for the G^2 hypothesis test is set to 0.05, and the max Sepset size of the conditioning set is set to 10, in all algorithms. The posterior probability cut-off threshold of mFGS-BS is set to 0.5, and the default iss of BDeu in mFGS-BS, RFCI-BSC and GFCI is set to 1. We also apply a runtime limit of four hours to each graph learnt/experiment for all algorithms.

5.4 Results

The results are separated into four subsections. We start with subsection 5.4.1, where we measure the sensitivity to the order of interventional datasets. We use the Alarm network to generate 5 and 10 interventional datasets with sample sizes 1k and 10k by intervening on a random single variable per dataset and 5% of the variables in the data are made latent. Then, we randomise 20 orderings of 5 and 10 interventional datasets, and evaluate the results. In subsection 5.4.2, we assess the impact of each of the three factors described in subsection 5.2.2 on graphical learning accuracy. Subsection 5.4.3 compares the results of mFGS-BS to those of the other algorithms when we intervene on a single variable per interventional data set, and subsection 5.4.4 when we intervene on five variables per interventional data set.

5.4.1 Assessing the sensitivity of the ordering of interventional datasets

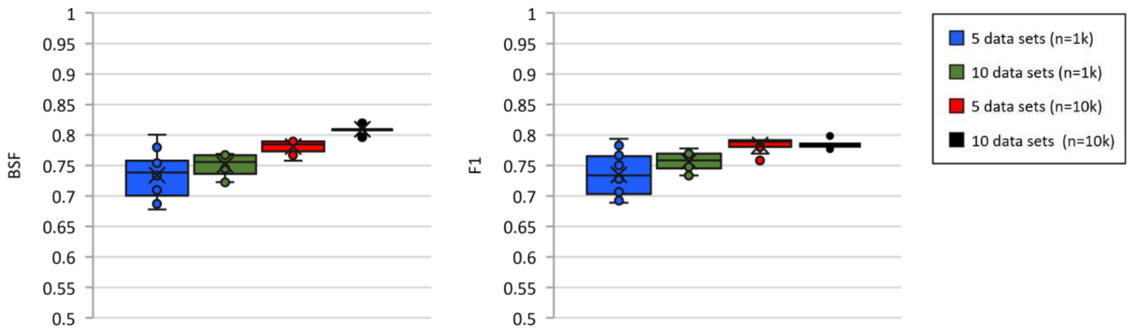


Figure 5.4 The boxplots show the BSF and F1 scores of mFGS-BS from 20 random interventional data orderings generated from the Alarm network, assuming one intervened variable and 5% latent variables per dataset, over two sample sizes and two numbers of interventional datasets. The boxplots report the average values (the symbol x in the box) along with the median (the middle line of the box), and the maximum and minimum scores (the whiskers of the box). The lower edge of the boxplot represents the first quartile, while the higher edge of the boxplot represents the third quartile.

The mFGS-BS algorithm updates the posterior probabilities of directed edges by taking into consideration a single interventional dataset at a time. In this subsection, we evaluate how this ordering might influence the graphical performance of the algorithm. This experiment involves the different combinations of 5 and 10 interventional datasets, and sample sizes 1k and 10k. The boxplot in Figure 5.4 shows the BSF and F1 scores of mFGS-BS when applied to each hyperparameter setting involving the Alarm network. Each of the four sample sizes involves 20 randomised orderings of interventional data. The results show that the average BSF score is 0.73±0.0363 when we have 5 interventional datasets at 1k sample size each, and the variability decreases to 0.81±0.0058 for 10 interventional datasets at 10 sample size each. We observe that the average F1 scores are mostly consistent with the BSF scores. Both the BSF and F1 scores show that there is a minor deviation in the scores obtained from mFGS-BS, depending on the ordering of interventional datasets, and the standard deviation decreases with the number and size of the interventional datasets.

5.4.2 Assessing the impact of Factors 1, 2, and 3

We assess the impact of the three factors described in subsection 5.2.2 by modifying Equation (5.5) to consider one, or combinations of two, factors at a time. As shown in Table 5.6, mFGS-BS-1 refers to considering Factor 1 only, mFGS-BS-23 considers Factors 2 and 3, etc. The impact is measured in terms of graphical accuracy, based on the metrics Precision, Recall, F1 and BSF shown in Table 5.6. The experiments are based on the Alarm network and assume 5% latent variables (one in this case), and sample sizes 1k and 10k.

The results in Table 5.6 depict the average learning performance over 10 experiments, from considering just one interventional dataset to considering 10 interventional datasets. We repeat these experiments five times, and each time we randomly choose a new variable to be targeted for intervention. Considering one factor alone, the results clearly show considerable drop in performance across almost all cases. Combinations of two factors increase performance, particularly when Factor 3 is included in the combination. Although Factor 1 appears to be less important than Factors 2 and 3, considering all three factors (i.e., the default mFGS-BS) does lead to a slightly better overall performance across all combinations.

Metric	**n**	**mFGS-BS**	**mFGS-BS-1**	**mFGS-BS-2**	**mFGS-BS-3**	**mFGS-BS-12**	**mFGS-BS-13**	**mFGS-BS-23**
Precision	1k	0.79	0.45	**0.76**	0.58	0.45	**0.82**	0.78
Recall		0.74	**0.71**	0.56	0.36	0.71	**0.73**	0.58
F1		0.77	0.55	**0.64**	0.44	0.55	**0.77**	0.66
BSF		0.74	0.**65**	0.56	0.36	0.65	**0.73**	0.58
Precision	10k	0.79	0.64	**0.76**	0.63	0.63	**0.78**	0.77
Recall		0.75	**0.74**	0.69	0.55	**0.74**	**0.74**	0.71
F1		0.77	0.68	**0.72**	0.59	0.68	**0.76**	0.74
BSF		0.75	**0.72**	0.69	0.55	0.71	**0.74**	0.70

Table 5.6 The impact of Factors 1, 2 and 3 (refer to subsection 5.2.2) on graphical performance, where mFGS-BS considers all of the three factors (default version), mFGS-BS-1 considers Factor 1 only, mFGS-BS-12 considers Factors 1 and 2 only, etc. The results represent average performance over multiple experiments with synthetic Alarm network data, as described in subsection 5.4.2.

5.4.3 Results based on one variable targeted for intervention per interventional dataset

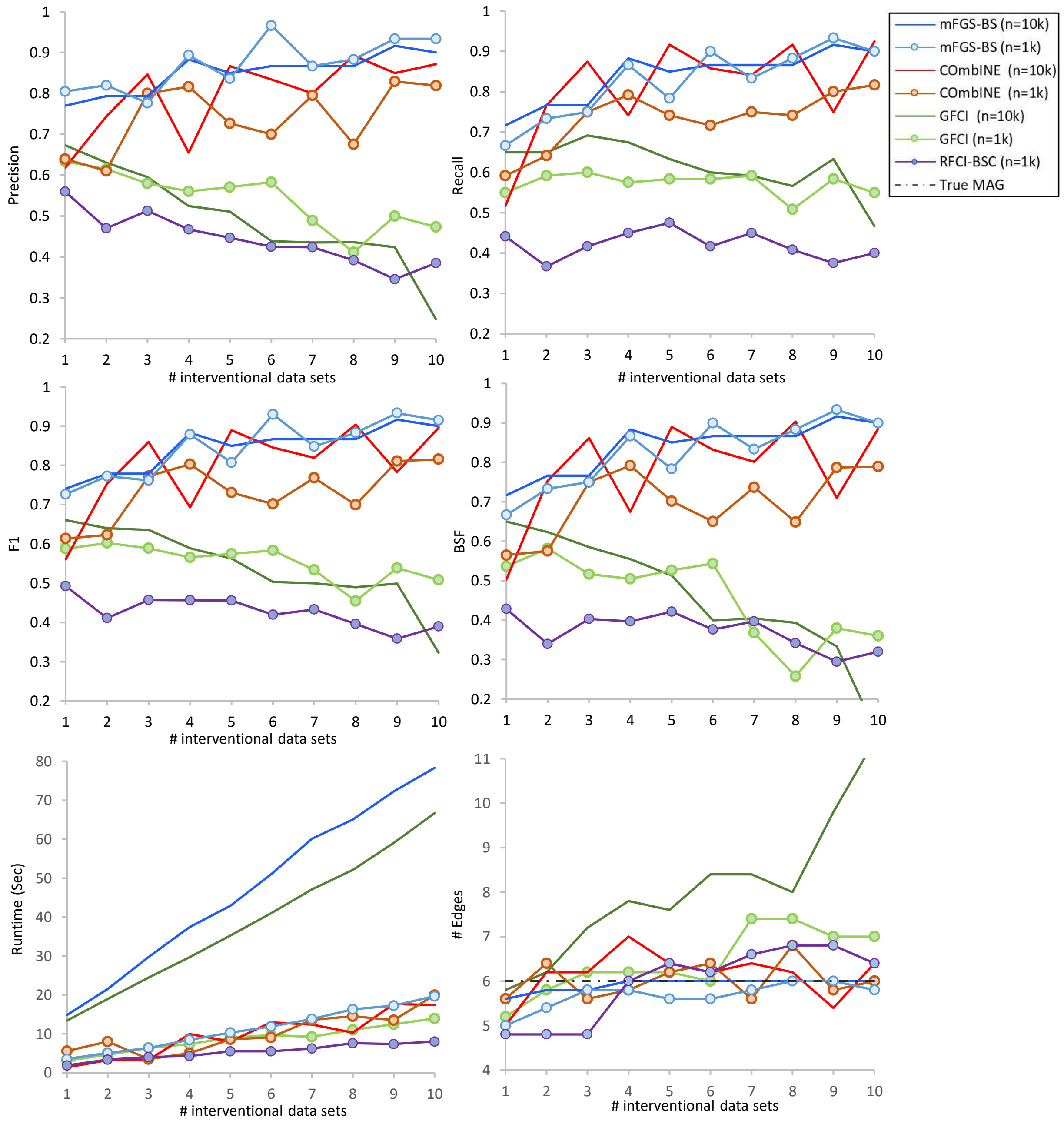


Figure 5.5 Average performance of the algorithms when applied to synthetic data generated from the Asia network, assuming one intervened variable and 10% latent variables per dataset, over two sample sizes.

In this subsection, we assume that each interventional dataset contains a single variable that is randomly targeted for intervention. Because RFCI-BSC failed to generate a PAG within the four-hour runtime limit for almost all cases in which the sample size is 10k (presumably because it generates multiple PAGs through bootstrapping), we restrict its comparisons to experiments where the sample size is up to 1k. Figure 5.5 shows the results obtained by applying the algorithms to the Asia network over two sample sizes. The x-axis represents the total number of interventional datasets considered for learning, and the y-axis represents the specified scoring metric, runtime, or the number of edges learnt. Each data point in these graphs represents the average result across five iterations. Each iteration involves new datasets and new variables targeted for intervention. The results show that mFGS-BS outperforms GFCI and RFCI-BSC, and to a lesser degree COmbINE which demonstrates erratic performance, across all four metrics and two sample sizes. Importantly, the results show that both mFGS-BS and COmbINE continue to improve with the number of interventional datasets. Conversely, the graphical accuracy of GFCI and RFCI-BSC decreases with the number of interventional datasets, and this is expected since these two algorithms use pooled data, where the post-interventional and pre-interventional distributions may conflict. GFCI produces a high number of learnt edges, and this number continues to increase with the number of datasets and greatly surpasses the number of true edges. Lastly, COmbINE is found to be considerably faster than both mFGS-BS and GFCI at 10k sample size.

Figure 5.6 repeats the results for the Sports network, which is also a small network. However, compared to Asia, the Sports network contains a considerably higher number of free parameters. Overall, the results show that the algorithms deliver a rather similar performance when the number of datasets is low, with the gap in performance increasing as the number of datasets increases. The accuracy of mFGS-BS increases faster with the number of datasets, and this eventually makes the gap in performance important at higher number of datasets. This is partly because the accuracy of COmbINE does not improve with the number of interventional datasets, and there is no obvious explanation for this observation. Interestingly, while COmbINE is the fastest algorithm on Asia, it is the slowest on Sports. A possible explanation is the number of free parameters, which is 1,049 in Sports compared to just 18 in Asia, despite the two networks having just one variable difference. This suggests that COmbINE might not scale well with dense networks, or with networks that contain multinomial, rather than Boolean variables, whereas RFCI-BSC fails to return an output and instead returns an out-of-memory error.

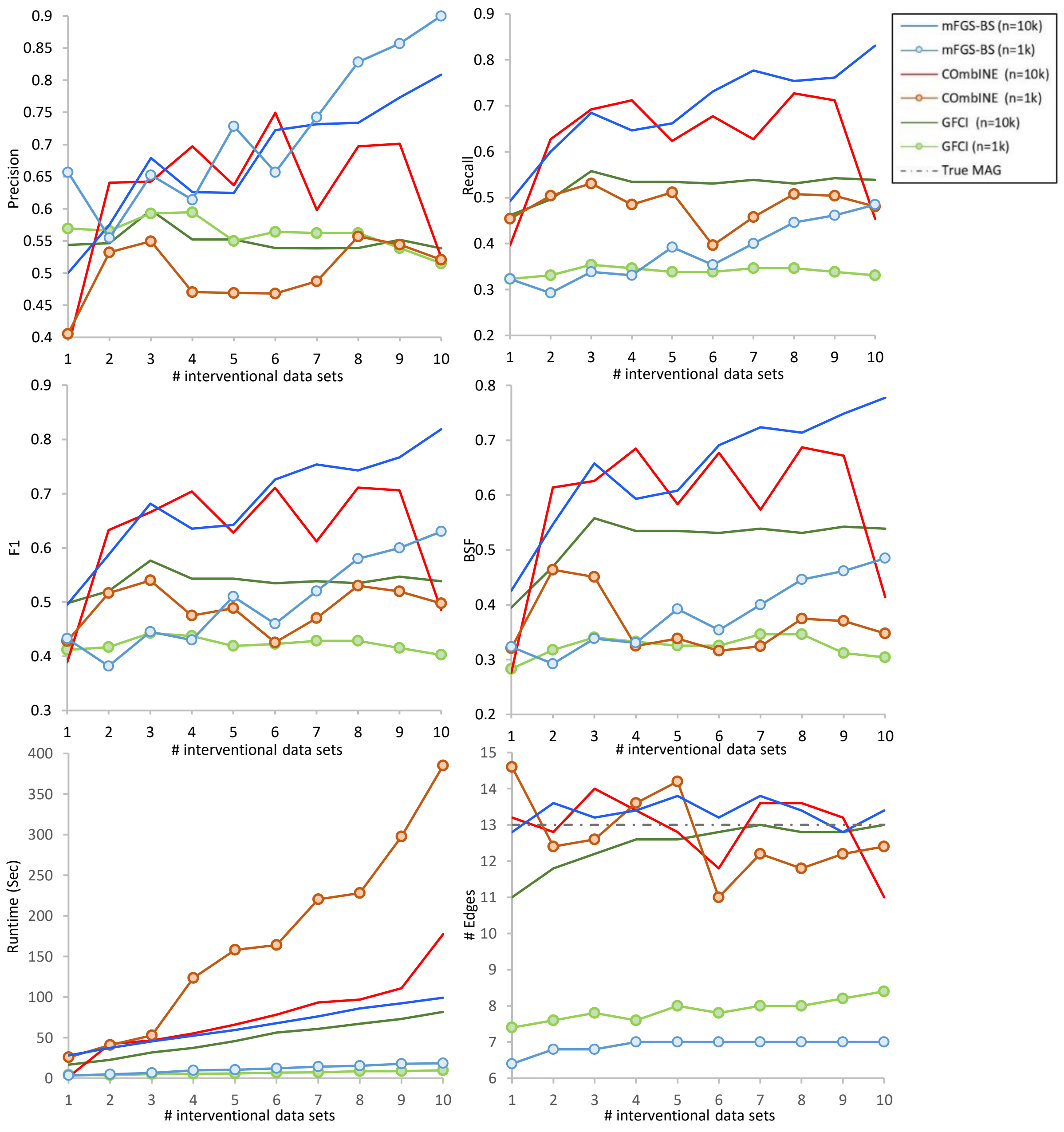


Figure 5.6 Average performance of the algorithms when applied to synthetic data generated from the Sports network, assuming one intervened variable and 10% latent variables per dataset, over two sample sizes.

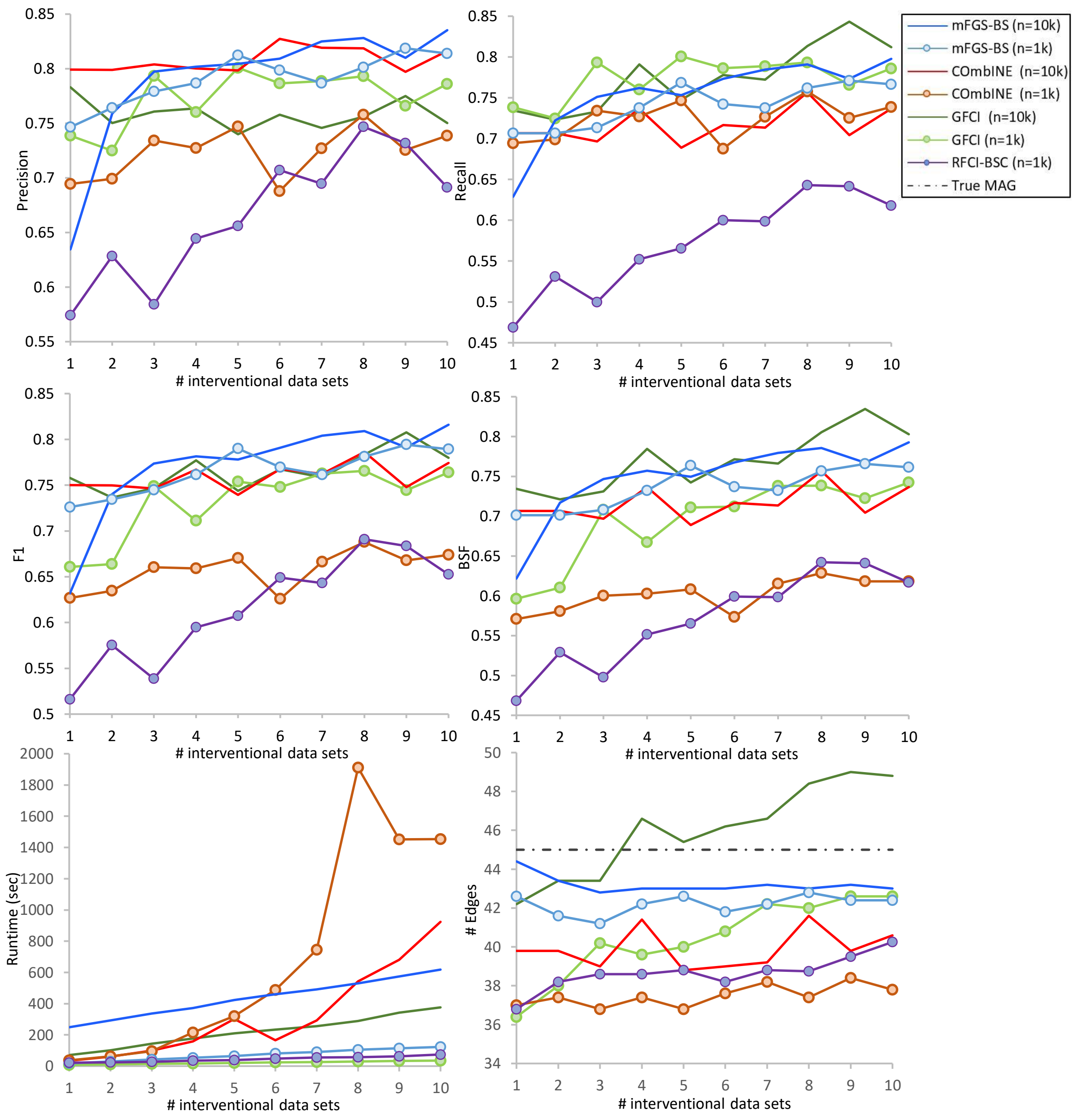


Figure 5.7 Average performance of the algorithms when applied to synthetic data generated from the Alarm network, assuming one intervened variable and 5% latent variables per dataset, over two sample sizes.

Figures 5.7 and 5.8 repeat the results for the medium networks Alarm and Property respectively. While there are some variations in the results, the overall conclusions that can be derived from these results are consistent with those derived from the smaller networks of Asia and Sports. A notable exception is that COmbINE performs better than mFGS-BS, in terms of BSF and recall, in Property. However, this result is restricted to the sample size of 10k, and this is because COmbINE fails to generate a result within the four-hour runtime limit for sample size 1k and RFCI-BSC fails to return a result when the experiments rely on more than two interventional datasets. Because COmbINE does not return a result for any of these larger networks within the four-hour time limit, we have put these results in Appendix A (see Figures A1 and A2). Overall, the larger networks show that GFCI outperforms mFGS-BS slightly in ForMed, perhaps because any differences between the observational and interventional data with just one intervened node is relatively minor in this larger network. mFGS-BS outperforms GFCI considerably in Pathfinder in terms of graphical accuracy. Pathfinder is the network with the highest number of free parameters considered in this study, and this complexity might explain why all algorithms perform relatively poorly on Pathfinder compared to the other networks.

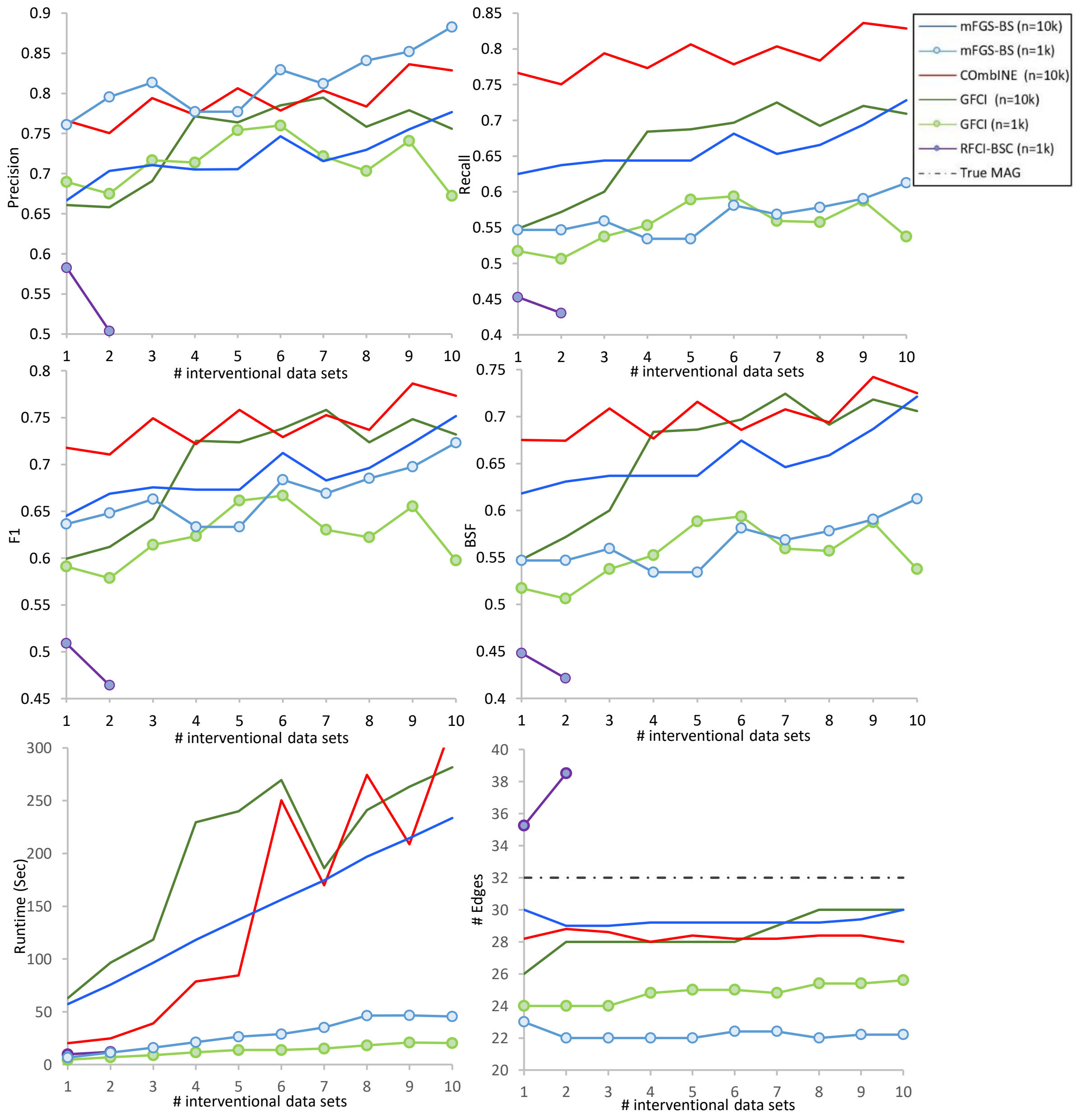


Figure 5.8 Average performance of the algorithms when applied to synthetic data generated from the Property network, assuming one intervened variable and 5% latent variables per dataset, over two sample sizes.

Table 5.7 summarises the average results across all experiments in which a single variable is targeted for intervention. The results show that mFGS-BS performed best in the small and medium networks and across all four graphical metrics, followed by COmbINE, then GFCI and finally RFCI-BSC. In terms of runtime, however, GFCI is found to be the fastest algorithm in most experiments, followed by mFGS-BS, then COmbINE, and finally RFCI-BSC which could not process any of the larger networks within the runtime limit.

Algorithm	n	Asia	Sports	Property	Alarm	ForMed	Pathfinder	Asia	Sports	Property	Alarm	ForMed	Pathfinder
				Precision						**Recall**			
mFGS-BS	1k	**0.87**	**0.72**	**0.81**	**0.79**	**0.90**	**0.29**	**0.83**	0.38	**0.57**	**0.74**	0.46	**0.16**
	10k	**0.85**	**0.68**	0.66	0.79	**0.79**	**0.50**	**0.84**	**0.69**	0.64	0.75	0.67	**0.32**
COmbINE	1k	0.74	0.50	T	0.72	T	T	0.73	**0.48**	T	0.60	T	T
	10k	0.80	0.63	**0.79**	**0.81**	T	T	0.81	0.62	**0.70**	0.72	T	T
GFCI	1k	0.54	0.56	0.71	0.77	0.75	0.16	0.57	0.34	0.55	0.70	**0.56**	0.11
	10k	0.49	0.55	0.74	0.76	0.77	0.12	0.62	0.53	0.66	**0.77**	**0.71**	0.11
RFCI-BSC	1k	0.44	M	0.54	0.67	T	T	0.42	M	0.44	0.57	T	T
				F1						**BSF**			
mFGS-BS	1k	**0.85**	**0.50**	**0.67**	**0.77**	0.61	**0.20**	**0.83**	**0.38**	**0.57**	**0.74**	0.46	**0.14**
	10k	**0.84**	**0.68**	0.65	**0.77**	0.72	**0.39**	**0.84**	**0.65**	0.63	0.75	0.66	**0.31**
COmbINE	1k	0.73	0.49	T	0.66	T	T	0.70	0.36	T	0.60	T	T
	10k	0.80	0.62	**0.74**	0.76	T	T	0.78	0.58	**0.70**	0.72	T	T
GFCI	1k	0.55	0.42	0.62	0.73	**0.64**	0.13	0.46	0.32	0.55	0.69	**0.56**	0.09
	10k	0.54	0.54	0.70	**0.77**	**0.73**	0.12	0.46	0.52	0.66	**0.77**	**0.70**	0.08
RFCI-BSC	1k	0.43	M	0.49	0.62	T	T	0.37	M	0.43	0.57	T	T
				Learnt Edges						**Runtime**			
mFGS-BS	1k	5.68	6.90	22.22	42.18	72.32	124.52	11.26	11.35	28.43	72.21	333.68	1026.19
	10k	5.92	13.34	31.00	43.20	117.86	148.28	47.31	64.44	213.08	434.71	1556.64	2756.12
COmbINE	1k	6.02	12.70	T	37.48	T	T	10.14	169.59	T	677.86	T	T
	10k	6.14	12.94	28.32	39.90	T	T	**9.65**	76.89	**147.34**	325.85	T	T
GFCI	1k	6.44	7.88	24.80	40.44	105.72	161.52	8.68	**6.66**	**14.19**	**21.90**	**39.12**	**63.03**
	10k	8.06	12.46	28.58	46.00	129.90	210.04	38.76	**49.39**	162.28	**219.94**	**1014.59**	**548.98**
RFCI-BSC	1k	5.96	M	36.88	38.65	T	T	**5.37**	M	M	44.59	T	T

Table 5.7 Average performance across all experiments in which a single variable is targeted for intervention per dataset, where M indicates out-of-memory error, and T indicates failure to complete learning within the four-hour runtime limit. The best performance values are shown in bold.

5.4.4 Results based on five variables targeted for intervention per interventional dataset

This subsection focuses on the results when the number of intervened variables is increased from one (subsection 5.4.3) to five, for each interventional data. Because the Asia and Sports networks contain less than 10 variables, we do not consider them here since it would be unrealistic to assume that half of the network variables are targeted for intervention. Instead, we consider the networks of Property, Alarm, ForMed and Pathfinder where the number of variables ranges from 27 to 109.

Figure 5.9 presents the results based on the Property network and shows that both mFGS-BS and COmbINE improve their performance relative to the corresponding results in Figure 5.8 which consider only one intervened variable. Table 5.7, which summarises the average results obtained when considering five intervened variables, shows that mFGS-BS performs best across all metrics at 1k sample size, whereas COmbINE performs best across all metrics at 10k sample size for the Property network. However, as shown in Figure 5.9, the runtime of COmbINE increases much faster with the number of datasets, and fails to generate any results within the four-hour runtime limit when the number of datasets is three or more. RFCI-BSC, on the other hand, returned an out-of-memory error when applied to these datasets. Therefore, the average results reported in Table 5.8 may underestimate the performance of

COmbINE and RFCI-BSC for sample size 1k, since the average is derived solely by focusing on a lower number of datasets on which the performance tends to be worse.

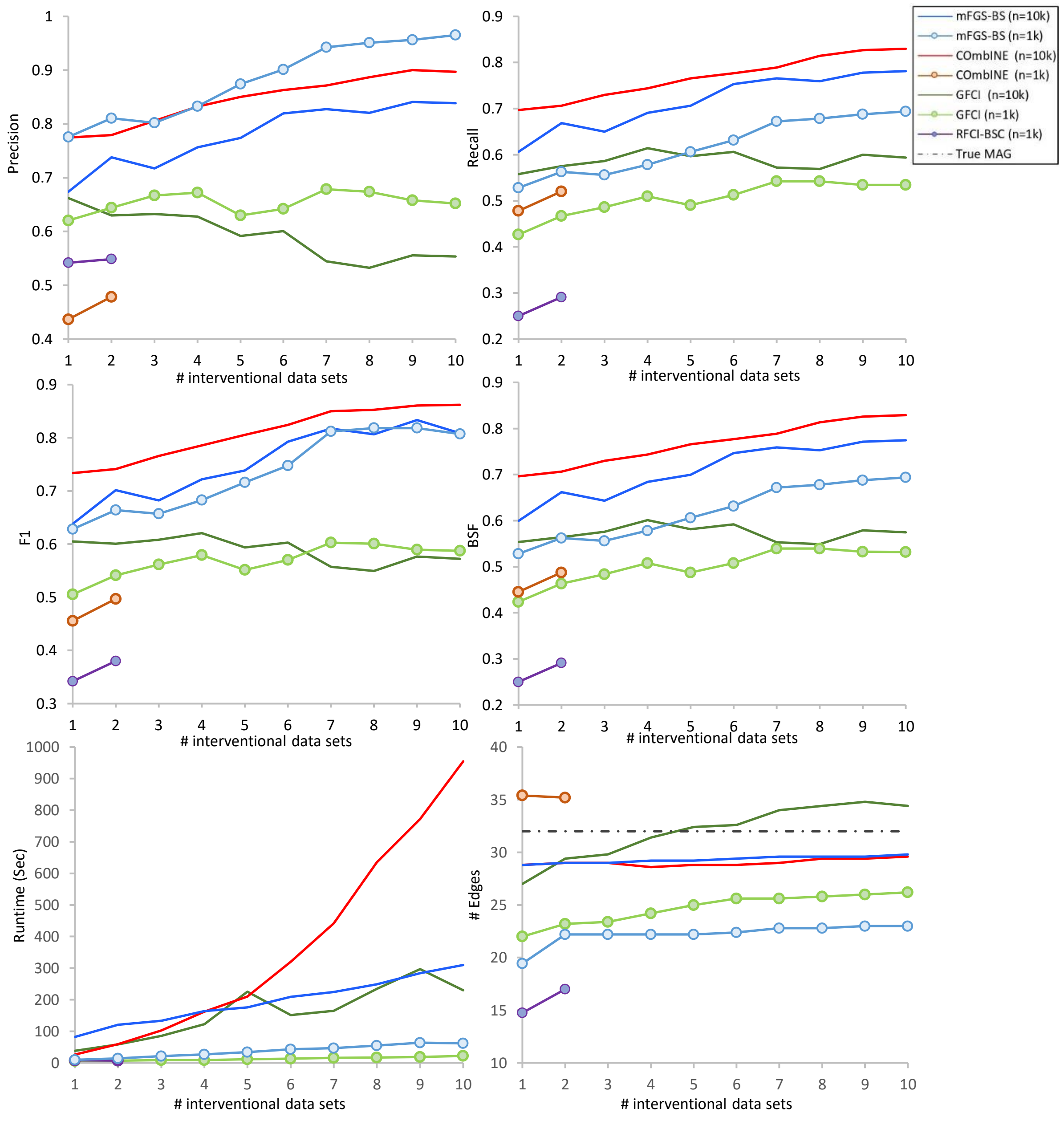


Figure 5.9 Average performance of the algorithms when applied to synthetic data generated from the Property network, assuming five intervened variables and 5% latent variables per dataset, over two sample sizes. The runtime of COmbINE at 1k sample size is not shown in the charts, because its runtime is much higher.

Figure 5.10 repeats the results for the Alarm network. As before, COmbINE failed to produce a result for all experiments within the four-hour time limit. However, the results of COmbINE this time extend up to six interventional datasets and enable us to draw reasonably confident conclusions. mFGS-BS performs best overall and across almost all the different number of datasets and sample sizes. Both mFGS-BS and COmbINE perform better compared to the case of a single intervened variable, and continue to improve with the number of datasets, whereas GFCI and RFCI-BSC do not.

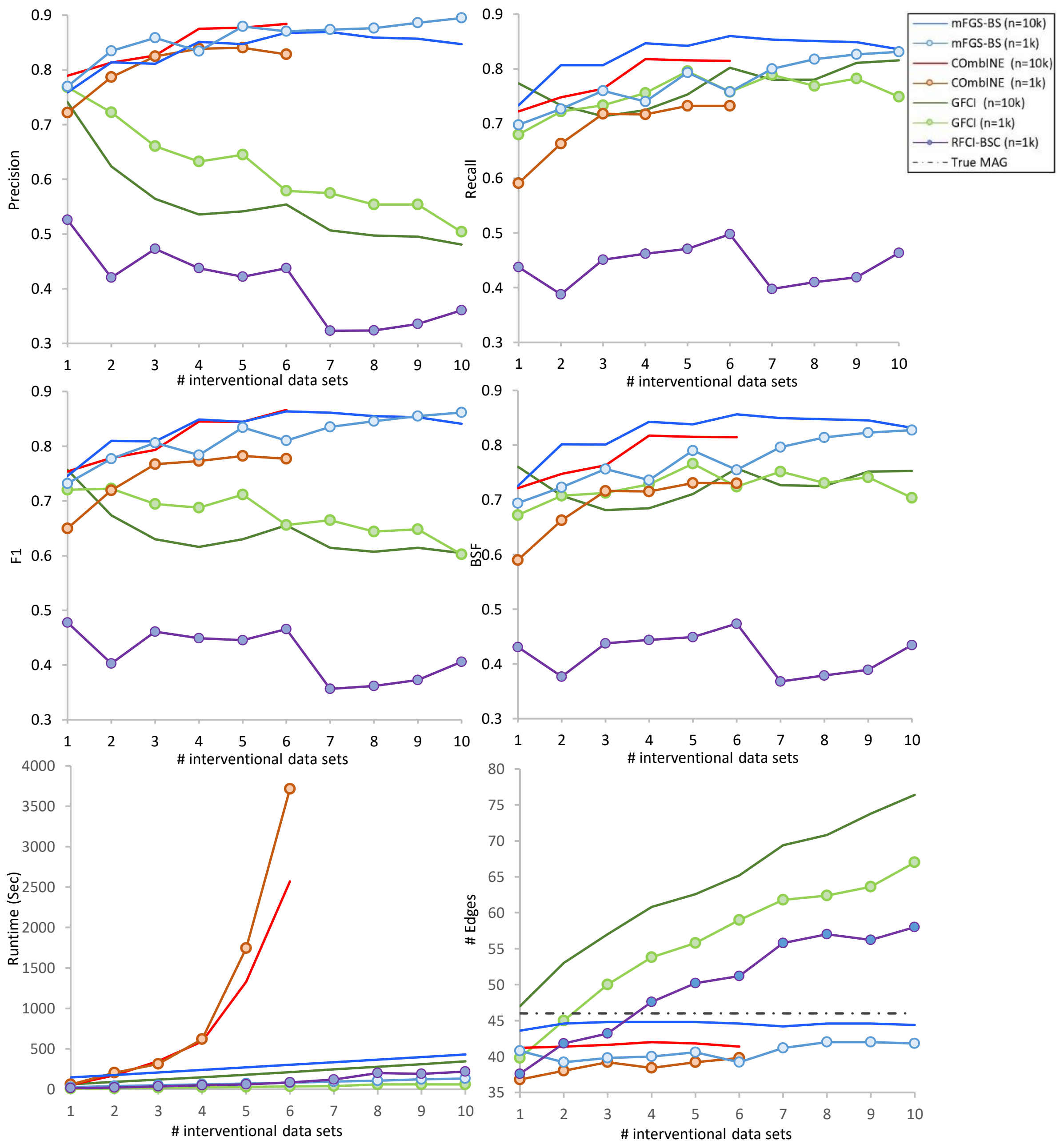


Figure 5.10 Average performance of the algorithms when applied to synthetic data generated from the Alarm network, assuming five intervened variables and 5% latent variables per dataset, over two sample sizes.

For the large and very large networks, COmbINE and RFCI-BSC fail to produce any results. On the other hand, both mFGS-BS and GFCI are able to generate results for all experiments across both sample sizes. The experimental results obtained from ForMed and Pathfinder case studies can be found in Figures A3 and A4 of Appendix A. Note that, in the case of these larger networks, five intervened variables represent a relatively low number. Still, as shown in Table 5.8, mFGS-BS performs considerably better than GFCI and RFCI-BSC across almost all experiments. The only case in which GFCI performs slightly better than mFGS-BS is for ForMed at 1k sample size, where GFCI averages scores of 0.58 and 0.59 for BSF and Recall respectively, whereas mFGS-BS averages scores of 0.57 for both metrics. On the other hand, the cases in which mFGS-BS outperforms GFCI involve much higher discrepancies in scores. For example, the most extreme case involves the Pathfinder case study where mFGS-BS averages a Precision score of 0.52 at 10k sample size, whereas GFCI averages a score of just 0.14.

Algorithms	**n**	**Property**	**Alarm**	**ForMed**	**Pathfinder**	**Property**	**Alarm**	**ForMed**	**Pathfinder**
		Precision				**Recall**			
mFGS-BS	1k	**0.88**	**0.86**	**0.83**	**0.31**	**0.62**	**0.78**	0.57	**0.17**
	10k	0.78	0.82	**0.80**	**0.52**	0.72	**0.79**	**0.69**	**0.35**
COmbINE	1k	0.46	0.81	T	T	0.50	0.69	T	T
	10k	**0.85**	**0.84**	T	T	**0.77**	0.78	T	T
GFCI	1k	0.65	0.62	0.74	0.17	0.50	0.75	**0.59**	0.13
	10k	0.59	0.55	0.69	0.14	0.59	0.77	0.67	0.13
RFCI-BSC	1k	0.55	0.41	T	T	0.27	0.44	T	T
		F1				**BSF**			
mFGS-BS	1k	**0.74**	**0.81**	**0.68**	**0.22**	**0.62**	**0.77**	0.57	**0.15**
	10k	0.75	**0.81**	**0.74**	**0.42**	0.71	**0.79**	**0.69**	**0.34**
COmbINE	1k	0.48	0.74	T	T	0.47	0.69	T	T
	10k	**0.81**	**0.81**	T	T	**0.77**	0.78	T	T
GFCI	1k	0.57	0.68	0.65	0.14	0.50	0.72	**0.58**	0.10
	10k	0.59	0.64	0.68	0.13	0.57	0.73	0.66	0.10
RFCI-BSC	1k	0.36	0.42	T	T	0.27	0.42	T	T
		Learnt Edges				**Runtime**			
mFGS-BS	1k	22.22	40.66	95.54	126.96	37.69	77.98	413.96	762.02
	10k	29.32	43.44	121.42	152.32	195.36	417.33	2,411.11	2,737.49
COmbINE	1k	35.30	38.57	T	T	5,420.72	1,108.76	T	T
	10k	29.04	41.57	T	T	368.40	845.34	T	T
GFCI	1k	24.70	55.82	111.40	171.36	**12.91**	**34.80**	**55.19**	**66.45**
	10k	32.02	63.60	136.80	220.00	**160.72**	**199.41**	**478.82**	**546.57**
RFCI-BSC	1k	15.88	49.86	T	T	M	99.8	T	T

Table 5.8 Average performance across all experiments in which five variables are targeted for intervention per dataset, where M indicates out-of-memory error, and T indicates failure to complete learning within the four-hour runtime limit. The best performance values are shown in bold.

The main conclusions from the results are:

- mFGS-BS is found to be sensitive to the ordering of interventional datasets. However, the sensitivity is relatively small in terms of graphical accuracy, and decreases with the number and the size of interventional datasets.

- Employing all three factors to determine edge direction produces the most accurate graphs (refer to subsection 5.4.2). Factor 1, which determines the probability of directed edges given the output of FGS, and Factor 2 which determines the probability of directed edges based on the ratio of Sepsets determining v-structure are found to have a stronger impact (in terms of increasing the F1 and BSF scores) than Factor 3 which relies on changes in objective score between observational and interventional data.

- mFGS-BS is found to be more accurate than the other algorithms when we simulate just one intervened variable. Specifically, mFGS-BS generates the highest F1 and BSF scores for the Asia, Sports and Alarm networks in most of the experiments (refer to Table 5.7). COmbINE and RFCI-BSC often fail to generate a result within the four-hour runtime limit when applied to the larger networks. The average BSF and F1 scores of mFGS-BS are approximately 45% and 38% higher compared to GFCI across all networks respectively, while the average BSF and F1 scores of COmbINE are 16% and 15% higher compared to GFCI over all experiments in which COmbINE generates a result respectively.

- The performance of both mFGS-BS and COmbINE continues to improve with the number of interventional datasets, while the performance of GFCI and RFCI-BSC does not. This highlights the advantage of algorithms that consider additional datasets independently. Moreover, the number of edges learnt by mFGS-BS tends to be lower compared to the number of edges present in the true MAGs, for the medium, large and very large networks. Note that while GFCI generates more edges when the number of interventional datasets increase, its overall performance in terms of BSF and F1 scores does not increase.

- The overall performance of mFGS-BS and COmbINE continues to improve with the number of variables targeted for intervention as expected, since the higher number of interventions can be viewed as providing additional causal information to the model. The average BSF scores increase by approximately 9% and 11% when considering five, instead of one, intervened variables per interventional data for the mFGS-BS and COmbINE algorithms respectively.

- The runtime of mFGS-BS, relative to the other three algorithms, appears to be worst in small and medium networks. However, the runtime of mFGS-BS, RFCI-BSC and GFCI scale linearly with the number of interventional datasets. In contrast, the empirical results suggest that COmbINE does not scale well with additional datasets. One explanation might be because COmbINE uses the MINISAT application to solve SAT instances encoded from results of CI tests, and the time to solve these SAT instances increases exponentially with the number of variables. A rather unexpected finding is that the computational time of COmbINE is higher when the sample size is 1k compared to 10k. This might be because the results of CI tests learnt from low sample sizes contain more conflicts compared to those obtained when the sample size of the input data is higher. Lastly, GFCI is found to be the fastest algorithm in almost all of the experiments, as expected, since it does not consider each input dataset independently.

5.5 Conclusions

This chapter describes the hybrid mFGS-BS algorithm which produces a PAG by learning the probabilities of each directed edge from one observational dataset and one or more interventional datasets in a causally insufficient setting. The posterior probabilities learnt from one dataset are considered as candidate objective priors for learning from the next dataset. Three other mechanisms contribute to the objective priors used with each dataset: colliders identified from the observational data; the CPDAGs produced by running the FGS algorithm on each dataset; and a score-based approach relating to intervention targets. Pairs of nodes which have a directed edge in each direction with a probability above a given threshold are treated as having a bidirected edge between them, so that the algorithm produces a PAG.

The results of mFGS-BS were compared to those obtained by COmbINE, which also enables learning from multiple observational and interventional datasets. We have also compared the results against the RFCI-BSC and GFCI algorithms with pooled data, which serves as the baseline performance not accounting for variables targeted for intervention. The empirical evaluation was based on six case studies of different complexity, with varying numbers of intervened variables, interventional datasets, and sample sizes. Overall, the results show that mFGS-BS considerably outperforms the baseline algorithms and outperforms COmbINE in most of the experiments, in terms of graphical accuracy. RFCI-BSC and GFCI consider a single dataset of pooled data rather than each input dataset independently. GFCI was the faster algorithm because it performs fewer CI tests by design, whereas RFCI-BSC tends to fail to produce a result when applied to larger networks and sample sizes. Lastly, mFGS-BS offers considerable improvements in learning efficiency compared to COmbINE, which failed to produce any results, within the four-hour runtime limit, for the larger networks.

A limitation of mFGS-BS is that it is sensitive to the ordering of the data sets and assumes equal sample size across all input data sets. This is, of course, an unrealistic assumption in practice. Future research directions could focus on adjusting mFGS-BS such that the local BDeu scores can be normalised to enable learning from multiple datasets with varying sample sizes, or by employing a resampling technique to generate multiple datasets with an equal sample size. Other future research directions could focus on enabling learning from interventional datasets that contain imperfect and uncertain interventions (refer to subsection 2.1.2), in addition to perfect interventions.

Chapter 6

Discovery and density estimation of latent confounders with evidence lower bound

Discovering and parameterising latent confounders represent important and challenging problems in causal structure learning and density estimation respectively. In this chapter, we focus on both discovering and learning the distribution of latent confounders. This task requires solutions that come from different areas of statistics and machine learning. We combine elements of Variational Bayesian methods, expectation-maximisation, hill-climbing search, and structure learning under the assumption of causal insufficiency. We propose two algorithms; Incremental Latent Confounder search with VBEM (ILC-V) that maximises model-selection accuracy, and Hill-Climbing Latent Confounder search with VBEM (HCLC-V) that improves computational efficiency in exchange for minor reductions in model-selection accuracy. The former algorithm maximises accuracy and is suitable for small networks, whereas the latter algorithm balances accuracy with computational complexity and so is suitable for moderate size networks.

This chapter is organised as follows: subsection 6.1 provides the preliminary information and related works relevant to this chapter, subsection 6.2 describes the two proposed algorithms, subsection 6.3 describes the evaluation setup, subsection 6.3 presents the results, and we provide conclusions in subsection 6.4.

6.1 Preliminaries

6.1.1 Conjugate-exponential family models

For density estimation of latent confounders, we consider conjugate-exponential family models for discrete data. We assume a Dirichlet prior that serves as a conjugate prior of a multinomial likelihood (Bishop, 2006), whose posterior distribution is also Dirichlet. We use the empirical Bayes method by Gelman et al. (2003) to determine the prior parameters from data and assume a Dirichlet prior $\mathrm{Dir}(\theta_i|a_{ij})$ where a_{ij} is a hyperparameter set to '1' for uniform distribution, and θ_i denotes parameters $\sum_j \theta_{ij} = 1$ where j represents the number of states. Since we perform structure learning and density estimation under causal insufficiency, some variables will not be observed in the data, leading to an incomplete-data marginal likelihood $p(D|G)$ of a DAG G.

6.1.2 Variational Bayesian Expectation-Maximization (VBEM)

Marginalising out the parameters over latent confounders L_i in $p(D|G)$ makes the task of learning prohibitively expensive and intractable. We address this issue by approximating distributions of latent variables using the computationally efficient Variational Bayesian Expectation-Maximization (VBEM) algorithm (Beal and Ghahramani, 2002) that enables tractable solutions. The VBEM algorithm combines elements of variational inference (Jordan et al., 1999) and Expectation-Maximisation (EM; Friedman, 1988). It uses an alternated optimisation technique to find a surrogate distribution $q(L, \theta)$ from any exponential family Q (e.g., Gaussian, Dirichlet, multinomial) and optimises towards the true distribution $p(L, \theta| D, G)$. VBEM offers an approximate solution that guarantees to monotonically increase the objective score, and scales better with large data compared to Markov Chain Monte Carlo (MCMC) (Hastings, 1970).

The objective of VBEM is to minimise the discrepancy between two distributions $q(L, \theta)$ and $p(L, \theta|D, G)$. It uses the reverse Kullback-Leiber (KL) divergence for this task, which is the standard choice for variational inference, defined as follows:

$$\begin{aligned}
\mathrm{KL}(q \parallel p) &= \iint dL d\theta q(L, \theta) \log \frac{q(L, \theta)}{p(L, \theta|D, G)} \\
&= \mathbb{E}_q \left[\log \frac{q(L, \theta)}{p(L, \theta|D, G)}\right] \\
&= \mathbb{E}_q[\log p(D|G)] - \left\{\mathbb{E}_q[\log p(L, \theta, D|G)] - \mathbb{E}_q[\log q(L, \theta)]\right\} \quad (6.1)
\end{aligned}$$

Because the incomplete-data marginal likelihood $p(D|G)$ is intractable to compute, we consider $p(D|G)$ to be a constant. The aim is to minimise $\mathrm{KL}(q \parallel p)$, which is equivalent to maximising the Evidence Lower Bound (ELBO). Therefore, we can minimise $\mathrm{KL}(q \parallel p)$ without having to know the true distribution $p(L, \theta|D, G)$ and $p(D|G)$. We can describe ELBO as the objective function:

$$\mathrm{ELBO} = \mathbb{E}_q[\log p(L, \theta, D|G)] - \mathbb{E}_q[\log q(L, \theta)] \quad (6.2)$$

where $q(L, \theta)$ is assumed to be the factorisation of the free distributions $q_L(L)$ and $q_\theta(\theta)$. We maximise ELBO using a function $\mathcal{F}$ of both $q_L(L)$ and $q_\theta(\theta)$ as follows (Beal and Ghahramani, 2006):

$$\begin{aligned}
\mathrm{ELBO} &= \mathcal{F}\big(q_L(L), q_\theta(\theta)\big) \\
&= \iint dL d\theta q_\theta(\theta) q_L(L)\big[\log p(L, \theta, D|G) - \log\big(q_L(L) q_\theta(\theta)\big)\big] \quad (6.3)
\end{aligned}$$

To maximise $\mathcal{F}$, VBEM calculates $q_L(L)$ and $q_\theta(\theta)$ while holding the other fixed at iteration t. The two steps for each iteration t are:

a) **VB-E step:** estimates the posterior distribution over latent confounders $q_L^{t+1}(L) = \prod_{i=1}^{|L|} q_{L_i}^{t+1}(L_i)$ given $q_\theta^t(\theta)$ from the last iteration by taking the functional derivatives in Equation (6.3) with respect to $q_{L_i}(L_i)$, where $|L|$ is the number of latent confounders.

b) **VB-M step:** estimates $q_{\theta}^{t+1}(\theta)$ given the posterior distribution $q_{L}{}^{t+1}(L)$ taken from the VB-E step by taking the functional derivatives in Equation (6.3) with respect to $q_{\theta}(\theta)$.

VBEM iterates over the VB-E and VB-M steps until the difference in ELBO becomes smaller than a given threshold, indicating convergence. Since ELBO is not a score-equivalent function, it generates different values for graphs that belong to the same Markov equivalence class. A revised version called p-ELBO was proposed by Rodriguez-Sanchez et al. (2020) that includes a penalty term to avoid the $|L_i|!$ equivalent ways of assigning sets of parameters that result in the same distribution (non-identifiability), and it is defined as p-ELBO = ELBO $-\sum_{i=1}^{|L|} \log |L_i|!$, where $|L_i|$ is the number of states in L_i.

6.1.3 Related works

ELBO was used as the objective function of a neural network in Variational Autoencoder (VAE) by Kingma and Welling (2013). VAE for heterogeneous Mixed type data (VAEM) was used by Ma et al. (2020) for density estimation of latent variables in deep generative models. VAE assumes each observed variable has a latent parent, whereas VAEM is an extension of VAE that assumes an additional latent confounder that serves as a parent of all latent variables.

The ELBO score was extended to p-ELBO by Rodriguez-Sanchez et al. (2020, 2022), which was used as the objective score in Constrained Incremental Learner (CIL) and Greedy Latent Structure Learner (GLSL) algorithms. CIL learns a tree-structured BN that assumes any two nodes are connected by one directed path only, whereas GLSL learns a DAG BN. Both algorithms start from an empty graph and perform various search operations including a) add or remove latent variables as parents of observed variables, b) increase the number of states of latent variables, and c) perform edge operations such as add, remove, or reverse edges, aiming to maximise p-ELBO. Searching for latent confounders often means iterating over all pairs of observed variables, which is computationally expensive. Instead, these algorithms offer a strategy that focuses on a set of pairs of variables that provide the highest MI. Empirical results show that GLSL outperforms CIL, but at the expense of high computational complexity.

6.2 The two proposed algorithms for learning of latent confounders

This subsection describes the two learning strategies we have implemented for latent confounder discovery and density estimation. Subsection 6.2.1 describes how we use existing algorithms to draw a PAG output, which in turn is given as an input to the two algorithms we propose, which in turn use the input PAG to search for different MAGs and DAGs with parameterised latent confounders. We describe the two algorithms in subsections 6.2.2 and 6.2.3 respectively. Both algorithms assume the input data are discrete. Consistent with previous studies that assume causal insufficiency, we assume that the latent confounders have no parents and that each latent confounder must have at least two children. We further assume a Dirichlet prior $q_{\theta}(\theta)$ over all parameters as described in subsection 6.1.1, and we use p-ELBO as the objective function which is computed using the VBEM algorithm as described in subsection 6.1.2.

6.2.1 Searching for MAGs and DAGs given a PAG input

The FCI algorithm and some of its variants discussed in subsections 2.2.1.3 and 2.2.3 represent the state-of-the-art in recovering ancestral graphs under the assumption of causal insufficiency. Any of these algorithms can be used to draw PAGs that can be given as input to the two proposed algorithms. A set of Markov equivalent MAGs can be then derived from that PAG. However, because the number of possible latent confounders that can be explored for a given MAG is generally intractable, we shall assume the minimum number of latent confounders satisfying the m-separation criteria.

Assumption 1: The optimal number of latent confounders is the minimum number of latent confounders that retain the CIs of a given MAG.

Figure 6.1 presents a simple PAG that contains two bidirected edges, along with a MAG and three DAGs that satisfy the CI statements of the PAG. Converting a MAG into possible DAGs implies that each DAG retains the CIs of that MAG by reducing the criteria of m-separation to d-separation. In this example, the DAG that contains the minimum number of latent confounders, with reference to the MAG in Figure 6.1b, is shown in Figure 6.1c. The DAGs in Figures 6.1d and 6.1e contain a higher number of latent confounders than the minimum required to satisfy all the CIs of the given MAG. Because the algorithms we describe in subsections 6.2.2 and 6.2.3 rely on **Assumption 1**, they will never explore DAGs that contain a higher number of latent confounders than the minimum required, and would not visit DAGs such as those shown in Figures 6.1d and 6.1e.

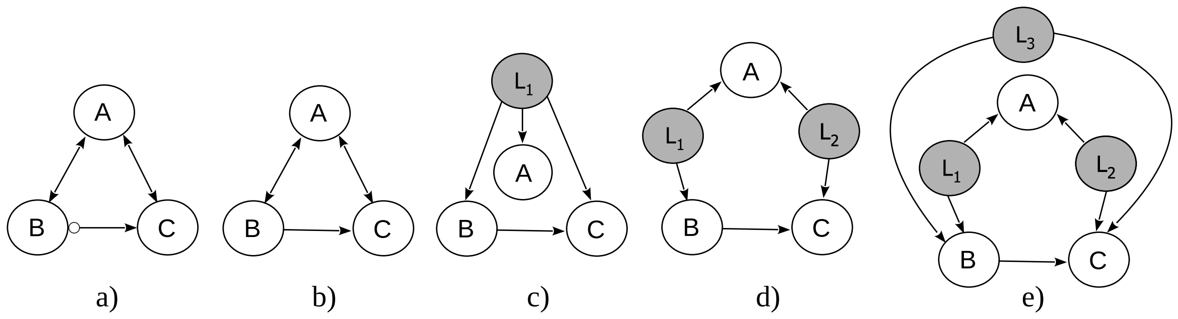


Figure 6.1 A PAG (a) along with one of its MAGs (b), and three DAGs (c, d, e) with different latent confounders (grey nodes) derived from the given MAG, where $A \not\perp B$, $A \not\perp C$ and $B \not\perp C$.

6.2.2 Incremental Latent Confounder search with VBEM (ILC-V)

The first algorithm, which we call ILC-V, is described in Algorithm 3. It takes a PAG input (Step 1) and uses the ZML algorithm available in R (Malinsky and Spirtes, 2017) to enumerate all Markov equivalent MAGs of that PAG (Step 3). It then constructs DAGs for each MAG, starting from the MAGs that contain the minimum number of bidirected edges (Step 4). Each latent confounder modelled at Step 4 is assumed to be binary, and the optimal DAG is the one that maximises p-ELBO using the VBEM algorithm made available as a Java library by Rodriguez-Sanchez (2021).

At Step 5, Algorithm 3 calls Algorithm 3b to determine the optimal number of states for each latent confounder. This is achieved by iterating over each latent confounder present in the highest scoring DAG determined at Step 4, and greedily increasing the number of states by one at a time, for each latent confounder. Algorithm 3b returns a DAG that contains the optimal number of states for each latent confounder, or the maximum number of states S if the objective score continues to increase with the number of states. To improve computational complexity, the objective score p-ELBO is applied to a subgraph G_S that contains the auxiliary latent confounders and their children, since the conditional distributions of the remaining nodes remain unchanged in the BN. The final Step 6 generates the final DAG BN and revises the p-ELBO score.

Algorithm 3: Incremental Latent Confounder search with VBEM (ILC-V)

```
Input:  A structure learning algorithm that generates PAG, max Sepset size k, significant threshold $\alpha$, observational data
        $D$, converge threshold c, max number of bidirected edges m, a runtime limit t.
Output: A DAG BN that contains latent confounders as observed variables, along with the conditional distributions.
Step 1  PAG ← Run a structure learning algorithm with $\alpha$ and k given $D$
Step 2  $S$ ← max number of states in $D$
        current number of bidirected edges ← count the total number of bidirected edges in PAG
        score_improve = TRUE
        best_pELBO = - Infinity
Step 3  List of MAGs $\mathcal{L}_M$ ← Enumerate all Markov equivalent MAGs from PAG
Step 4  While score_improve = TRUE or current number of bidirected edges ≤ m or elapsed time ≤ t
            best_local_pELBO = - Infinity
            For each MAG in $\mathcal{L}_M$ where its #bidirected edges = current number of bidirected edges
                Construct new DAG G that contains all edges → present in MAG and generate boolean
                    auxiliary latent confounders for edges ↔ present in MAG as per Assumption 1
                current_pELBO ← run VBEM until p-ELBO converges with c given $D$ and G
                If current_pELBO > best_pELBO
                    best_pELBO = current_pELBO
                    best_DAG = G
                If current_pELBO > best_local_pELBO
                    best_local_pELBO = current_pELBO
            current number of bidirected edges++
            If best_pELBO > best_local_pELBO
                score_improve = FALSE
Step 5  If $S$ > 2
            get best_DAG with (potentially) multinomial latent confounders ← run Algorithm 3b given best_DAG,
            $D$, c and S
Step 6  get best_pELBO and return Output ← run VBEM until p-ELBO converges with c given $D$ and best_DAG
```

Algorithm 3b: Greedy search for the optimal number of states for each latent confounder

```
Input:  A DAG G with auxiliary boolean latent confounders, max states S for each latent confounder, observational data
        $D$, converge threshold c.
Output: A DAG G with auxiliary (potentially) multinomial latent confounders.
Step 1  score_improve = TRUE
        best_pELBO = - Infinity
Step 2  For each latent confounder $i$ in DAG G
            While score_improve = TRUE or number of states ≤ S
                current_pELBO ← run VBEM until p-ELBO converges with c given D and subgraph $G_S$
                If current_pELBO > best_pELBO
                    best_pELBO = current_pELBO
                Else
                    score_improve = FALSE
                    number of states of latent confounder $i$ --
                number of states of latent confounder $i$ ++
                    Update the number of states of latent confounder $i$ in $G_S$ and G
Step 3  Return G with the optimal number of states for each latent confounder
```

6.2.3 Hill-Climbing Latent Confounder search with VBEM (HCLC-V)

Algorithm 4: Hill-Climbing Latent Confounder search with VBEM (HCLC-V)

```
Input:  A structure learning algorithm that generates PAG, max Sepset size k, significant threshold $\alpha$, observational data $D$,
        converge threshold c, max number of bidirected edges m, a runtime limit t.
Output: A DAG BN that contains latent confounders as observed variables, along with the conditional distributions.
  Step 1   Same as in Algorithm 3
  Step 2   Same as in Algorithm 3
  Step 3   List of best_latent_confounder $\mathcal{L}_L = \emptyset$
  Step 4   While score_improve = TRUE or current number of bidirected edges $\leq$ m or elapsed time $\leq$ t
               best_local_pELBO = - Infinity
               While all pairs A o—o B in PAG are not orientated
                   Construct new DAG G by changing all o→ present in PAG to → and generate boolean
                         auxiliary latent confounders for edges ↔ present in PAG as per Assumption 1
                   Orientate A → B or A ← B in G from all pairs A o—o B with the maximum p-ELBO using VBEM
               For each pair A o—o B or A o→ B in PAG which is not in $\mathcal{L}_L$
                   Construct new MAG that contains all edges → present in $G$ and add the edge A ↔ B and others
                      C ↔ D given $\mathcal{L}_L$
                      Construct new DAG G′ that contains all edges → present in MAG and generate
                      boolean auxiliary latent confounders for edges ↔ present in MAG as per Assumption 1
                   current_pELBO ← run VBEM until p-ELBO converges with c given D and G′
                   If current_pELBO > best_pELBO
                      best_pELBO = current_pELBO
                      best_DAG = G′
                      Add the auxiliary latent confounders to $\mathcal{L}_L$
                   If current_pELBO > best_local_pELBO
                      best_local_pELBO = current_pELBO

               current number of bidirected edges++
               If best_pELBO > best_local_pELBO
                   score_improve = FALSE
  Step 5   Same as in Algorithm 3
  Step 6   Same as in Algorithm 3
```

Because ILC-V (Algorithm 3) is computationally expensive, as we later show in subsection 6.4, one might be interested in a computationally efficient version that minimally decreases the objective score of Algorithm 3. A problem with ILC-V is that when the input PAG contains a high number of invariant edges o—o or o→, enumerating all possible MAGs can quickly cause memory allocation problems. To address this, we introduce a modified version of ILC-V, which we call HCLC-V, that skips Markov equivalence checks. This means that HCLC-V no longer needs to check the CIs for each DAG visited, and this saves enormous computational time. Instead, HCLC-V iterates over possible edge orientations as described in Step 4 of Algorithm 4, and continues to follow the incremental search strategy of ILC-V in terms of the number of bidirected edges. Moreover, a list with the best-found latent confounders from one iteration is carried forward to the next iteration (see Steps 3 and 4 in Algorithm 4). Lastly, since HCLC-V relies on hill-climbing search, it stops exploration when a local maximum is reached.

6.3 Evaluation

The experimental setup involves four real-world BNs taken from the Bayesys repository (Constantinou et al., 2020), described in Table 6.1. We generated synthetic data of 1k and 10k samples using the *bnlearn* R package (Scutari, 2019). One dataset is created for each latent confounder listed in Table 6.1. This process was applied to both sample sizes, and led to a total of 22 datasets.

BN	Variables	Edges	Max in-degree	Free parameters	Potential latent confounders
Asia	8	8	2	18	Smoke
Sports	9	15	2	1,049	RDlevel
Property	27	31	3	3,056	propertyPurchaseValue, borrowing, otherPropertyExpenses
Alarm	37	46	4	509	INTUBATION, HYPOVOLEMIA, LVFAILURE, ERRCAUTER, PULMEMBOLUS, KINKEDTUBE

Table 6.1 The properties of the four real-world networks considered for evaluation.

We have used the constraint-based FCI and the hybrid GFCI algorithms described in subsections 2.2.1.3 and 2.2.3, to generate PAGs to be provided as input to ILC-V and HCLC-V. This produced four different result-combinations, which we refer to as ILC-V_{FCI}, HCLC-V_{FCI}, ILC-V_{GFCI} and HCLC-V_{GFCI} in subsection 6.4. The GFCI algorithm was tested using the Tetrad-based *rcausal* R package (Wongchokprasitti, 2019), and the FCI algorithm was tested using the *pcalg* R package (Kalisch et al., 2012). The ILC-V and HCLC-V implementations are available online at https://github.com/kiattikunc/ILC_and_HCLC_with_VBEM. Regarding the hyperparameters of FCI and GFCI, we set the G^2 significance threshold to default $\alpha = 0.05$ and the Sepset to $k = -1$ for unlimited size of conditioning set. For ILC-V and HCLC-V, we set the maximum number of bidirected edges to $m = 4$ to enable us to carry out experiments within reasonable runtimes, and the convergence threshold of VBEM to $c = 0.01$.

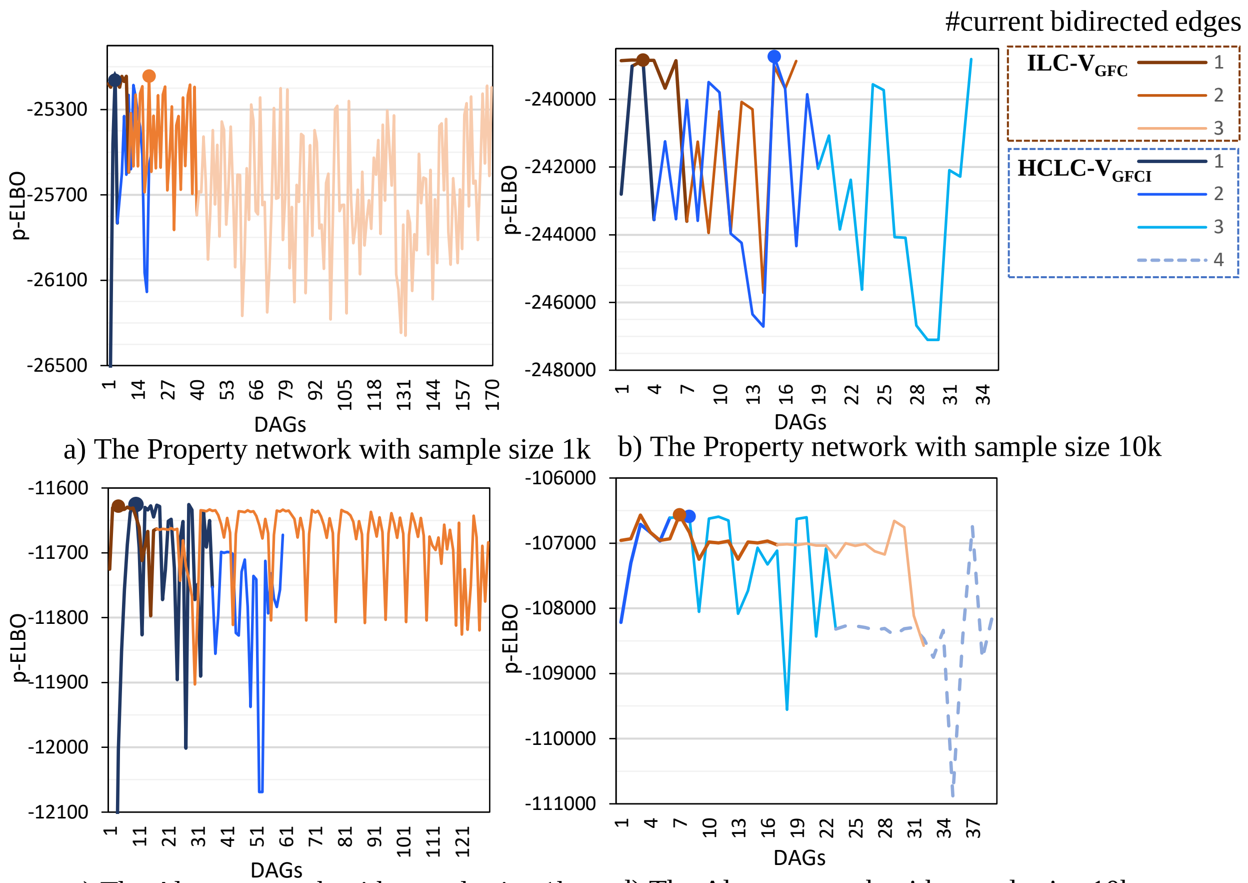
We assess the accuracy of ILC-V and HCLC-V in terms of the objective score p-ELBO and learning runtime, with reference to those obtained by the GLSL and CIL algorithms discussed in subsection 6.1.3. We consider the p-ELBO score by Rodriguez-Sanchez et al. (2020), which is an improved version of ELBO in tackling identifiability (Bishop, 2016) in discrete variables. Note that maximising ELBO can be viewed as being consistent with minimising KL-divergence between the true and surrogated distributions of latent confounders. However, as pointed out by Wallach et al. (2009), a possible issue with ELBO is that it may not accurately estimate the true distributions in latent variable models which, in turn, implies that p-ELBO is not perfect. We assume p-ELBO is a better alternative to traditional LL measures previously used to evaluate density estimation (Rodriguez-Sanchez et al., 2022), since LL is known to be prone to overfitting.

GLSL and CIL are tested using the Java library by Rodriguez-Sanchez (2021) with $mi = 10$ regarding the number of pairs of variables to be considered with the highest MI, and with maxNumberParents_latent $= -1$ for GLSL to assume no parents for density estimation of latent confounders to enable us to carry out experiments within reasonable runtimes. We impose a runtime limit of 12 hours to each experiment and set hyperparameter t to 12 hours for both ILC-V and HCLC-V, to ensure that they return a result within the 12-hour runtime limit. Experiments that do not complete learning within the specified runtime limit are denoted as “Timeout”. All experiments are based on 8GB of RAM. The experiments involving the Asia, Sports and Property networks were carried out on the Intel Core i5-8250 CPU at 1.80 GHz, whereas the experiments involving the Alarm network on the M1 CPU at 3.2 GHz.

6.4 Results

6.4.1 The difference in search space explored by ILC-V and HCLC-V

This subsection investigates the difference in search space explored between the two proposed algorithms, ILC-V and HCLC-V. The comparison assumes that the PAG inputs are produced by GFCI, and relies on Step 4 (which represents the main difference between the two algorithms) where the latent confounders are assumed to be binary.



a) The Property network with sample size 1k b) The Property network with sample size 10k

c) The Alarm network with sample size 1k d) The Alarm network with sample size 10k

Figure 6.2 The p-ELBO scores produced at Step 4 by the two algorithms, where • indicates the highest score achieved by the specified algorithm. The results in a) and b) are based on the Property network with variable ‘otherPropertyExpenses’ being the latent confounder and in c) and d) are based on the Alarm network with variable ‘INTUBATION’ being the latent confounder, and assume the input PAG is produced by GFCI.

Figure 6.2 presents the results based on the Property network (27 nodes) for both sample sizes 1k and 10k. Figure 6.2a shows that ILC-V$_{GFCI}$ produces a slightly higher p-ELBO score than HCLC-V$_{GFCI}$, but that ILC-V$_{GFCI}$ achieved that by exploring considerably more search space than HCLC-V$_{GFCI}$; i.e., visited a total of 170 DAGs versus 20 DAGs. The charts depict different colours to illustrate how the two algorithms differ at visiting DAGs derived from MAGs that contain increasing numbers of bidirected edges. Specifically, Figure 6.2a shows that ILC-V$_{GFCI}$ visited all DAGs derived from MAGs containing up to three bidirected edges, whereas HCLC-V$_{GFCI}$ ended at a local maximum while visiting DAGs derived from MAGs containing up to two bidirected edges.

Figure 6.2b, on the other hand, shows that the higher sample size helped ILC-V$_{GFCI}$ to both find a higher objective score and complete learning faster than HCLC-V$_{GFCI}$. This is because ILC-V$_{GFCI}$ found no DAG derived from MAGs containing two bidirected edges to have a higher score than the highest scoring DAG derived from MAGs containing one bidirected edge, which caused ILC-V$_{GFCI}$ to skip MAGs containing three bidirected edges. On the other hand, HCLC-V$_{GFCI}$ ended up visiting a higher number of DAGs, but note this does not necessarily imply that the algorithm was slower; i.e., recall that HCLC-V skips checking for Markov equivalence between graphs. Figure 6.2c and 6.2d repeat the analysis of Figure 6.2a and 6.2b with application to the Alarm network (37 nodes), and show that the results are consistent with those produced for the Property network. The only difference here is that, at 10k sample size, the p-ELBO score of HCLC-V$_{GFCI}$ matched that of the generally slower ILC-V$_{GFCI}$.

6.4.2 Performance of ILC-V and HCLC-V relative to other algorithms

We compare the results produced by ILC-V and HCLC-V to those produced by the CIL and GLSL algorithms described in subsection 6.1.3 which, to the best of our knowledge, are the two algorithms that are most relevant to this work, which involves both the discovery and density estimation of latent confounders.

BN (Latent confounder)	**True DAG**	**ILC-V$_{FCI}$**	**HCLC-V$_{FCI}$**	**ILC-V$_{GFCI}$**	**HCLC-V$_{GFCI}$**	**CIL**	**GLSL**
	p-ELBO (sample size 1k)						
Asia (smoke)	-2,258	-1,845	-1,845	-1,807	-1,807	-1,796	**-1,679**
Sports (Rdlevel)	-11,742	**-9,296**	-9,417	**-9,296**	-9,417	-10,228	-10,228
Property (propertyPurchaseValue)	-25,254	-34,496	-34,532	**-24,565**	-24,596	-29,040	-28,076
Property (borrowing)	-25,254	-35,042	-35,080	Memory	**-24,044**	-28,518	-27,534
Property (otherPropertyExpenses)	-25,254	-35,929	-35,979	**-24,079**	**-24,079**	-29,382	-28,363
Alarm (INTUBATION)	-11,220	Memory	-14,802	**-10,966**	-11,068	-13,777	-11,581
Alarm (HYPOVOLEMIA)	-11,220	Memory	-14,660	**-10,908**	-11,010	-13,721	-11,117
Alarm (LVFAILURE)	-11,220	Memory	-14,821	**-11,074**	-11,075	-13,989	-11,307
Alarm (ERRCAUTER)	-11,220	Memory	-14,678	-11,024	**-11,017**	-13,693	-11,254
Alarm (PULMEMBOLUS)	-11,220	Memory	-15,081	**-11,053**	-11,055	-13,994	-11,294
Alarm (KINKEDTUBE)	-11,220	Memory	-14,948	**-10,889**	-10,963	-13,896	-11,203
Average rank		5.1	5.0	1.8	1.9	3.8	2.9
	p-ELBO (sample size 10k)						
Asia (smoke)	-22,508	-17,860	-17,860	-17,601	-17,601	-17,039	**-16,135**
Sports (Rdlevel)	-108,800	**-92,014**	-92,864	**-92,014**	-92,864	-99,741	-99,741
Property (propertyPurchaseValue)	-235,622	-285,084	-285,084	**-238,090**	-238,267	-283,142	-275,212
Property (borrowing)	-235,622	-277,035	-277,035	**-239,289**	-239,520	-277,440	-269,719
Property (otherPropertyExpenses)	-235,622	-284,024	-284,038	-237,178	**-236,998**	-285,975	-277,949
Alarm (INTUBATION)	-105,739	-119,906	-119,845	**-104,919**	-105,096	-133,084	Timeout
Alarm (HYPOVOLEMIA)	-105,739	Memory	-126,194	**-101,997**	-102,960	-131,819	Timeout
Alarm (LVFAILURE)	-105,739	Memory	-129,574	-103,761	**-103,720**	-134,606	Timeout
Alarm (ERRCAUTER)	-105,739	Memory	-121,536	**-103,492**	-103,530	-132,280	Timeout
Alarm (PULMEMBOLUS)	-105,739	Memory	-126,811	-103,652	**-103,624**	-135,116	Timeout
Alarm (KINKEDTUBE)	-105,739	Memory	-125,698	-108,480	**-102,803**	-134,869	Timeout
Average rank		4.4	3.7	1.5	1.8	4.4	4.2

Table 6.2 The p-ELBO scores for each algorithm and dataset combination and across both sample sizes, where Memory indicates out-of-memory error in enumerating the possible MAGs, and Timeout indicates failure to complete learning within the 12-hour time limit. The best scores are indicated in bold.

Table 6.2 presents the p-ELBO score for each algorithm and dataset combination, plus the p-ELBO scores of the true DAGs, for both sample sizes 1k and 10k. Supplementary inference-based scores and runtimes can be found in Tables B1 and B2, in Appendix B. The average ranks show that ILC-V_{GFCI} performs best in terms of maximising the p-ELBO score across both sample sizes, followed by HCLC-V_{GFCI}. The CIL algorithm is found to be the worst performing algorithm at sample size 10k, whereas GLSL mostly outperforms both ILC-V_{FCI} and HCLC-V_{FCI}, but not ILC-V_{GFCI} and HCLC-V_{GFCI}. This means that ILC-V and HCLC-V benefit from the PAG input of GFCI, and suggests that the hybrid learning GFCI might be better than FCI at recovering PAGs; an observation consistent with previous studies (Constantinou et al., 2021). Note that while the true DAG will not always have the highest p-ELBO score, the highest scores produced by the algorithms tend to be very close to those of the true DAG, and this helps to validate the results.

While ILC performs best in general, it does not scale well with the size of the network. As shown in Table 6.2, ILC-V returns an out-of-memory error (for 8GB RAM) for most experiments with Alarm, specifically when paired with FCI, caused by the large number of possible MAGs derived from the input PAG. The cumulative runtime across all 10k datasets was 14, 34, 46 and 88 hours for CIL, HCLC-V_{GFCI}, ILC-V_{GFCI} and GLSL respectively, with a similar trend observed across 1k sample sizes. On overage, HCLC-V is found to be 1.4 times faster than ILC-V, which in turn is found to be 1.6 times slower than CIL and 4.5 times faster than GLSL which failed to complete the Alarm network experiments at 10k sample size; suggesting that its computational efficiency might not scale well with sample size.

6.5 Conclusions

This chapter investigates two novel algorithms that can be used for both discovery and density estimation of latent confounders in BN structure learning from discrete observational data. The first algorithm (ILC-V) aims to maximise model-selection accuracy by exploring sets of Markov equivalent MAGs, starting from the set of MAGs that contain the lowest number of bidirected edges and - while the objective score increases with each set - moving to sets of MAGs with increasing numbers of bidirected edges. The second algorithm (HCLC-V) aims to balance accuracy relative to computational efficiency by employing hill-climbing over the MAG space, enabling application to larger networks.

Both algorithms require a PAG to be provided as an input, which means that the proposed algorithms need to be paired with a structure learning algorithm that recovers ancestral graphs. Because the input PAG will typically indicate multiple possible latent confounders, the ILC-V and HCLC-V algorithms use p-ELBO as the objective function to determine the number as well as the position of the latent confounders, thereby contributing to the discovery process, in addition to density estimation, of latent confounders.

The two proposed algorithms are evaluated relative to two recent and relevant implementations that also optimise for p-ELBO. The empirical results show meaningful improvements in maximising the objective score, and in some ways in reducing time complexity; although the latter remains a major issue. Two important limitations are that a) both algorithms rely on a PAG input to be provided by some other structure learning algorithm, and b) the results are based on experiments that assume a single latent confounder only, which was necessary to ensure that most experiments complete within the 12-hour runtime limit.

Chapter 7

Tuning structure learning algorithms with out-of-sample and resampling strategies

One of the challenges practitioners face when applying structure learning algorithms to their data involves determining a set of hyperparameters; otherwise, a set of hyperparameter defaults is assumed. The optimal hyperparameter configuration often depends on multiple factors, including the size and density of the usually unknown underlying true graph, the sample size of the input data, and the structure learning algorithm. This chapter describes a novel hyperparameter tuning method, called the Out-of-sample Tuning for Structure Learning (OTSL), that employs out-of-sample and resampling strategies to estimate the optimal hyperparameter configuration for structure learning, given the input data set and structure learning algorithm.

This chapter is organised as follows: subsection 7.1 provides preliminary information regarding hyperparameter tuning for structure learning algorithms, subsection 7.2 describes the proposed algorithm, subsection 7.3 describes the evaluation setup, subsection 7.4 presents the results, and we provide conclusions in subsection 7.5.

7.1 Preliminaries

Subsections 2.2.1.1 and 2.2.2.1 provide background information on different functions that can be used to perform CI tests, and on different model-selection functions that serve as objective functions in score-based structure learning. In this chapter, we investigate solutions that could optimise the hyperparameter α that serves as the significance threshold in the functions that test for CI covered in subsection 2.2.1.1, and the hyperparameter iss that represents the equivalent sample size that determines the strength of prior beliefs in the $\mathrm{BDeu}_{\mathrm{iss}}$ score described in subsection 2.2.2.1. In addition to the standard functions used for CI tests and the BDeu score, we explore hyperparameter optimisation of the Extended BIC score described in subsection 7.7.1, and which represents a variant of the BIC score already described in subsection 2.2.2.1.

7.1.1 Extended BIC

Chen and Chen (2012) presented a modified version of BIC which they call Extended BIC (EBIC) that can be used to control the density of the learnt graph. This is achieved by introducing the hyperparameter $0 \leq \gamma$ that penalises the number of free parameters in the BN, which in turn are inversely proportional to the number of arcs in the learnt graph. This is equivalent to saying that large values of γ will favour sparser graphs. EBIC is defined as:

$$\mathrm{EBIC}_{\gamma}(G, D) = \mathrm{LL}(G, D) - \frac{\log(n)}{2} F - \gamma \log(V)\, F, \qquad 0 \leq \gamma$$

Foygel and Drton (2010) studied the impact of the hyperparameter $\gamma' \in [0,1]$ and found that $\gamma' = 0.5$ is best in most synthetic experiments. However, it is acknowledged that the optimal value of γ' is not invariant and hence, its optimisation remains an open question. In this work, we define $\mathrm{EBIC}_{\mathrm{normalised}\ \gamma}$ as:

$$\mathrm{EBIC}_{\mathrm{normalised}\ \gamma}(G, D) = \mathrm{LL}(G, D) - \frac{\log(n)}{2} F - \gamma' \log(V)\, F, \quad 0 \leq \gamma' \leq 1$$

where the hyperparameter $0 \leq \gamma$ is normalised to $\gamma' \in [0,1]$. Thus, γ is the hyperparameter of EBIC_{γ} and $\mathrm{EBIC}_{\mathrm{normalised}\ \gamma}$ where $\mathrm{EBIC}_{\gamma=0} = \mathrm{EBIC}_{\mathrm{normalised}\ \gamma=0} = \mathrm{BIC}$.

7.1.2 Related works

An issue with structure learning algorithms is that they come with a set of unoptimised hyperparameters. Because there is little guidance on how to choose these hyperparameters, most papers in the literature use these algorithms with either their hyperparameter defaults, or test them over a restricted set of different plausible hyperparameter values; a process that can be very time consuming. Hyperparameter tuning for structure learning algorithms can be divided into in-sample tuning and out-of-sample tuning methods, where the former utilises all available data and the latter uses a subset of the available data as a test data to tune hyperparameter configurations on data points that were not included in the training set.

In-sample tuning approaches include the Stability Approach to Regularization Selection (StARS) by Liu et al. (2010), which optimises for model stability by selecting the hyperparameter configuration that generates the most stable learnt graphs over perturbations of the input data. Out-of-sample tuning approaches include the Out-of-sample Causal Tuning (OCT) by Biza et al. (2020, 2022), which performs cross-validation to identify the Markov Blankets (MBs) for each variable. The MB of a variable A represents a set of variables that make A independent of all other variables, and can serve as a feature selection method. Specifically, the MB of A includes the parents of A, its children, and the parents of its children. The OCT algorithm uses MBs to obtain a Random Forest model and optimises hyperparameters for predictive accuracy over test data. Experimental results showed that it performed well against the in-sample StARS approach discussed above.

7.2 Out-of-sample Tuning for Structure Learning (OTSL)

This subsection describes the algorithm we propose for hyperparameter tuning, which we call Out-of-sample Tuning for Structure Learning (OTSL). OTSL determines the optimal hyperparameter configuration for a structure learning algorithm by performing out-of-sample resampling and optimisation on test data.

7.2.1 Resampling with replacement with multiple training and test datasets

Resampling with replacement or bootstrapping (Efron and Tibshirani, 1994) is commonly used for sampling in statistics and machine learning. Unlike traditional cross-validation where each fold is drawn from a dataset without replacement, bootstrapping involves resampling with replacement to produce new data for validation that may contain multiple instances of the original cases. Although cross-validation is a common optimisation technique for selecting a model based on its estimated predictive capability, the studies by McLatchie et al. (2023) and Piironen et al. (2016) empirically show that cross-validation led to the learning of complex models, particularly in the case with small datasets and a high number of variables. Consequently, resampling with replacement in structure learning was used to improve the accuracy of the learnt graph (Chun, 2011; Guo et al., 2022). We adopt this strategy for the OTSL algorithm and use resampling with replacement to generate multiple datasets for training and testing from a single observational dataset, where the training datasets are used for structure learning and the test datasets for hyperparameter tuning.

7.2.2 Tuning hyperparameters on test data

Subsection 2.2.2.1 describes the BIC and BDeu_{iss} scores, which are commonly used as objective functions in score-based structure learning algorithms. However, an issue with these model-selection scores is that the graph they score the highest tends not to be the ground truth graph. The model averaging MAHC that we describe in Chapter 3 demonstrates that output graphs with slightly lower average BIC score may improve the graphical accuracy of the learnt graph, especially in the presence of data noise which is inevitably present in real data. This model averaging approach motivates the design of the proposed tuning approach, especially in that it focuses on maximising model-selection by taking the average over multiple data splits.

We use the illustrations in Figure 7.1 to motivate our optimisation strategy, which is based on the HC algorithm and synthetic ALARM data with sample size 10k. Figure 7.1a presents the relationship between the graphical metric F1 (refer to subsection 7.4) and the objective score BDeu_{iss} when iss varies between 1 and 20. The tuning method involves resampling with replacement, where the input dataset of 10k is resampled 10 times and, at each iteration, split 9-to-1 for training and testing (refer to Algorithm 5). Specifically,

a) BDeu_{iss} is the tuning score optimised for different iss hyperparameters. Note that at each iteration of iss, the tuned score represents the average BDeu_{iss} score over 10 iterations of resampling (refer to Algorithms 5 and 5b).
b) F1 is the score for each graph recovered at different values of iss in BDeu_{iss}.

The illustration shows that it may be possible to optimise for iss in BDeu_{iss} such that it improves the F1 score. Specifically, Figure 7.1a shows that the optimal value for iss in BDeu_{iss} is 6, which in turn leads to a 0.57% improvement in F1 relative to the unoptimised hyperparameter default when $\text{iss} = 1$.

Figure 7.1b repeats the same exercise and assumes that the tuning score is $\text{EBIC}_{\text{normalised}\ \gamma}$, where γ in EBIC_{γ} varies between 0 and 19. In this example, we notice that the optimal γ hyperparameter is $\gamma = 3$ and happens to lead to the highest F1 score; an improvement of 11.63% relative to the unoptimised $\text{EBIC}_{\text{normalised}\ \gamma}$ score when $\gamma = 0$.

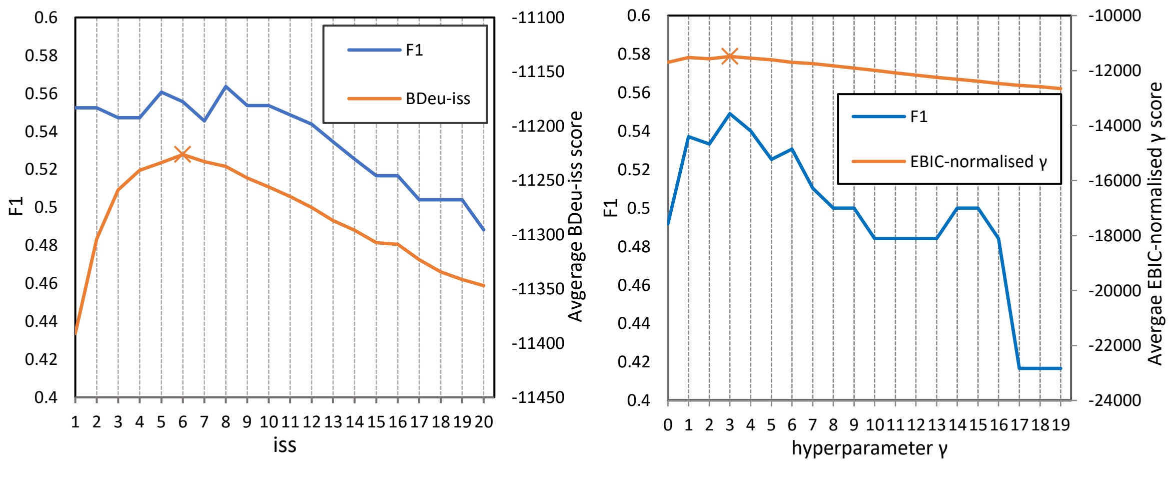


Figure 7.1 The F1 scores over different hyperparameter values for BDeu_{iss} and EBIC_{γ}. The illustration is based on the HC algorithm and synthetic ALARM data with a sample size 10k.

7.2.3 The Out-of-sample Tuning for Structure Learning (OTSL) algorithm

Algorithm 5 describes the OTSL algorithm. As described in Algorithm 5, OTSL takes as input a dataset D, the number of iterations K for resampling (we assume 10 as default), the tuning score (we explore BDeu_{iss} and $\text{EBIC}_{\text{normalised}\ \gamma}$ in this study) and a list of configurations C that specify the structure learning algorithm along with its hyperparameters and a range of those hyperparameters to be explored. OTSL starts by resampling K training and test datasets given the input data. It then applies the specified structure learning algorithm with configurations C to each training dataset in K, and optimises the hyperparameters of either BDeu_{iss} or $\text{EBIC}_{\text{normalised}\ \gamma}$ on each corresponding test dataset in K, across different input graphs and lists of configurations C. Each scoring function generates the scores to be tuned for each combination of input graph (i.e., learnt structure by a given algorithm), dataset, and set of configurations C. The optimal configuration is the one that generates the highest average tuning score over K training and test datasets, and is returned as the optimal configuration. This process is described in Algorithms 5, 5a and 5b, where Algorithms 5a and 5b describe the tuning process for $\text{EBIC}_{\text{normalised}\ \gamma}$ and BDeu_{iss} respectively.

Algorithm 5: **O**ut-of-sample **T**uning for **S**tructure **L**earning (OTSL)

```
Input: dataset D, a list of configurations C, iteration K, score for tuning
Output: $c'$
1:   The sample size of train data X = the number of instances of D × (K-1) /K
2:   The sample size of test data Y = the number of instances of D / K
3:   For k = 1 to K
4:       $D_{k,training}$ ← resample with replacement (D) with sample size X
5:       $D_{k,test}$ ← resample with replacement (D \ $D_{k,training}$) with sample size Y
6:   For c ∈ C // find the optimal configuration
7:       For k = 1 to K
8:           $G_{c,k}$ ← structure learning algorithm ($D_{k,training}$, c)
9:           If $G_{c,k}$ is CPDAG
10:              $G_{c,k}$ ← CPDAGtoDAG ($G_{c,k}$)
11:          If $G_{c,k}$ is PDAG
12:              $G_{c,k}$ ← PDAGtoDAG ($G_{c,k}$)
13:          $S_{c,k}$ ← score_for_tuning($G_{c,k}$, $D_{k,test}$, c) // scoring functions given test data and hyperparameters
14:      $S_c$ = average $S_{c,k}$ over K
15:  $c'$ = arg max $S_c$
16:  return $c'$
```

Algorithm 5a: **score_for_tuning (**$\text{EBIC}_{\text{normalised } \gamma}$)

```
Input: DAG G, dataset $\mathcal{D}$, configuration c from a list of configurations C
Output: $\text{EBIC}_{\text{normalised } \gamma}$
1:   If c contains γ
2:       score = $\text{EBIC}_{\text{normalised } \gamma}(G, \mathcal{D})$
3:   Else
4:       score = $\text{EBIC}_{\text{normalised } \gamma=0}(G, \mathcal{D})$
```

Algorithm 5b: **score_for_tuning (**BDeu_{iss})

```
Input: DAG G, dataset $\mathcal{D}$, configuration c from a list of configurations C
Output: $\text{BDeu}_{\text{iss}}$
1:   If c contains iss
2:       score = $\text{BDeu}_{\text{iss}}(G, \mathcal{D})$
3:   Else
4:       score = $\text{BDeu}_{\text{iss}=1}(G, \mathcal{D})$
```

7.3 Evaluation

We consider 10 real-world BNs whose properties are provided in Table 7.1. Six of them are taken from the bnlearn (Scutari, 2019) and Bayesys (Constantinou et al., 2020) repositories and are used to generate synthetic data with sample sizes of 1k and 10k. In addition to the six synthetically generated datasets, we also consider four real datasets which we discuss in more detail in subsection 7.4.2.

Synthetic data	Data source	Variables	Edges	Max in-degree	Free parameters
Asia	Bayesys (Constantinou et al., 2020)	8	8	2	18
Sports		9	15	2	1,049
Property		27	31	3	3,056
Alarm		37	46	4	509
Hailfinder	bnlearn (Scutari, 2019)	56	66	4	2,656
Hepar2		70	123	6	1,453
Real data	**Data source**	**Variables**	**Sample size**		
Diarrhoea	Bayesys (Constantinou et al., 2020)	28	259,627		
COVID-19		65	866		
ForMed		56	953		
Weather	NCEP (Kalnay et al., 1996)	648	900		

Table 7.1 The properties of the 10 case studies.

Structure learning algorithm	Configuration			
	CI test / Objective function	Hyperparameter		
		α	γ	iss
Constraint-based				
PC-Stable	Chi^2, MI, MI-sh	0.01, [0.05], 0.1	[0]	[1]
Score-based				
HC, FGS	BDeu$_{\text{iss}}$, EBIC$_{\gamma}$	-	[0], 1, 2, …, 19	[1], 2, 3, …, 20
Hybrid based				
MCMC	Chi^2/ BDeu$_{\text{iss}}$	[0.05]	-	[1], 2, 3, …, 20
MMHC	Chi^2/ BDeu$_{\text{iss}}$, EBIC$_{\gamma}$	0.01, [0.05]	[0], 1, 2, …, 9	[1], 2, 3, …, 10

Table 7.2 The algorithms tested for hyperparameter optimisation, along with the set of hyperparameters optimised. Brackets indicate the hyperparameter defaults. The size of the separation-set for CI tests is set to -1 to allow for an unlimited size of conditioning sets.

Table 7.2 lists the five structure learning algorithms considered for hyperparameter optimisation, spanning all three classes of structure learning. The description of these algorithms can be found in subsections 2.2.1.3, 2.2.2.2 and 2.2.3. Because OTSL is designed to optimise either EBIC$_{\text{normalised}\ \gamma}$ or BDeu$_{\text{iss}}$, we follow a somewhat different strategy when optimising constraint-based learning algorithms which do not involve score-based hyperparameters such as iss and γ. As shown in Table 7.2, the PC-Stable algorithm is tuned by exploring the three different thresholds for significance test α by maximising either EBIC or BDeu given their hyperparameter defaults; i.e., we iterate over hyperparameter values for α – not for iss or γ – when the input algorithm is constraint-based. Specifically, a) for constraint-based PC-Stable we optimise hyperparameter α which represents the statistical significance threshold for either Chi^2, MI, or MI-sh (refer to subsection 2.2.1.1), b) for score-based HC and FGS we optimise hyperparameter γ in EBIC$_{\gamma}$ and iss in BDeu$_{\text{iss}}$ (refer to subsections 7.1.1 and 2.2.2.1), and c) for hybrid algorithms MCMC and MMHC we optimise for all three possible hyperparameters. However, as shown in Table 7.2, we reduce the size of the set of possible hyperparameters to be explored for hybrid algorithms due to the much larger number of possible combinations of hyperparameters they produce. For example, if we were to explore the same range of hyperparameters for hybrid MMHC, then that would require 1,920 structure learning experiments for that algorithm alone; i.e., 3 hyperparameters for $\alpha \times 20$ for γ or 3 hyperparameters for $\alpha \times 20$ for iss for each case study and sample size. Because MCMC is the most computationally expensive algorithm amongst the structure learning algorithms considered, we restrict the value of hyperparameter α to its default 0.05 and instead optimise hyperparameter iss over a larger range in BDeu$_{\text{iss}}$.

We use the F1 and SHD graphical metrics to assess synthetic experiments as described in subsection 2.2.4. The scores reported in this study reflect comparisons between the learnt and the corresponding true DAGs. If a structure learning algorithm produces a CPDAG then a random DAG is generated from the learnt CPDAG.

The hyperparameter optimisation performance of OTSL is assessed with reference to other hyperparameter tuning methods that are proposed for tuning structure learning algorithms, and specifically the StARS and OCT approaches discussed in subsection 7.1.2. In addition, we also consider the BIC and the AIC model-selection functions as baselines for tuning, consistent with how they are used in other relevant studies for evaluation purposes, where tuning is determined by the hyperparameter value that maximises the given model-selection function (Biza et al., 2020).

We conduct all experiments by performing 10 iterations of resampling for both OTSL and StARS, and assuming a 10-fold cross-validation for OCT. We set a runtime limit of 24 hours for each experiment and yet, this was not enough to complete all experiments. Because most tuning experiments failed to complete learning on the real-world Diarrhoea and Weather datasets within the runtime limit, we modify the experimental setup for these two datasets. The issue with the Diarrhoea dataset is that it contains a large number of samples (259,627), which we address by modifying the resampling technique such that it creates 10 sets of training data restricted to a sample size of 9k, and 10 sets of test data restricted to a sample size of 1k, derived from the 259,627 instances of the Diarrhoea dataset. On the other hand, the issue with the Weather dataset is that it contains a large number of variables, and we address this by reducing the number of iterations for resampling to 5 for the Weather dataset.

Experiments with real data provide no access to ground truth. As a result, it is difficult to judge the unsupervised learning performance of these algorithms on real data. Therefore, we use real data to primarily investigate the issues we may face, specifically with large datasets as discussed above, and to illustrate how OTSL influences the structure learning performance of the different algorithms considered, in terms of model-selection, goodness-of-fit, and model dimensionality.

We test PC-Stable, HC and MMHC using the *bnlearn* R package (Scutari, 2019). FGS using Tetrad-based *rcausal* R package (Wongchokprasitti, 2019), and MCMC (the order-MCMC version) using the *BiDAG* R package (Suter et al., 2023). The model-selection scores of BIC and AIC, as well as the StARS and OCT tuning algorithms are tested using the MATLAB implementations available at https://github.com/mensxmachina/OCT. The implementation of OTSL is made available online at https://github.com/kiattikunc/OTSL. All experiments were conducted on a high performance computing cluster with 32 GBs of RAM, whereas the experiments involving the FGS algorithm were ran on a laptop with an M1 CPU at 3.2 GHz and 8GB of RAM.

7.4 Results

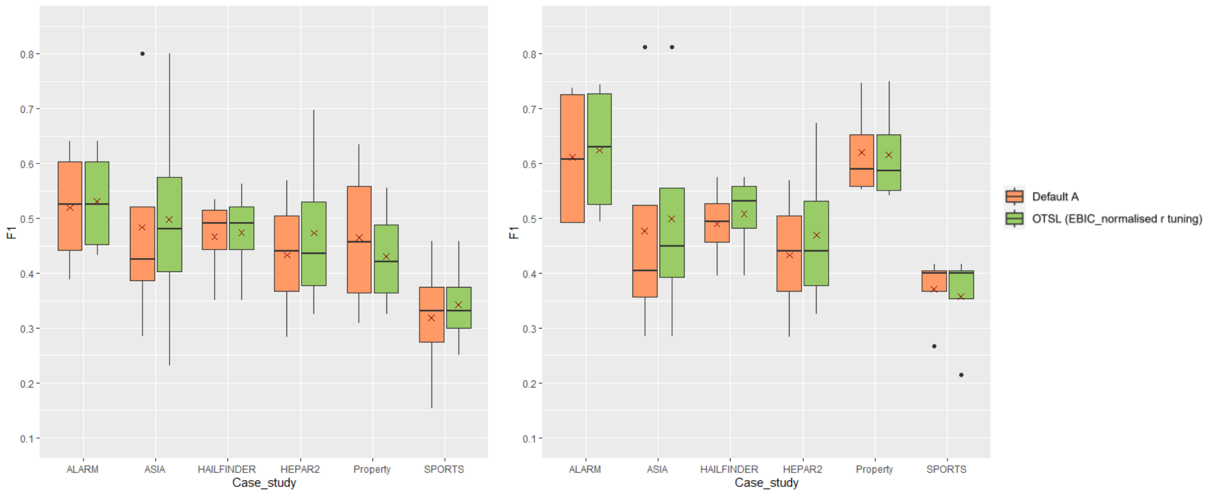
7.4.1 Results based on synthetic data

7.4.1.1 Impact of hyperparameter tuning on graphical structure

We assume two different cases for hyperparameter defaults: a) *Default A* where α = 0.05 for Chi^2 test and $\gamma = 0$ for EBIC$_\gamma$, and b) *Default B* where and α = 0.05 for Chi^2 test and iss = 1 for BDeu$_{\text{iss}}$. Figure 7.2 compares the F1 scores obtained by the four specified algorithms across all synthetic experiments, with and without (i.e., *Default A)* hyperparameter optimisation. In this set of experiments, hyperparameter optimisation is restricted to EBIC$_{\text{normalised }\gamma}$ and hence, the MCMC algorithm is not included in these results since EBIC$_\gamma$ is not available in the *BiDAG* R package. Figure 7.2a depicts the results when trained with datasets of sample size 1k, whereas Figure 7.2b depicts the results when trained with datasets of sample size 10k.

Across the 12 comparisons shown in both Figures 7.2a and 7.2b, the results show that the hyperparameter tuning applied by OTSL improves the average F1 scores in 9 cases, and slightly decreases performance in 3 cases; i.e., for Property at both 1k and 10k sample sizes and for Sports at 10k sample size. In Figure 7.2a, the average F1 score across all DAGs learnt over the six cases and four structure learning algorithms is 0.448 for default configurations, and increases to 0.458 (or by ~2.3%) when tuning the hyperparameters of EBIC$_{\text{normalised }\gamma}$. Figure 7.2b repeats these experiments for sample sizes 10k and shows that the results remain consistent with those obtained when the sample size is set to 1k. Specifically, the average F1 score across all DAGs is 0.5 for the default configurations, and increases to 0.513 (or by ~2.5%) when tuned with OTSL.

Figures 7.3a and 7.3b repeat the experiments of Figures 7.2a and 7.2b, and use BDeu$_{\text{iss}}$ as the tuning score instead of EBIC$_{\text{normalised }\gamma}$. In this case, however, the results show that the hyperparameter tuning applied by OTSL did not improve the average F1 scores. Specifically, the average F1 scores for the default configurations (*Default B*) are 0.51 and 0.56 for sample sizes 1k and 10k respectively, and 0.506 and 0.56 respectively when tuned with OTSL. According to the boxplots, this small difference could be explained by random variability.

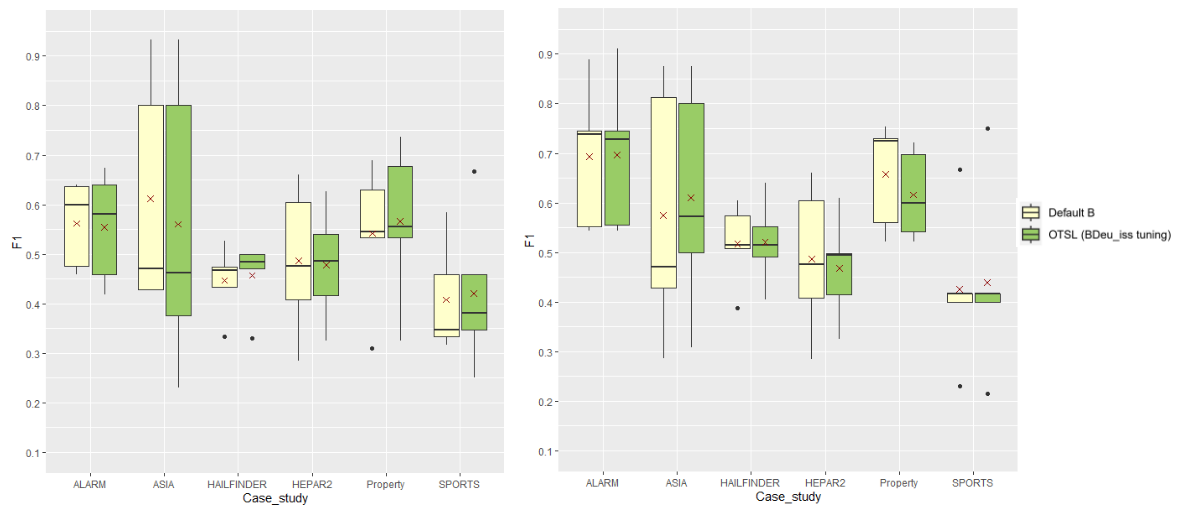


(a) (b)

Figure 7.2 The average F1 scores with and without hyperparameter tuning. Untuned algorithms assume *Default A* configuration and tuned algorithms assume OTSL with EBIC$_{\text{normalised }\gamma}$ as the tuning score. The average scores are derived over four structure learning algorithms (excluding MCMC that does not support EBIC$_\gamma$), and six synthetic case studies. The boxplots represent the highest and lowest F1 scores with outliers, × is the mean and – is the median. The lower edge of the boxplot represents the first quartile, while the higher edge of the boxplot represents the third quartile. Figure (a) depicts the scores for datasets with sample size 1k, and (b) with sample size 10k.

Table 7.3 details the average change in F1 and SHD scores for each structure learning algorithm relative to the hyperparameter defaults (*Default A)*, when we tune their hyperparameters with OTSL and $\text{EBIC}_{\text{normalised}\gamma}$ as the tuning score, as well as when we randomise the hyperparameter values averaged over 10 iterations. The results depicted in Table 7.3 show that randomising the hyperparameters leads to an average *decrease* of 1.71% in F1 score, and a decrease of 4.89% in SHD score, relative to the results obtained when assuming hyperparameter defaults. On the other hand, the F1 and SHD scores *increase* by 3.9% and 6.12% respectively when optimising the hyperparameters using OTSL. However, the constraint-based PC-Stable generates poor tuning performance with F1 and SHD scores decreasing by 1.81% and 1.08% respectively. This might suggest that the score-based tuning applied by OTSL to tune constraint-based CI tests might not be appropriate.

Table 7.4 repeats the experiments but assumes *Default B* configurations, and that the tuning score is BDeu_{iss} instead of $\text{EBIC}_{\text{normalised}\,\gamma}$ previously assumed in Table 7.3. In this case, the results show that both randomising and optimising the hyperparameter iss of BDeu_{iss} decreases graphical scores relative to those obtained by assuming hyperparameter defaults. In other words, it seems that assuming iss = 1 for BDeu_{iss} produces strong performance with little, if any, room for improvement via hyperparameter tuning, and this is consistent with what is reported by Steck (2008) and Uneo (2011) who recommend to set iss = 1, especially when the distributions of the variables are assumed to be skewed or when the true underlying structure is assumed to be sparse. Our results show that randomising the iss hyperparameter of BDeu_{iss} *decreases* the F1 and SHD scores by 4.86% and 11.02%, whereas optimising iss with OTSL *increases* the F1 scores by 0.12% and *decreases* the SHD scores by 3.36%. These results suggest that the BDeu_{iss} function may not be suitable for hyperparameter tuning, at least compared to $\text{EBIC}_{\text{normalised}\,\gamma}$, and that setting iss = 1 might indeed be sufficient, in general.



(a) (b)

Figure 7.3 The average F1 scores with and without hyperparameter tuning. Untuned algorithms assume *Default B* configuration and tuned algorithms assume OTSL with BDeu_{iss} as the tuning score. The average scores are derived over five structure learning algorithms, and six synthetic case studies. The boxplots represent the highest and lowest F1 scores with outliers, × is the mean and – is the median. The lower edge of the boxplot represents the first quartile, while the higher edge of the boxplot represents the third quartile. Figure (a) depicts the scores for datasets with sample size 1k, and (b) with sample size 10k.

Algorithm	Change in F1 relative to Default A		Change in SHD relative to Default A	
	Random configuration	Tuning with $\text{EBIC}_{\text{normalised}\ \gamma}$	Random configuration	Tuning with $\text{EBIC}_{\text{normalised}\ \gamma}$
PC-Stable	**0.95%**	-1.18%	-1.30%	-1.08%
HC	6.29%	**12.96%**	8.29%	**23.40%**
FGS	-14.21%	-0.10%	-26.31%	**1.54%**
MMHC	0.00%	**3.91%**	-0.43%	**0.61%**
Average	-1.71%	**3.90%**	-4.89%	**6.12%**

Table 7.3 The change in average F1 and SHD scores for each algorithm, after randomising their hyperparameters and after tuning them with OTSL. The experiments consider all six synthetic case studies and both 1k and 10k sample sizes. The hyperparameter defaults are α = 0.05 for Chi^2 test and $\gamma = 0$ for EBIC_γ (*Default A*). The best performance values are shown in bold.

Algorithm	Change in F1 relative to Default B		Change in SHD relative to Default B	
	Random configuration	Tuning with BDeu_{iss}	Random configuration	Tuning with BDeu_{iss}
PC-Stable	0.78%	**2.40%**	-2.00%	**0.64%**
HC	-10.48%	-8.64%	-18.95%	-11.33%
FGS	-14.77%	**2.25%**	-34.65%	-9.68%
MCMC	**2.79%**	2.51%	2.40%	**3.25%**
MMHC	-2.69%	**2.06%**	-2.11%	**0.31%**
Average	-4.86%	**0.12%**	-11.02%	-3.36%

Table 7.4 The change in average F1 and SHD scores for each algorithm, after randomising their hyperparameters and after tuning them with OTSL. The experiments consider all six synthetic case studies and both 1k and 10k sample sizes. The hyperparameter defaults are α = 0.05 for Chi^2 test and $\gamma = 0$ for BDeu_{iss} (*Default B*). The best performance values are shown in bold.

7.4.1.2 Assessing OTSL relative to existing tuning algorithms for structure learning

We compare the results of OTSL with those obtained by the out-of-sample tuning OCT and the in-sample tuning StARS. We also consider the baseline tuning results obtained by the model-selection scores BIC and AIC. This process involves applying the other four approaches to the same experiments presented in subsection 7.4.1.1, and comparing the changes to the F1 and SHD scores across all hyperparameter tuning approaches.

Structure learning algorithm	Hyperparameter tuning method									
	Out-of-sample		In-sample			Out-of-sample		In-sample		
	OTSL with $\text{EBIC}_{\text{normalised}\ \gamma}$ tuning	OCT	Model selection with BIC	Model selection with AIC	StARS	OTSL with $\text{EBIC}_{\text{normalised}\ \gamma}$ tuning	OCT	Model selection with BIC	Model selection with AIC	StARS
	Change of F1 relative to Default A					Change of SHD relative to Default A				
PC-Stable	-1.18%	**1.88%**	4.32%	3.30%	0.72%	-1.08%	**0.09%**	1.21%	1.56%	-0.25%
HC	**12.96%**	6.23%	-0.65%	-1.60%	-3.62%	**23.40%**	-35.43%	-8.76%	-9.90%	-34.77%
FGS	-0.10%	**1.26%**	-6.87%	-7.43%	1.55%	**1.54%**	-0.67%	-5.71%	-7.37%	-2.59%
MMHC	3.91%	3.03%	11.55%	**19.88%**	3.36%	0.61%	-4.89%	22.47%	**23.08%**	17.09%
Average	**3.90%**	3.10%	2.09%	3.54%	0.50%	**6.12%**	-10.22%	2.30%	1.84%	-5.13%

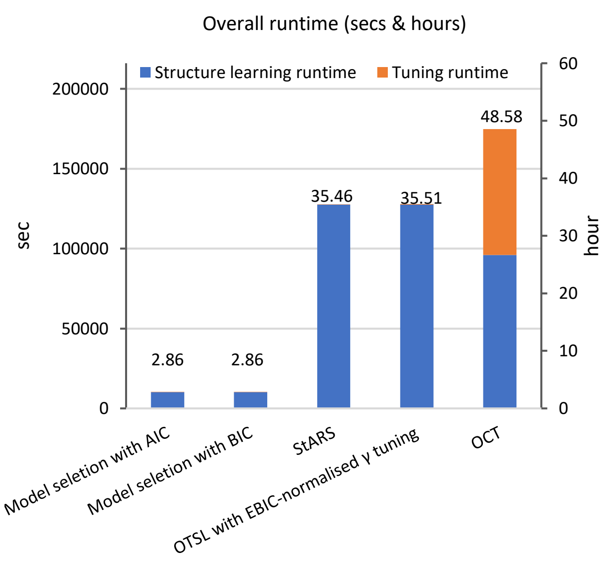
Table 7.5 The average change in F1 and SHD scores due to hyperparameter tuning by the specified tuning method. The averages are derived from all six synthetic case studies and over both sample sizes. The structure learning algorithms assume *Default A* hyperparameter configuration (Chi^2 test with α = 0.05, and EBIC_γ with $\gamma = 0$). The highest improvements in graphical accuracy are shown in bold.

Structure learning algorithm	Hyperparameter tuning method									
	Out-of-sample		In-sample			Out-of-sample		In-sample		
	OTSL with $\mathrm{BDeu_{iss}}$ tuning	OCT	Model selection with BIC	Model selection with AIC	StARS	OTSL with $\mathrm{BDeu_{iss}}$ tuning	OCT	Model selection with BIC	Model selection with AIC	StARS
	Change of F1 relative to Default B					Change of SHD relative to Default B				
PC-Stable	**2.40%**	1.26%	-6.87%	-7.43%	1.55%	**0.64%**	-0.67%	-5.71%	-7.37%	-2.59%
HC	-8.64%	-11.25%	-4.26%	**1.88%**	-11.21%	-11.33%	-40.02%	-2.57%	**-2.06%**	-11.15%
FGS	**2.25%**	-0.43%	-6.02%	-6.94%	-7.91%	**-9.68%**	-41.77%	-15.96%	-17.17%	-42.71%
MCMC	**2.51%**	-2.39%	-0.93%	0.18%	-0.15%	**3.25%**	-10.80%	-7.93%	1.97%	-2.29%
MMHC	**2.06%**	-2.47%	0.41%	-0.59%	-3.56%	**0.31%**	-1.65%	-0.60%	-0.28%	-2.09%
Average	**0.12%**	-3.06%	-3.53%	-2.58%	-4.26%	**-3.36%**	-18.98%	-6.55%	-4.98%	-12.17%

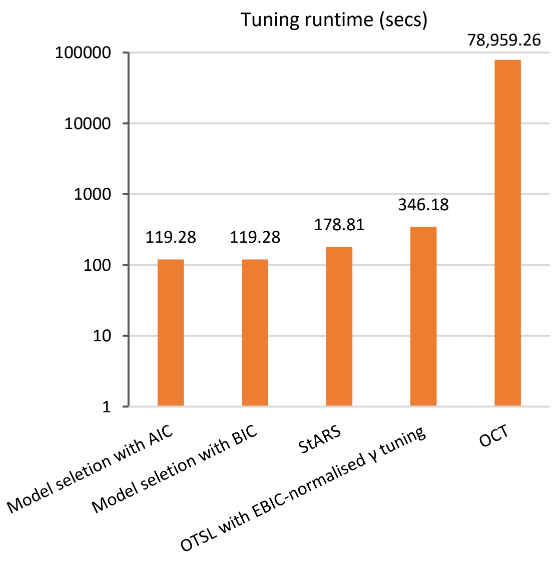
Table 7.6 The average change in F1 and SHD scores due to hyperparameter tuning by the specified tuning method. The averages are derived from all six synthetic case studies and over both sample sizes. The structure learning algorithms assume *Default B* hyperparameter configuration (Chi^2 test with α = 0.05, and $\mathrm{BDeu_{iss}}$ with iss = 1). The highest improvements in graphical accuracy are shown in bold.

Tables 7.5 and 7.6 summarise these results for both *Default A* and *Default B* hyperparameter configurations respectively. Table 7.5 shows that while none of the hyperparameter tuning approaches improves the graphical accuracy for all four structure learning algorithms, most of the approaches do improve the average structure learning performance across all algorithms. Specifically, all five tuning approaches improve the average F1 score across the four structure learning algorithms considered, although only three out of the five tuning approaches also improve the SHD score. The OTSL algorithm with $\mathrm{EBIC_{normalised\ \gamma}}$ tuning increases both the F1 (up by 3.9%) and SHD (up by 6.12%) scores the most across all the tuning approaches considered. Interestingly, the F1 and SHD scores provide contradictory conclusions about the impact on graphical structure for OCT and StARS algorithms, and this inconsistency between the F1 and SHD metrics is in agreement with other studies (Constantinou et al., 2021). For example, the F1 metric suggests that the hyperparameter tuning of OCT improves the structure learning performance of all four structure learning algorithms, whereas the SHD metric suggests that OCT decreases the graphical accuracy of three out of the four structure learning algorithms.

Table 7.6 presents the same results when the hyperparameter tuning approaches are applied to the iss hyperparameter of $\mathrm{BDeu_{iss}}$. Overall, the results are consistent with those presented in Tables 7.3 and 7.4, in that hyperparameter tuning appears to be successful for $\mathrm{EBIC_{normalised\ \gamma}}$ but not for $\mathrm{BDeu_{iss}}$. While tuning with $\mathrm{BDeu_{iss}}$ is found to be rather inadequate for all tuning methods, OTSL is found to perform considerably better compared to the other tuning approaches with an *increase* of 0.12% in the average F1 score (improved the scores of four out of the five algorithms) and a *decrease* of 3.36% in the average SHD score (improved the scores of three out of the five algorithms).



(a)



(b)

Figure 7.4 (a) Overall runtime (structure learning and tuning) and (b) tuning runtime, summed over all six synthetic datasets and two sample sizes, across all five structure learning algorithms.

We also assess the computational complexity of OTSL by comparing its hyperparameter tuning and overall structure learning runtimes against those produced by the other hyperparameter tuning approaches. Provisional results show that the runtimes are similar for both $\text{EBIC}_{\text{normalised}\gamma}$ and BDeu_{iss}, but here we focus on $\text{EBIC}_{\text{normalised }\gamma}$ which produces the best tuning performance. Figure 7.4a depicts the total runtimes (hyperparameter tuning and structure learning) across all six case studies, two sample sizes, and five structure learning algorithms, whereas Figure 7.4b shows the runtime for the same experiments but restricted to the hyperparameter tuning phase. As expected, optimisation with model-selection functions such as BIC and AIC results in very low runtimes, since they do not involve out-of-sample or resampling strategies, whereas OTSL, OCT and StARS perform 10 iterations of either in-sample or out-of-sample tuning for each hyperparameter configuration and this leads to considerably higher runtimes. Overall, the results in Figure 4a show that the computational runtime of OTSL is similar to that of StARS, and considerably faster than that of OCT. Importantly, the tuning runtimes of OTSL and StARS represent just 0.2% and 0.4% of the total structure learning runtime respectively, whereas the tuning runtime of OCT represents 43% of its total structure learning runtime. Figure 7.4b shows that the tuning runtime of OTSL is slower than the tuning runtime of StARS, but much faster than the tuning runtime of OCT.

7.4.2 Applying OTSL to real data

While previous subsections focused on evaluating OSTL in terms of how its tuning improves the recovery of the ground truth graphs that were used to generate synthetic data, this subsection illustrates how OTSL could be used in practice with application to four different real datasets that come from different disciplines. As discussed in subsection 7.3, real data do not come with an access to ground truth and hence, the purpose here is to illustrate how tuning influences the structure learning performance of the different algorithms considered when applied to real data. We consider the following four discrete datasets, where the first three are obtained from the Bayesys repository (Constantinou et al., 2020) and the fourth from the National Center for

Environmental Prediction (NCEP) and the National Center for Atmospheric Research (NCAR) in the USA, known as the NCEP/NCAR Reanalysis Project (Kalnay et al., 1996):

a) **ForMed**: A case study already covered in subsection 3.1, which involves assessing and managing the risk of violence in released prisoners with history of violence and mental health (Coid et al., 2016; Constantinou et al., 2015). The data was collected through interviews and assessments comprising risk factors for 953 individual cases, and contains a total of 56 categorical variables.
b) **COVID-19**: A dataset that captures pandemic data about the COVID-19 outbreak in the UK (Constantinou et al., 2023). The data comprises of 18 variables that capture information related to viral tests, infections, hospitalisations, vaccinations, deaths, COVID-19 variants, population mobility such as usage of transportation, schools, and restaurants, as well as various government policies such as facemasks and lockdowns. The data instances represent daily information, spanning from January 30th, 2020, to June 13th, 2022, resulting in a total of 866 instances.
c) **Diarrhoea**: Pre-processed survey data collated from the Demographic and Health Survey (DHS) program, which was used to investigate the factors associated with childhood diarrhoea in India (Kitson and Constantinou, 2021). The dataset captures relevant cases from 2015 to 2016 and contains 28 variables and 259,627 instances.
d) **Weather**: A dataset that captures the monthly means of air temperature and other climatological data for each location as measured by latitude (y coordinate) and longitude (x coordinate) over the global grid system (Kalnay et al., 1996). The dataset merges information obtained from multiple sources, i.e., balloons, satellites, and buoys. It provides a comprehensive 75-year record from 1948 to 2022 of global atmospheric field analyses. We used the *bnlearn* R package (2019) to discretise the dataset. Because the raw data is too big for our experiments, we also resized the spatial dataset from 2.5 degree x 2.5 degree global grids to 10 degree x 10 degree global grids, and reduced the total number of variables from 10,512 (144x73) to 648 (36x18). Therefore, the dataset used in this study contains a total of 648 variables and 900 instances.

We apply the structure learning algorithms to each of the four datasets, and tune their hyperparameters using OTSL. We only consider *Default A* hyperparameter configuration with $\mathrm{EBIC}_{\mathrm{normalised}\,\gamma}$ for tuning, which was shown to be more suitable for hyperparameter optimisation. Table 7.7 presents the results obtained by applying the specified structure learning algorithms to the ForMed dataset and tuning their hyperparameters with OTSL. We report the model-selection score BIC, the goodness-of-fit score LL, the number of free parameters as a measure of model dimensionality, and the tuning scores $\mathrm{EBIC}_{\mathrm{normalised}\,\gamma}$. Table 7.7 shows that out of the four structure learning algorithms considered, only one (HC with $\gamma = 2$) had its hyperparameter changed following tuning with OTSL. The tuning scores $\mathrm{EBIC}_{\mathrm{normalised}\,\gamma}$ in Table 7.7 suggest that the graph produced by MMHC, presented in Figure 7.5, might be the 'best' structure to consider amongst those learnt by the different algorithms, although this suggestion contradicts the BIC score which suggests that the best structure may be the one learnt by HC.

Tables 7.8, 7.9, and 7.10, and corresponding Figures 7.6, 7.7, and 7.8, repeat the above analyses for case studies COVID-19, Diarrhoea and Weather respectively. Note that only two algorithms are reported for the Weather case study, and this is because HC did not complete learning within the 24-hour time limit, while FGS returned a memory allocation error. The results show that OTSL modified the hyperparameters of three and four, out of the four, structure learning algorithms in COVID-19 and Diarrhoea cases respectively, and for one out of the two algorithms for dataset Weather. Table 7.8 shows that FGS produces the best structure for the COVID-19 case study (see Figure 7.6) according to both $\text{EBIC}_{\text{normalised }\gamma}$ and BIC scores. On the other hand, the results in Table 7.9 suggest that the graph produced by FGS is the best structure according to $\text{EBIC}_{\text{normalised }\gamma}$, which once more contradicts the BIC score that scores the graph produced by HC the highest. Lastly, in Table 7.10 both $\text{EBIC}_{\text{normalised }\gamma}$ and BIC are in agreement that MMHC produced the best structure shown in Figure 7.8. The nodes in Figure 7.8 represent random variables of a monthly temperature for each location, whereas the arcs represent the spatial dependencies of surface temperatures for each grid[2].

Structure learning algorithm	CI test / Objective score	Optimal hyperparameter from OTSL (Tuning with $\text{EBIC}_{\text{normalised }\gamma}$)	Tuning score $\text{EBIC}_{\text{normalised }\gamma}$ from OTSL	Score of learnt graph		
				BIC	LL	Free parameters
PC-Stable	Chi^2	$\alpha = 0.05$	-4,099	-40,791	-39,775	296
	MI	$\alpha = 0.05$	-4,113	-40,799	-39,760	303
	MI-sh	$\alpha = 0.05$	-4,092	-40,837	-39,774	310
HC	EBIC_γ	$\gamma = 2$	-3,974	**-37,062**	-35,442	540
FGS	EBIC_γ	$\gamma = 0$	-4,195	-42,343	-41,846	145
MMHC	**Chi^2 / EBIC_γ**	**$\alpha = 0.05$ / $\gamma = 0$**	**-3,942**	-38,183	-37,744	439

Table 7.7 The tuning, model-selection, goodness-of-fit, and dimensionality scores of the graphs learnt by the specified structure learning algorithms when applied to the ForMed dataset, with OTSL tuning. The best performance values are shown in bold.

[2] The orange arcs represent short-distance temperature dependencies, while the red arcs show the teleconnected dependencies. We observe that the local short-distance arcs are dense, representing atmospheric thermodynamic processes, while the teleconnected dependencies are represented by only three arcs. One of these teleconnected dependencies indicates the El Niño effects, which are caused by temperatures along the equator in the Pacific Ocean (Yamasaki et al., 2008).

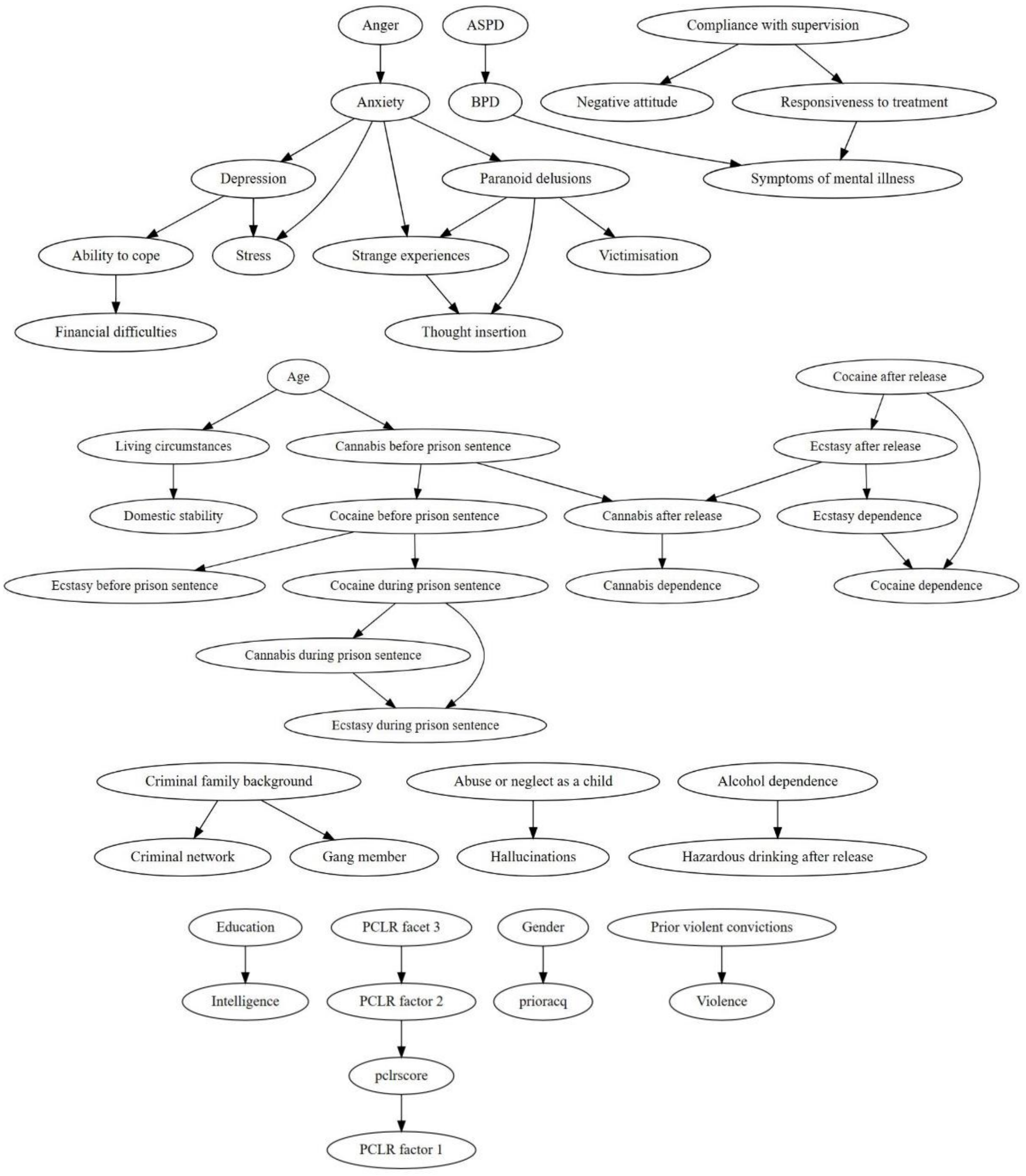


Figure 7.5 The DAG learnt by MMHC for the ForMed dataset with OTSL tuning (Table 7.7).

Structure learning algorithm	CI test / Objective score	Optimal hyperparameter from OTSL (Tuning with $\text{EBIC}_{\text{normalised } \gamma}$)	Tuning score $\text{EBIC}_{\text{normalised } \gamma}$ from OTSL	Score of learnt graph		
				BIC	LL	Free parameters
PC-Stable	Chi^2	$\alpha = 0.1$	-1,392	-13,666	-13,270	117
	MI	$\alpha = 0.1$	-1,395	-13,768	-13,190	171
	MI-sh	$\alpha = 0.01$	-1,395	-13,768	-13,190	171
HC	EBIC_γ	$\gamma = 5$	-1,249	-10,725	-8,787	323
FGS	$\mathbf{EBIC_\gamma}$	$\boldsymbol{\gamma = 1}$	**-1,092**	**-9,038**	-9,918	260
MMHC	Chi^2 /EBIC_γ	$\alpha = 0.05$ / $\gamma = 0$	-1,257	-12,267	-12,129	138

Table 7.8 The tuning, model-selection, goodness-of-fit, and dimensionality scores of the graphs learnt by the specified structure learning algorithms when applied to the COVID-19 dataset, with OTSL tuning. The best performance values are shown in bold.

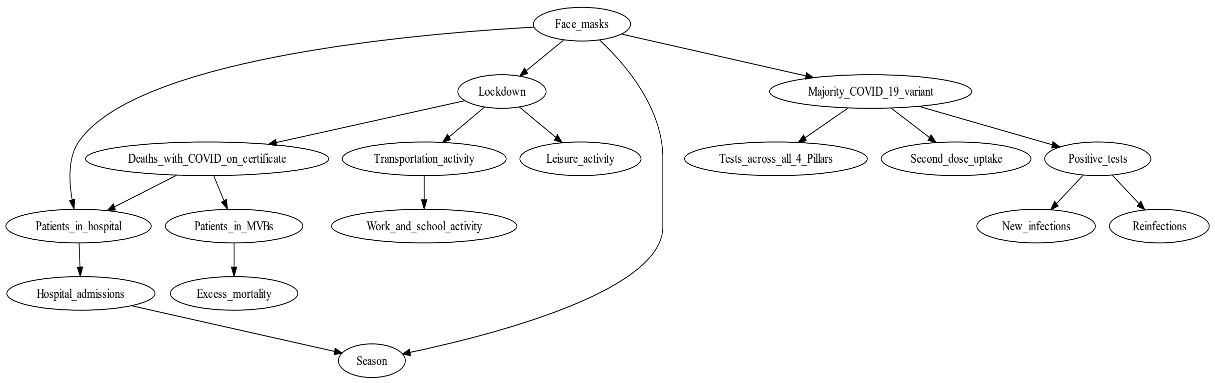


Figure 7.6 The DAG (sampled from the learnt CPDAG) learnt by FGS for the COVID-19 dataset with OTSL tuning (Table 7.8).

Structure learning algorithm	CI test / Objective score	Optimal hyperparameter from OTSL (Tuning with $\text{EBIC}_{\text{normalised } \gamma}$)	Tuning score $\text{EBIC}_{\text{normalised } \gamma}$ from OTSL	Score of learnt graph		
				BIC	LL	Free parameters
PC-Stable	Chi^2	$\alpha = 0.01$	-19,653	-4,910,813	-4,899,630	1,794
	MI	$\alpha = 0.1$	-19,400	-5,099,129	-5,088,588	1,691
	MI-sh	$\alpha = 0.01$	-19,506	-5,082,642	-5,068,485	1,691
HC	EBIC_{γ}	$\gamma = 2$	-19,257	**-4,776,526**	-4,748,359	9,389
FGS	$\mathbf{EBIC_{\gamma}}$	$\boldsymbol{\gamma = 0}$	**-19,175**	-4,944,463	-4,941,340	501
MMHC	Chi^2 / EBIC_{γ}	$\alpha = 0.01$ / $\gamma = 0$	-19,257	-4,979,854	-4,979,334	520

Table 7.9 The tuning, model-selection, goodness-of-fit, and dimensionality scores of the graphs learnt by the specified structure learning algorithms when applied to the Diarrhoea dataset, with OTSL tuning. The best performance values are shown in bold.

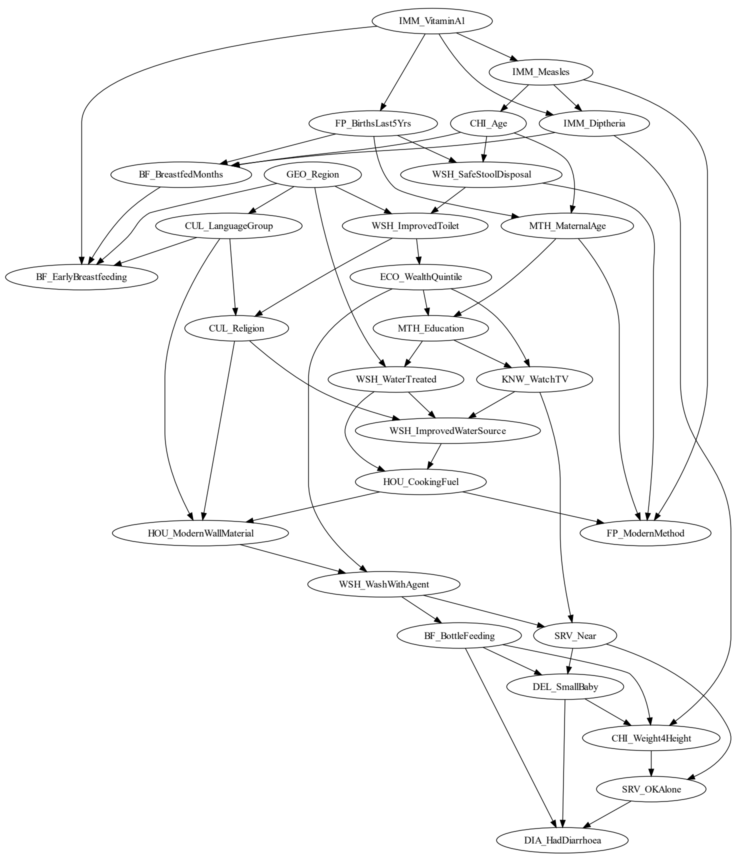


Figure 7.7 The DAG (sampled from the learnt CPDAG) learnt by FGS for the Diarrhoea dataset with OTSL tuning (Table 7.9).

Structure learning algorithm	CI test / Objective score	Optimal hyperparameter from OTSL (Tuning with $\text{EBIC}_{\text{normalised }\gamma}$)	Tuning score $\text{EBIC}_{\text{normalised }\gamma}$ from OTSL	Score of learnt graph		
				BIC	LL	Free parameters
PC-Stable	Chi^2	$\alpha = 0.05$	-67,594	-334,171	-319,219	4,396
	MI	$\alpha = 0.01$	-72,525	-378,849	-366,591	3,604
	MI-sh	$\alpha = 0.1$	-71,653	-346,160	-333,059	3,852
MMHC	**Chi^2 / EBIC$_\gamma$**	$\boldsymbol{\alpha = 0.05, \gamma = 0}$	**-66,350**	**-318,220**	**-314,724**	3,496

Table 7.10 The tuning, model-selection, goodness-of-fit, and dimensionality scores of the graphs learnt by the specified structure learning algorithms when applied to the Weather dataset, with OTSL tuning. The best performance values are shown in bold.

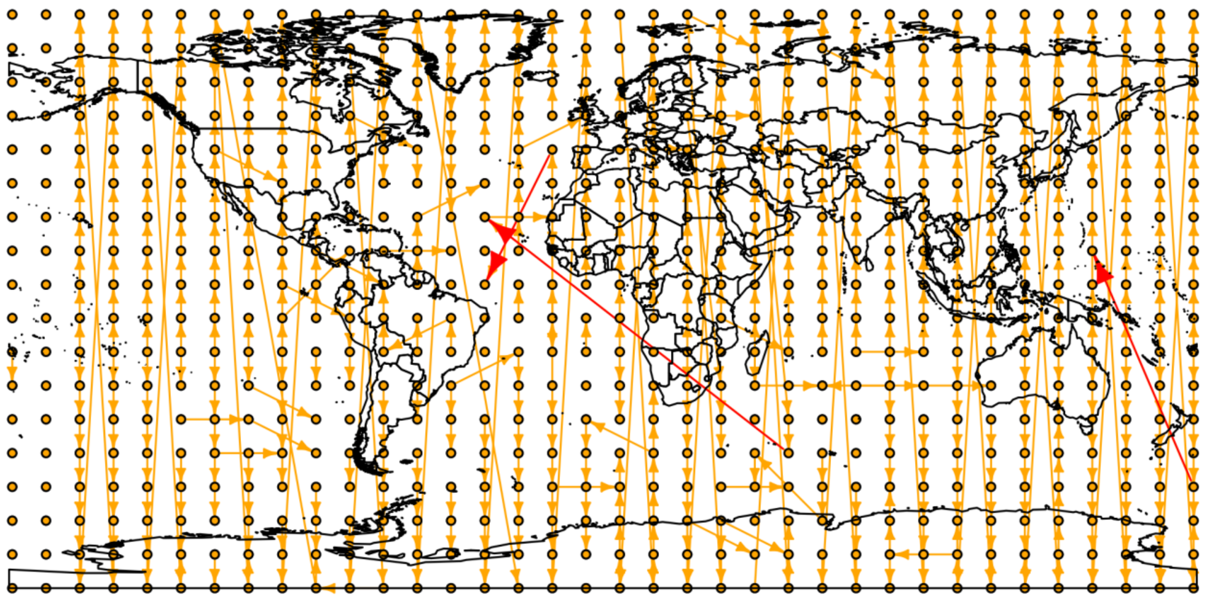


Figure 7.8 The DAG learnt by MMHC for the Weather dataset with OTSL tuning (Table 7.10). The vertices of the world map superimposed over the DAG represent latitude and longitude locations on 10x10 degree grids.

7.5 Conclusions

Learning causal models from observational data remains a major challenge. Traditionally, structure learning algorithms are evaluated and applied to real data with their hyperparameter defaults, or by iterating over a small set of possible hyperparameters. However, no specific set of hyperparameters is optimal for all input datasets which vary in sample size and dimensionality, and structure learning algorithms which vary in learning strategy. Therefore, the question of which hyperparameter values might be best for a given structure learning algorithm and input dataset combination remains an open question.

In this chapter, we propose and evaluate a hyperparameter tuning algorithm, called OTSL, that employs out-of-sample resampling and score-based tuning to find the optimal hyperparameters for a given structure learning algorithm, given the input data. We describe and implement OTSL with a focus on score-based learning, and determine the hyperparameters of different algorithms by optimising either the iss or γ hyperparameters of $\mathrm{BDeu}_{\mathrm{iss}}$ and $\mathrm{EBIC}_{\mathrm{normalised}\ \gamma}$ objective scores.

Synthetic experiments show that tuning with OTSL leads to reasonable improvements in structure learning in terms of the F1 and SHD scores, and when assuming EBIC_{γ} as the objective score for score-based learning. However, this level of improvement is not repeated for $\mathrm{BDeu}_{\mathrm{iss}}$, and this observation is consistent for OTSL and all the other tuning approaches investigated in this study. This is because the hyperparameter default of $\mathrm{iss} = 1$ in $\mathrm{BDeu}_{\mathrm{iss}}$ tends to lead to higher F1 and SHD scores compared to the graphs learnt when $\mathrm{iss} > 1$ (and hence benefits little, if any, from hyperparameter tuning), and this observation is consistent with past studies (Steck, 2008; Uneo, 2011).

The tuning performance of OTSL is evaluated with reference to other hyperparameter tuning approaches for structure learning. We have considered the OCT and StARS tuning approaches, as well as the BIC and AIC model-selection scores that serve as baselines for tuning hyperparameters. Overall, the results show that OTSL provides better tuning performance from results derived across different structure learning algorithms, case studies, and sample sizes. In terms of computational complexity, OTSL was found to be more efficient than OCT but slightly less efficient than StARS.

A limitation is that while OTSL can be applied to structure learning algorithms that come from different classes of learning, it is designed with score-based learning in mind and assumes that the optimal hyperparameters are those that maximise either the $\text{EBIC}_{\text{normalised}\,\gamma}$ or BDeu_{iss} objective scores, and this also applies when tuning CI functions in constraint-based learning. This might explain why the results from tuning score-based learning algorithms are better than those derived from tuning constraint-based learning. Another limitation is that, because OTSL optimises hyperparameters on test data, this process involves resampling multiple training and test datasets from a single input dataset, which impacts the computational efficiency of structure learning; a learning process that is already known to be computationally expensive.

Chapter 8

Conclusions and open problems

This thesis studies structure learning algorithms that recover graphical structure from data, with the main focus being on the problem of latent confounders. This final chapter begins by summarising the outputs and conclusions derived from each chapter, and ends by summarising open problems derived from a relevant paper I co-authored.

8.1 Concluding remarks

Chapter 3 begins by examining how well structure learning algorithms perform when applied to noise-free data that follow the ideal assumptions assumed by the algorithms, as well as imperfect noisy data that contain various types and levels of noise commonly found in real-world datasets. We investigated the impact of data noise using 15 structure learning algorithms from different learning classes. We considered case-study networks from different domains and of varying complexity, with different sample sizes, data noise types, and data noise rates. Although the chapter's main objective was to analyse the impact of data noise on structure learning performance, the results also summarise the performance of these algorithms with and without data noise. For instance, we found that non-exact or simpler learners are more resilient to data noise compared to exact or more sophisticated non-exact learners. The results also indicate that score-based learning generally outperforms constraint-based learning, but a higher fitting score does not necessarily mean a more accurate causal graph. Additionally, while algorithms designed to account for causal insufficiency performed well in noisy experiments involving latent variables, they did not perform as well under other types of data noise. A possible limitation is that this study tested all algorithms using their default hyperparameters as implemented in the structure learning software of the considered packages.

In total, these results were obtained from approximately 10,000 structure learning experiments with a total structure learning runtime of seven months. This large-scale empirical comparison of structure learning algorithms under different data noise assumptions is the first of its kind. The findings suggest that the traditional synthetic performance may overestimate real-world performance by anywhere between 10% and more than 50%. These results have significant implications as they indicate that structure learning accuracy reported in the literature, based on traditional synthetic data, overestimates real-world performance to a greater extent than previously assumed.

Chapter 3 concludes by introducing a novel structure learning algorithm called MAHC that combines pruning and model averaging strategies with hill-climbing search. Comparisons with other algorithms from various learning classes demonstrate that the combination of aggressive pruning and model averaging is effective and efficient, particularly in the presence of data noise. Specifically, the results show that MAHC performs competitively when the input data is clean and often outperforms other algorithms when the input data is noisy. These findings suggest that model averaging strategies may be better suited for learning from real data, assuming that real observations never satisfy the ideal conditions assumed in clean synthetic experiments and often contain different types of data noise, similar to those examined in this chapter.

Chapter 4 expands upon recent advancements in structure learning when dealing with causal insufficiency. It introduces a new algorithm called CCHM, which combines constraint-based and score-based learning techniques with causal effects to learn Gaussian BNs. The constraint-based aspect of CCHM incorporates elements from the state-of-the-art cFCI algorithm, while the score-based component employs a traditional hill-climbing greedy search that minimises the BIC score. CCHM utilises Pearl's do-calculus to orientate edges, a task that most constraint-based and score-based learning do to complete from observational data. The results indicate that CCHM outperforms state-of-the-art algorithms in the majority of the experiments, which include both randomised and real-world Gaussian BNs. However, a limitation of this research is that the algorithm assumes linear GBNs, and requires that the input data are continuous. Chapter 5 extends this and describes a hybrid algorithm, called mFGS-BS, that learns ancestral graphs by calculating the posterior probability of each directed edge being added to the learnt graph, from one observational data set and one or more interventional data sets. Overall, the results show that mFGS-BS improves structure learning accuracy relative to the state-of-the-art and it is computationally efficient. A limitation of mFGS-BS is that it is sensitive to the ordering of the data sets and assumes equal sample size across all input data sets, which is an unrealistic assumption in practice.

Chapter 6 describes two novel algorithms that can be used for both discovery and density estimation of latent confounders in BN structure learning from discrete observational data. Discovering and parameterising latent confounders represent important and challenging problems in causal structure learning and density estimation respectively. These tasks require solutions that come from different areas of statistics and machine learning. Chapter 6 combines elements of variational Bayesian methods, expectation-maximisation, hill-climbing search, and structure learning under the assumption of causal insufficiency. The first algorithm (ILC-V) aims to maximise model-selection accuracy by exhaustively exploring sets of Markov equivalent MAGs. The second algorithm (HCLC-V) aims to balance accuracy relative to computational efficiency by employing hill-climbing over the MAG space, enabling application to larger networks.

Both the ILC-V and HCLC-V algorithms require a PAG to be provided as an input and, because the input PAG will typically indicate multiple possible latent confounders, both algorithms employ the p-ELBO as the objective function to determine the number of the latent confounders, thereby contributing to the discovery process in addition to density estimation of latent confounders. The empirical results show meaningful improvements in maximising the objective score relative to the state-of-the-art, and in some ways in reducing time complexity; although the latter remains a major issue. Two important limitations are that a) both algorithms need to be paired with a structure learning algorithm that produces an ancestral graph, since they require a PAG input to be provided, and b) the results are based on experiments that assume the minimum possible number of latent confounders consistent with the PAG input, which was necessary to ensure that most experiments complete within the 12-hour runtime limit.

Lastly, Chapter 7 delves into the challenge of determining the optimal hyperparameter configuration for structure learning algorithms. Practitioners often encounter this problem when applying structure learning algorithms to their data and typically resort to using default hyperparameters as a solution. In this chapter, a novel hyperparameter tuning method called Out-of-sample Tuning for Structure Learning (OTSL) is described. OTSL utilises out-of-sample and resampling strategies to estimate the optimal hyperparameter configuration for a structure learning algorithm based on the input dataset. The findings indicate that the optimal hyperparameter configuration depends on various factors, including the size and density of the underlying true graph (which is usually unknown), the sample size of the input data, and the specific structure learning algorithm used for tuning. Synthetic experiments demonstrate that OTSL considerably improves graphical accuracy compared to default hyperparameters. Moreover, it outperforms competing algorithms in terms of graphical performance and computational efficiency. However, a limitation is that because OTSL is designed primarily for score-based algorithms, its effectiveness in tuning constraint-based algorithms is not adequate. Another limitation is that because OTSL optimises hyperparameters on test data via resampling, it negatively impacts the computational efficiency of structure learning, which is already known to be computationally expensive.

To facilitate future work, we make all graphs, models and data sets publicly available online as follows:

- CCHM algorithm package: https://github.com/kiattikunc/CCHM
- mFGS-BS algorithm package: https://github.com/kiattikunc/mFGES-BS
- ILC-V and HCLC-V algorithms package: https://github.com/kiattikunc/ILC_and_HCLC_with_VBEM
- OTSL algorithm package: https://github.com/kiattikunc/OTSL
- Case studies and datasets: Bayesys repository http://constantinou.info/downloads/bayesys/bayesys_repository.pdf

8.2 Open problems

In Constantinou et al. (2023) , we examine the challenges of causal structure learning using a unique COVID-19 UK pandemic dataset collated from various public sources (dataset described in subsection 7.4.2). Given that causal models allow us to simulate the impact of hypothetical interventions, we consider the COVID-19 problem, which necessitated prompt and unprecedented decisions in response to unforeseen events, as an ideal test scenario for causal structure learning. This section serves to outline the key issues in causal structure learning based on this case study, to be considered for future research directions.

We investigate the influence of different data formats (discrete, continuous, mixed) on 29 algorithms that belong to various learning classes (constraint-based, score-based, hybrid, continuous optimisation). We assess the outcomes generated by each algorithm, as well as groups of algorithms, in terms of graphical structure, model complexity, sensitivity analysis, confounding variables, predictive and interventional inference. Throughout our analysis, we identify the following open problems:

a) **Large inconsistencies in the learnt output:** The learnt structures show significant inconsistency amongst the different structure learning algorithms analysed. This inconsistency is observed in terms of the number of edges, the specific edges discovered, and the orientation of those edges within the generated graphs. More specifically, the learnt outputs vary in the number of edges - ranging from 7 to 98, and in the number of free parameters – ranging from 162 to above 5 billion. Notably, the level of inconsistency becomes more pronounced when comparing algorithms from different learning classes (e.g., score-based or constraint-based) and when considering different input data formats (e.g., categorical or continuous).

b) **Algorithms are sensitive to the format of the input data:** Many of the structural discrepancies cannot be fully explained by differences between algorithms alone. This is because the same algorithm would often produce very different graphs depending on the input data format.

c) **Algorithms that assume causal insufficiency are also inconsistent:** The inconsistency in the results also applies to algorithms that predict latent variables, by uncovering structures that emphasise potential spurious relationships caused by latent confounders. We would anticipate that these spurious edges would be discovered as edges in the learnt graphs by algorithms that do not account for latent confounders. However, our findings reveal that not only do the algorithms assuming causal insufficiency identify contrasting spurious edges, but many of the predicted spurious edges are absent in the majority of structures learnt by algorithms that do not incorporate latent variables. These inconsistencies in the confounding effects raise questions regarding the effectiveness of the structure learning algorithms that predict latent confounders.

d) **Predictive validation is not adequate:** The extent of inconsistency amongst the structure learning algorithms results in only trivial disparities in predictive validation. However, when the evaluation is expanded to include interventional or sensitivity analyses, substantial differences emerge. These empirical findings emphasise the limitations of predictive validation in being able to differentiate causal systems and offer meaningful insights into causal reasoning.

e) **Model averaging a possible – but an imperfect – solution:** A common approach towards reducing the inconsistency in the learnt graphs involves performing model averaging across a set of graphs, to obtain an average graph that is representative of that set of learnt graphs. This is something that we also investigate, by grouping algorithms in terms of learning class or data format as follows:

- **All_score-based**: the average graph derived from all score-based algorithms.

- **All_constraint-based**: the average graph derived from all constraint-based algorithms.
- **All_hybrid**: the average graph derived from all hybrid learning algorithms.
- **All_quartiles**: the average graph derived from all algorithms applied to the discrete dataset discretised using quartiles.
- **All_k-means**: the average graph derived from all algorithms applied to the discrete dataset discretised using k-means clustering.
- **All_continuous**: the average graph derived from all algorithms applied to the continuous dataset.
- **All_mixed:** the average graph derived from all algorithms applied to the mixed dataset.

While model averaging is found to indeed reduce variability, we also find that the average graphs for each group are all different from one another. Figure 8.1 illustrates the F1 score produced by each average graph relative to a knowledge-based causal graph about COVID-19.

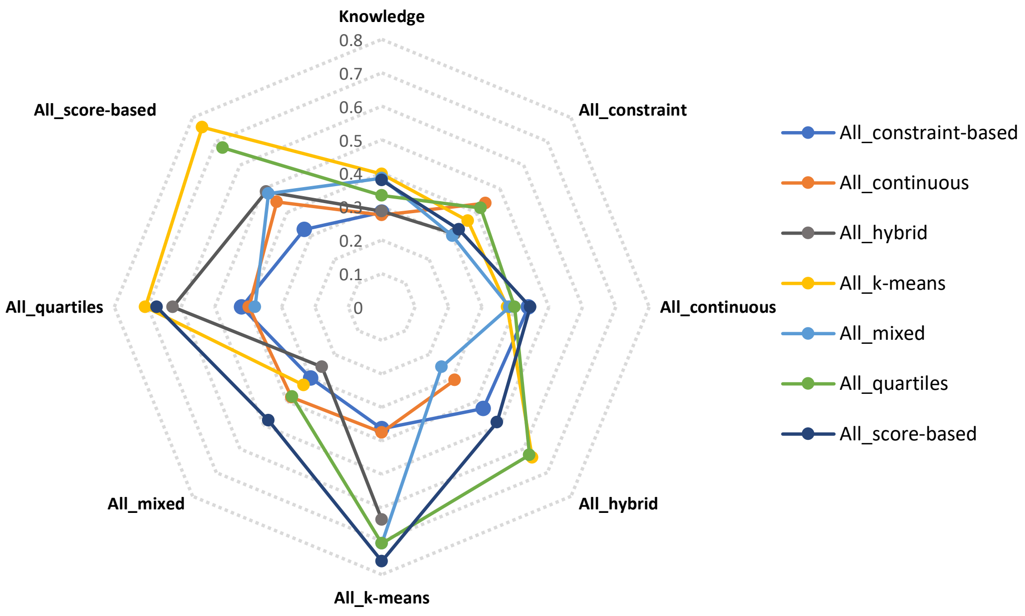


Figure 8.1 The F1 scores produced by each average graph with reference to the knowledge graph.

f) Learning from continuous data is not adequate: The algorithms designed for learning from continuous data demonstrate a tendency to generate considerably denser graphs compared to the graphs they would learn from discretised data. Furthermore, these denser graphs were found to deviate further from the knowledge-based causal graph. This finding also applies to continuous optimisation, which was initially viewed as a promising new learning class through the NOTEARS algorithm (Zheng et al., 2018), but has proven unsatisfactory in practical applications (Constantinou et al., 2021; Kaiser and Sipos, 2022). Based on these observations, we propose that structure learning from continuous data not only poses a substantial risk of model overfitting but also tends to recover numerous edges that are likely associational rather than causal.

Appendix A

The figures presented in this section complement the results presented in Chapter 5.

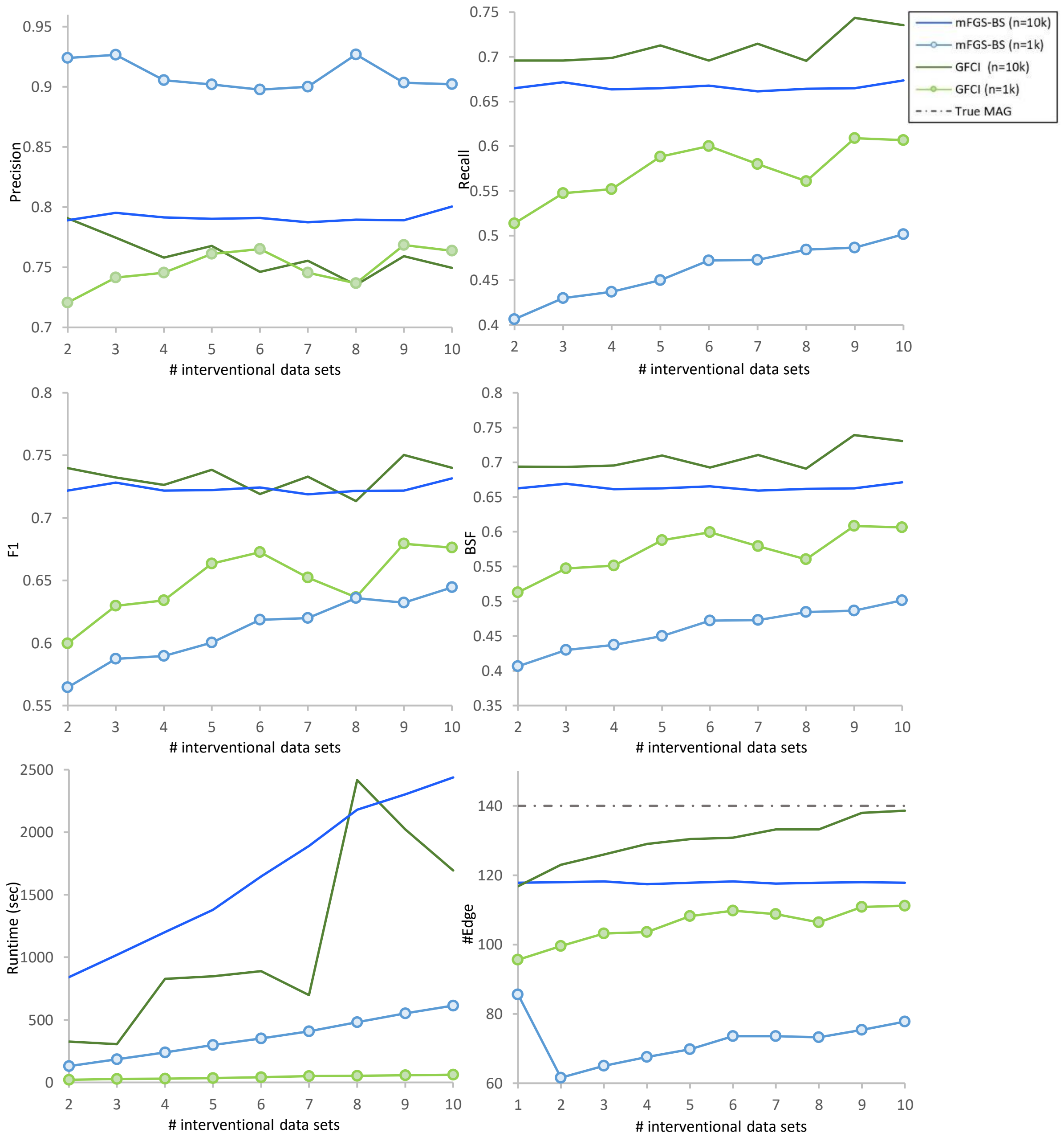


Figure A1 Average performance of the algorithms when applied to synthetic data generated from the Formed network, assuming one intervened variable and 5% latent variables per dataset, over two sample sizes.

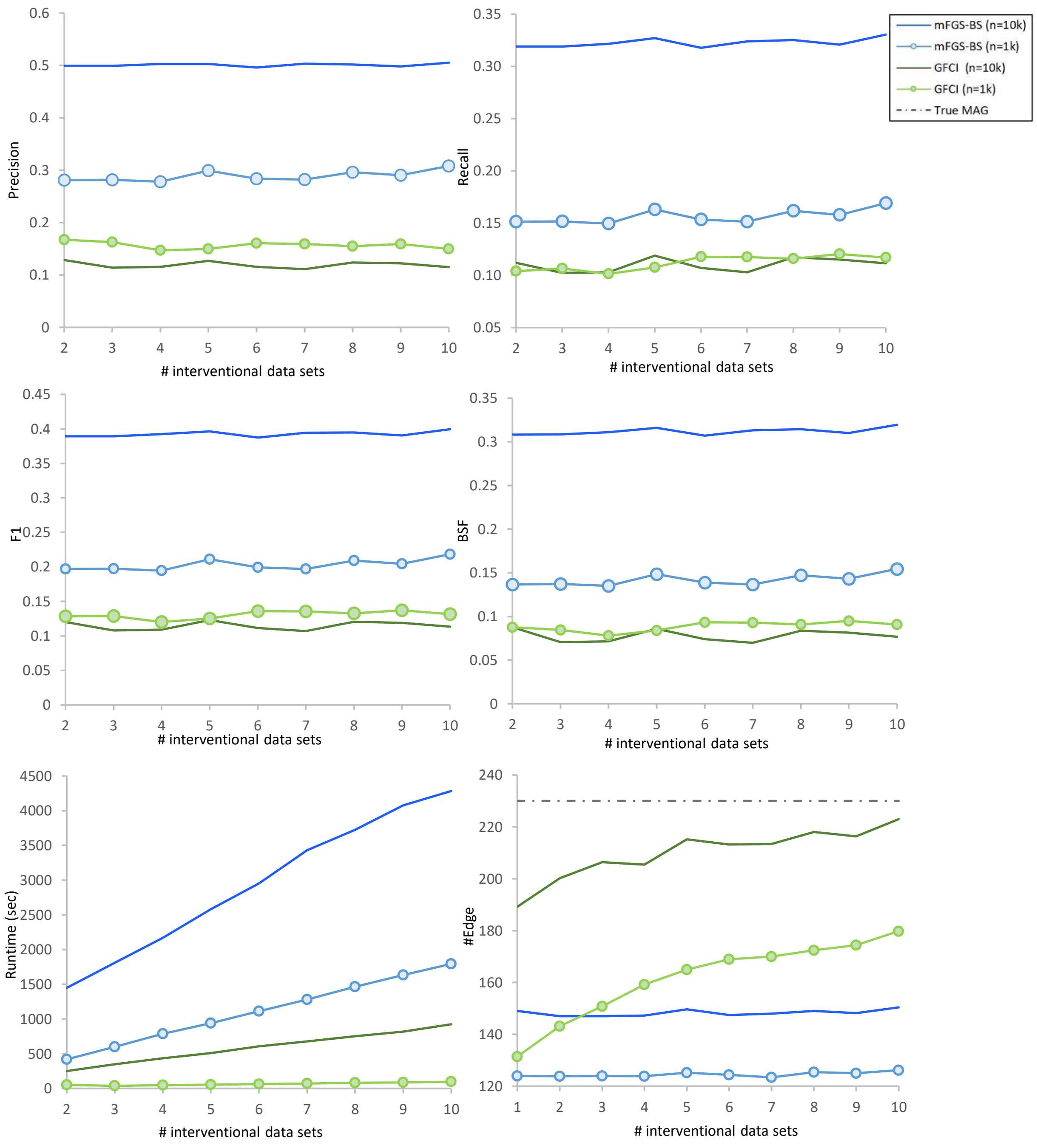


Figure A2 Average performance of the algorithms when applied to synthetic data generated from the Pathfinder network, assuming one intervened variable and 5% latent variables per dataset, over two sample sizes.

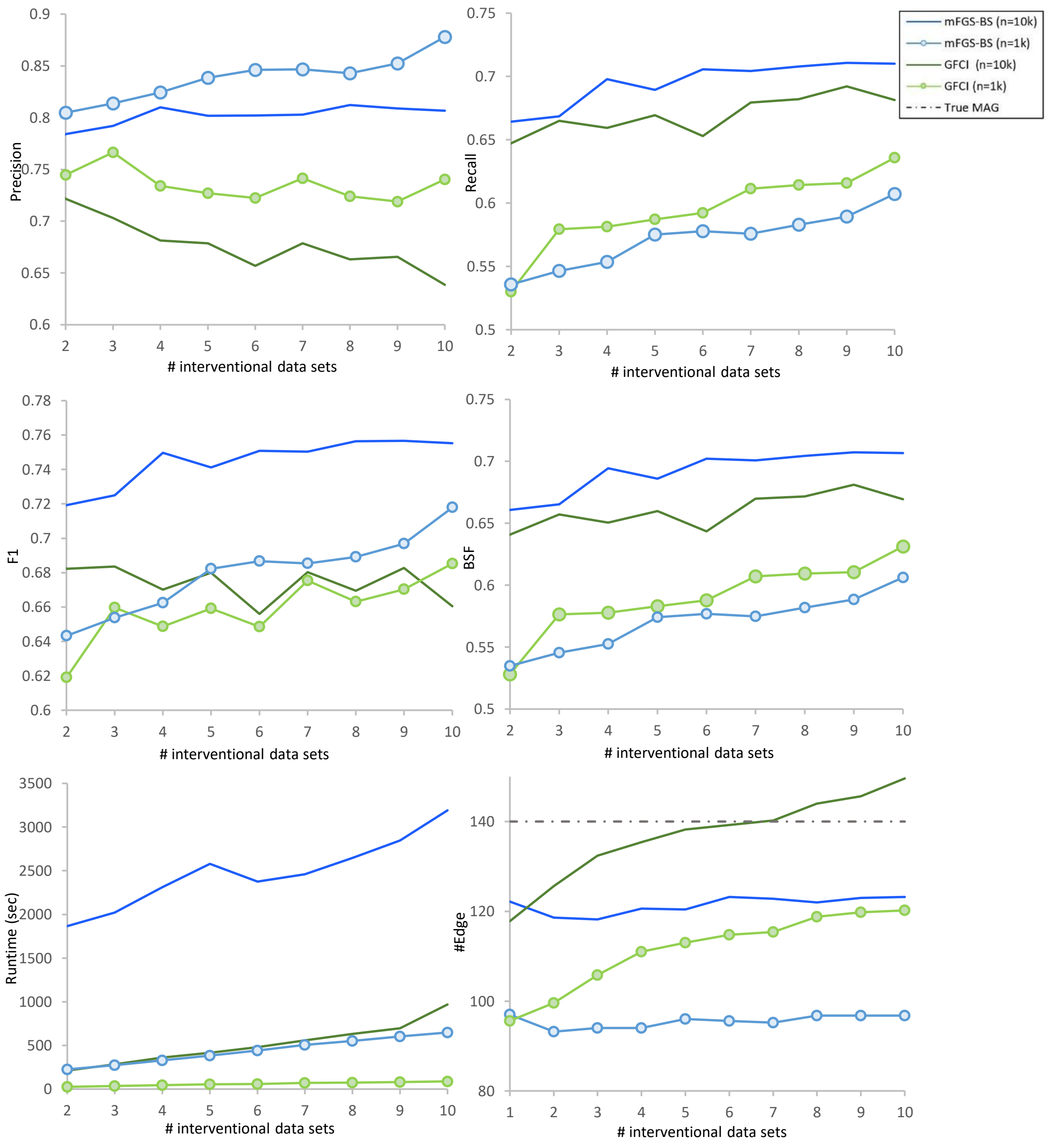


Figure A3 Average performance of the algorithms when applied to synthetic data generated from the Formed network, assuming five intervened variables and 5% latent variables per dataset, over two sample sizes.

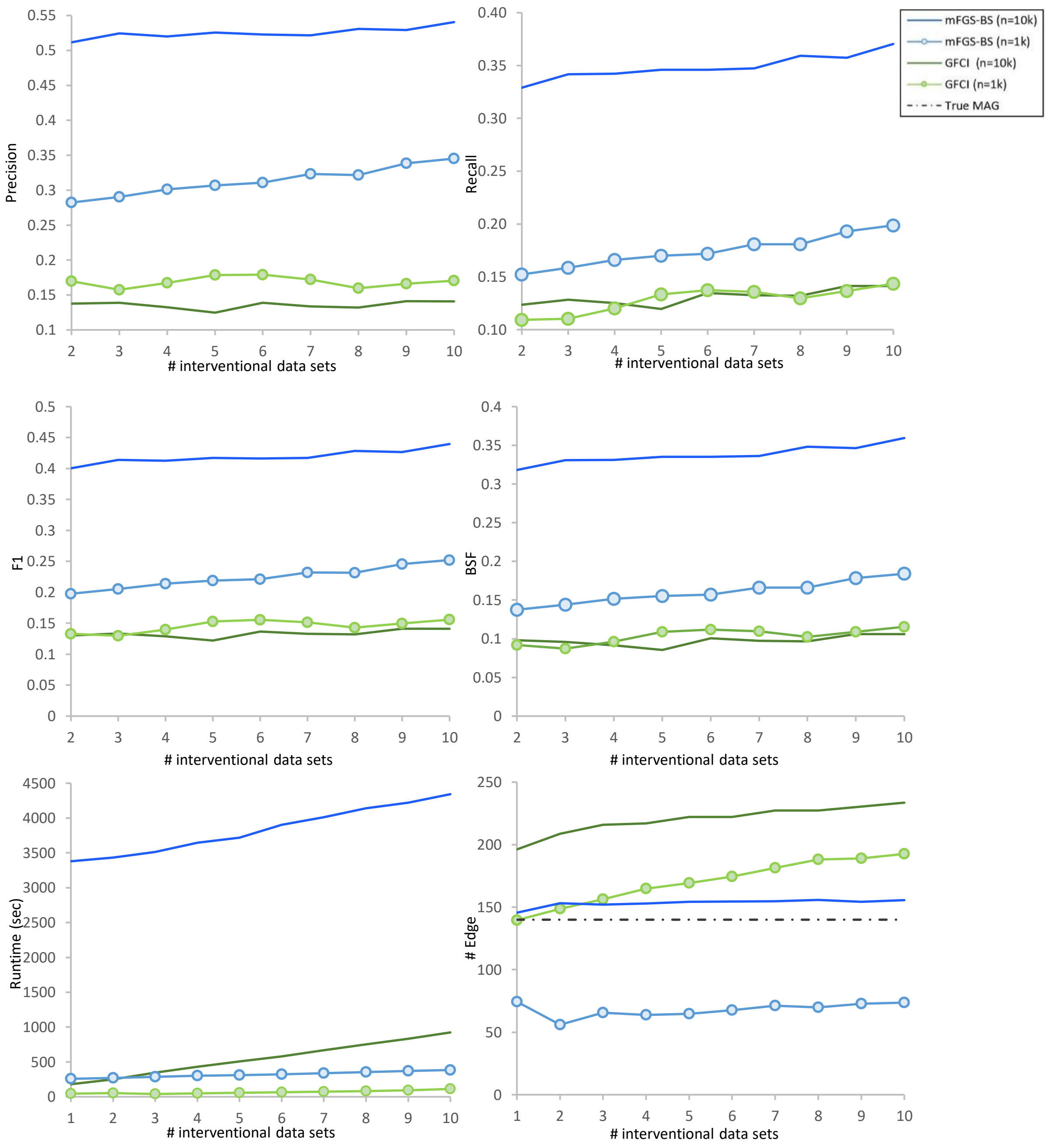


Figure A4 Average performance of the algorithms when applied to synthetic data generated from the Pathfinder network, assuming five intervened variables and 5% latent variables per dataset, over two sample sizes.

Appendix B

The tables presented in this section complement the results presented in Chapter 6.

Table B1 All scores for each algorithm and dataset combination with sample size of 10k, where Memory indicates out-of-memory error in enumerating the possible MAGs, and Timeout indicates failure to complete learning within the 12-hour time limit. The best scores are indicated in bold.

BN (Latent confounder)	**ILC-V$_{FCI}$**	**HCLC-V$_{FCI}$**	**ILC-V$_{GFCI}$**	**HCLC-V$_{GFCI}$**	**CIL**	**GLSL**	**ILC-V$_{FCI}$**	**HCLC-V$_{FCI}$**	**ILC-V$_{GFCI}$**	**HCLC-V$_{GFCI}$**	**CIL**	**GLSL**
			p-ELBO						**Runtime (Sec)**			
Asia (smoke)	-17,860	-17,860	-17,601	-17,601	-17,039	**-16,135**	**119**	100	155	106	302	947
Sports (Rdlevel)	**-92,014**	-92,864	**-92,014**	-92,864	-99,741	-99,741	**431**	727	612	789	533	628
Property (propertyPurchaseValue)	-285,084	-285,084	**-238,090**	-238,267	-283,142	-275,212	6235	5,630	59,806	51,545	**3,124**	18,270
Property (borrowing)	-277,035	-277,035	**-239,289**	-239,520	-277,440	-269,719	21,655	5,019	44,331	22,239	**3,317**	17,522
Property (otherPropertyExpenses)	-284,024	-284,038	-237,178	**-236,998**	-285,975	-277,949	3,332	**2,269**	3,893	18,367	3,182	19,273
Alarm (INTUBATION)	-119,906	-119,845	**-104,919**	-105,096	-133,084	Memory	**2,258**	5,502	14,357	22,508	6,584	Timeout
Alarm (HYPOVOLEMIA)	Memory	-126,194	**-101,997**	-102,960	-131,819	Memory	Memory	3,739	12,605	**3,565**	9,796	Timeout
Alarm (LVFAILURE)	Memory	-129,574	-103,761	**-103,720**	-134,606	Memory	Memory	**2,510**	16,050	3,920	5,439	Timeout
Alarm (ERRCAUTER)	Memory	-121,536	**-103,492**	-103,530	-132,280	Memory	Memory	**2,253**	21,023	3,594	5,979	Timeout
Alarm (PULMEMBOLUS)	Memory	-126,811	-103,652	**-103,624**	-135,116	Memory	Memory	**2,606**	12,306	3,163	5,841	Timeout
Alarm (KINKEDTUBE)	Memory	-125,698	-108,480	**-102,803**	-134,869	Memory	Memory	2,790	12,283	**2,029**	5,261	Timeout
			LL						**BIC**			
Asia (smoke)	-17,857	-17,857	-17,599	-17,599	-16,986	**-1,6069**	-17,977	-17,977	-17,764	-17,764	-17,087	**-16,208**
Sports (Rdlevel)	**-91,876**	-92,630	**-91,876**	-92,630	-99,327	-99,327	**-93,000**	-93,887	**-93,000**	-93,887	-100,009	-100,009
Property (propertyPurchaseValue)	-284,729	-284,780	**-237,915**	-238,285	-281,780	-274,089	-288,312	-288,450	-252,720	**-252,321**	-283,797	-276,576
Property (borrowing)	-276,671	-276,671	**-239,296**	-239,378	-276,006	-268,558	-279,835	-279,835	**-245,725**	-246,364	-278,096	-271,202
Property (otherPropertyExpenses)	-283,867	-283,867	**-237,110**	-237,256	-284,519	-276,810	-287,165	-287,796	**-244,874**	-245,048	-286,660	-279,421
Alarm (INTUBATION)	-119,729	-119,758	**-104,551**	-104,742	-132,401	Timeout	-121,921	-121,881	**-110,036**	-110,245	-133,594	Timeout
Alarm (HYPOVOLEMIA)	Memory	-126,100	**-101,914**	-102,852	-131,124	Timeout	Memory	-128,039	**-104,889**	-105,873	-132,340	Timeout
Alarm (LVFAILURE)	Memory	-129,488	-103,619	**-103,617**	-133,902	Timeout	Memory	-131,335	-106,691	**-106,712**	-135,141	Timeout
Alarm (ERRCAUTER)	Memory	-121,443	**-103,416**	-103,429	-131,578	Timeout	Memory	-123,317	**-106,386**	-106,427	-132,812	Timeout
Alarm (PULMEMBOLUS)	Memory	-126,715	-103,512	**-103,502**	-134,418	Timeout	Memory	-128,622	**-106,496**	-106,504	-135,639	Timeout
Alarm (KINKEDTUBE)	Memory	-116,421	-108,034	**-102,706**	-134,162	Timeout	Memory	-118,328	-111,299	**-105,262**	-135,405	Timeout

Table B2 All scores for each algorithm and dataset combination with sample size of 1k, where Memory indicates out-of-memory error in enumerating the possible MAGs, and Timeout indicates failure to complete learning within the 12-hour time limit. The best scores are indicated in bold.

BN (Latent confounder)	**ILC-V$_{FCI}$**	**HCLC-V$_{FCI}$**	**ILC-V$_{GFCI}$**	**HCLC-V$_{GFCI}$**	**CIL**	**GLSL**	**ILC-V$_{FCI}$**	**HCLC-V$_{FCI}$**	**ILC-V$_{GFCI}$**	**HCLC-V$_{GFCI}$**	**CIL**	**GLSL**
			p-ELBO						**Runtime (sec)**			
Asia (smoke)	-1,845	-1,845	-1,807	-1,807	-1,796	**-1,679**	45	114	40	61	**17**	109
Sports (Rdlevel)	**-9,296**	-9,417	**-9,296**	-9,417	-10,228	-10,228	135	68	353	74	55	**50**
Property (propertyPurchaseValue)	-34,496	-34,532	**-24,565**	-24,596	-29,040	-28,076	1,775	**188**	1,918	557	212	1,864
Property (borrowing)	-35,042	-35,080	Memory	**-24,044**	-28,518	-27,534	505	**41**	Memory	1,794	227	1,873
Property (otherPropertyExpenses)	-35,929	-35,979	**-24,079**	**-24,079**	-29,382	-28,363	533	**55**	10,311	628	222	1,964
Alarm (INTUBATION)	Memory	-14,802	**-10,966**	-11,068	-13,777	-11,581	Memory	**288**	4,862	2,004	429	1,878
Alarm (HYPOVOLEMIA)	Memory	-14,660	**-10,908**	-11,010	-13,721	-11,117	Memory	**291**	1,769	665	518	13,740
Alarm (LVFAILURE)	Memory	-14,821	**-11,074**	-11,075	-13,989	-11,307	Memory	**227**	1,082	473	377	7,763
Alarm (ERRCAUTER)	Memory	-14,678	-11,024	**-11,017**	-13,693	-11,254	Memory	**213**	5,509	386	307	6,432
Alarm (PULMEMBOLUS)	Memory	-15,081	**-11,053**	-11,055	-13,994	-11,294	Memory	**181**	1,184	418	303	14,328
Alarm (KINKEDTUBE)	Memory	-14,948	**-10,889**	-10,963	-13,896	-11,203	Memory	**294**	5,017	374	356	6,119
			LL						**BIC**			
Asia (smoke)	-1,843	-1,843	-1,806	-1,806	-1,757	**-1,628**	-1,932	-1,932	-1,930	-1,930	-1,840	**-1,794**
Sports (Rdlevel)	**-9,245**	-9,323	**-9,245**	-9,323	-9,983	-9,983	**-10,011**	-10,159	**-10,011**	-10,159	-10,460	-10,460
Property (propertyPurchaseValue)	-34,451	-34,467	**-24,427**	-24,478	-28,266	-27,405	-35,308	-35,330	-29,645	-29,552	-29,737	**-29,132**
Property (borrowing)	-34,998	-35,015	Memory	**-24,366**	-27,709	-26,846	-35,810	-35,833	Memory	-29,143	-29,235	**-28,691**
Property (otherPropertyExpenses)	-35,882	-35,914	**-24,240**	-24,548	-28,563	-27,685	-36,732	-36,746	-29,518	-29,874	-30,128	**-29,505**
Alarm (INTUBATION)	Memory	-14,722	-10,947	-11,002	-13,334	**-10,906**	Memory	-15,596	-14,239	-14,421	-14,229	**-13,355**
Alarm (HYPOVOLEMIA)	Memory	-12,823	**-10,791**	-10,900	-13,272	-10,446	Memory	-14,301	**-12,870**	-12,993	-14,190	-14,646
Alarm (LVFAILURE)	Memory	-13,177	**-10,956**	-10,962	-13,539	-10,619	Memory	-14,580	**-13,011**	-13,024	-14,457	-14,857
Alarm (ERRCAUTER)	Memory	-14,624	-10,902	-10,909	-13,241	**-10,571**	Memory	-15,473	**-12,989**	-12,995	-14,160	-14,816
Alarm (PULMEMBOLUS)	Memory	-15,009	-10,935	-10,938	-13,543	**-10,608**	Memory	-15,873	**-12,976**	-12,990	-14,451	-14,718
Alarm (KINKEDTUBE)	Memory	-14,901	-10,825	-10,850	-13,443	**-10,526**	Memory	-15,781	**-12,777**	-12,839	-14,368	-14,791

Bibliography

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

M. J. Beal and Z. Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Statistics*, 07 2002.

M. J. Beal and Z. Ghahramani. Variational Bayesian learning of directed graphical models with hidden variables. *Bayesian Analysis*, 1:793–831, 2006.

I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In J. Hunter, J. Cookson, and J. Wyatt, editors, AIME 89, pages 247–256, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg. ISBN 978-3-642-93437-7.

D. I. Bernstein, B. Saeed, C. Squires, and C. Uhler. Ordering-based causal structure learning in the presence of latent variables. In *International Conference on Artificial Intelligence and Statistics*, 2019.

C. M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

K. Biza, I. Tsamardinos, and S. Triantafillou. Tuning causal discovery algorithms. In M. Jaeger and T. D. Nielsen, editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 17–28. PMLR, 23–25 Sep 2020.

K. Biza, I. Tsamardinos, and S. Triantafillou. Out-of-sample tuning for causal discovery. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2022. doi: 10.1109/TNNLS.2022.3185842.

R. Bouckaert. Bayesian Belief Networks: From Construction to Inference. Universiteit Utrecht, Faculteit Wiskunde en Informatica, 1995. ISBN 9789039308486.

R. Castelo and A. Siebes. Priors on network structures. biasing the search for Bayesian networks. *International Journal of Approximate Reasoning*, 24(1):39–57, 2000. ISSN 0888-613X. doi: https://doi.org/10.1016/S0888-613X(99)00041-9.

J. Chen and Z. Chen. Extended BIC for small-n-large-p sparse GLM. *Statistica Sinica*, 22, 04 2012. doi: 10.5705/ss.2010.216.

J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1):43–90, 2002. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(02)00191-1.

Y. Chen and J. Tian. Finding the k-best equivalence classes of Bayesian network structures for model averaging. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), Jun. 2014. doi:10.1609/aaai.v28i1.9064.

D. M. Chickering. Optimal structure identification with greedy search. J. Mach. Learn. Res., 3 (null):507–554, Mar. 2003. ISSN 1532-4435. doi:10.1162/153244303321897717.

K. Chobtham and A. C. Constantinou. Bayesian network structure learning with causal effects in the presence of latent variables. In M. Jaeger and T. D. Nielsen, editors, *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 101–112. PMLR, 23–25 Sep 2020.

K. Chobtham and A. C. Constantinou. Discovery and density estimation of latent confounders in Bayesian networks with evidence lower bound. In *Proceedings of the 11th International Conference on Probabilistic Graphical Models*, volume 186 of *Proceedings of Machine Learning Research*, pages 121–132. PMLR, 05–07 Oct 2022.

K. Chobtham, A. C. Constantinou, and N. K. Kitson. Hybrid Bayesian network discovery with latent variables by scoring multiple interventions. *Data Mining and Knowledge Discovery* volume, 37: pages 476–520, 2023. doi: 10.1007/s10618-022-00882-9.

K. Chobtham and A. C. Constantinou. Tuning structure learning algorithms with out-of-sample and resampling strategies, arXiv:2306.13932, 2023. URL https://arxiv.org/abs/2306.13932.

Z. Chun. A survey of selective ensemble learning algorithms. *Chinese Journal of Computers*, 2011.

J. Coid, S. Ullrich, C. Kallis, M. Freestone, R. Gonzalez, L. Bui, A. Igoumenou, A. Constantinou, N. Fenton, W. Marsh, M. Yang, B. DeStavola, J. Hu, J. Shaw, M. Doyle, L. Archer-Power, M. Davoren, B. Osumili, P. Mccrone, and P. Bebbington. Improving risk management for violence in mental health services: a multimethods approach. *Programme Grants for Applied Research*, 4:1–408, 11 2016. doi: 10.3310/pgfar04160.

D. Colombo and M. H. Maathuis. Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research*, 15(116):3921–3962, 2014.

D. Colombo, M. Maathuis, M. Kalisch, and T. Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *Annals of Statistics* - ANN STATIST, 40, 04 2011. doi: 10.1214/11-AOS940.

A. C. Constantinou. Evaluating structure learning algorithms with a balanced scoring function. *CoRR*, arXiv: 1905.12666, 2019.

A. C. Constantinou. Learning Bayesian networks that enable full propagation of evidence. *IEEE Access*, 8:124845–124856, 2020.

A. C. Constantinou and N. Fenton. The future of the London buy-to-let property market: Simulation with temporal Bayesian networks. *PLOS ONE*, 12(6):1–30, 06 2017. doi: 10.1371/journal.pone.0179297.

A. C. Constantinou, N. E. Fenton, and M. Neil. Profiting from an inefficient association football gambling market: Prediction, risk and uncertainty using Bayesian networks. *Knowledge-Based Systems*, 50:60–86, 2013. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2013.05.008.

A. C. Constantinou, M. Freestone, W. Marsh, N. Fenton, and J. Coid. Risk assessment and risk management of violent reoffending among prisoners. *Expert Systems with*

Applications, 42(21): 7511–7529, 2015. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2015.05.025.

A. C. Constantinou, N. Fenton, and M. Neil. Integrating expert knowledge with data in Bayesian networks: Preserving data-driven expectations when the expert variables remain unobserved. *Expert Systems with Applications*, 56:197–208, 2016. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.02.050.

A. C. Constantinou, Y. Liu, K. Chobtham, Z. Guo, and N. K. Kitsoni. The Bayesys data and Bayesian network repository. Queen Mary University of London, London, UK. [Online], 2020. URL http://bayesian-ai.eecs.qmul.ac.uk/bayesys/.

A. C. Constantinou, Y. Liu, K. Chobtham, Z. Guo, and N. K. Kitson. Large-scale empirical validation of Bayesian network structure learning algorithms with noisy data. *International Journal of Approximate Reasoning*, 131:151–188, 2021. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2021.01.001.

A. C. Constantinou, Y. Liu, N. K. Kitson, K. Chobtham, and Z. Guo. Effective and efficient structure learning with pruning and model averaging strategies. *International Journal of Approximate Reasoning*, 151:292–321, 2022. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2022.09.016.

A. C. Constantinou, N. K. Kitson, Y. Liu, K. Chobtham, A. Hashemzadeh, P. A. Nanavati, R. Mbuvha, and B. Petrungaro. Open problems in causal structure learning: A case study of COVID-19 in the UK, *Expert Systems with Applications*, Vol 234, 2023, 121069, ISSN 0957-4174.

G. F. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, page 116–125, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606149.

T. M. Cover and J. A. Thomas. Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience, USA, 2006. ISBN 0471241954.

J. Cussens. Bayesian network learning with cutting planes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, page 153–160, Arlington, Virginia, USA, 2011. AUAI Press. ISBN 9780974903972.

J. Cussens. An upper bound for BDeu local scores. 2012.

C. P. de Campos, Z. Zeng, and Q. Ji. Structure learning of Bayesian networks using constraints. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 113–120, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553389.

A. de Waal, H. Koen, P. de Villiers, H. Roodt, N. Moorosi, and G. Pavlin. Construction and evaluation of Bayesian networks with expert-defined latent variables. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 774–781, 2016.

J. de Zoete, N. Fenton, T. Noguchi, and D. Lagnado. Resolving the so-called “probabilistic paradoxes in legal reasoning” with Bayesian networks. *Science and Justice*, 59(4):367–379, 2019. ISSN 1355-0306. doi: https://doi.org/10.1016/j.scijus.2019.03.003.

M. Drton, M. Eichler, and T. Richardson. Computing maximum likelihood estimates in recursive linear models with correlated errors. *Journal of Machine Learning Research*, 10, 01 2006. doi: 10.1145/1577069.1755864.

D. Eaton and K. Murphy. Exact Bayesian structure learning from uncertain interventions. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 107–114, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR.

B. Efron and R. Tibshirani. An Introduction to the Bootstrap. Monographs on statistics and applied probability. 1994.

R. Fisher. The design of experiments. 1935. Oliver and Boyd, Edinburgh, 1935.

R. Fisher. On the probable error of the correlation coefficient to a second approximation. In *Metron*, volume 1, pages 1–32, 1921.

R. Foygel and M. Drton. Extended Bayesian Information Criteria for Gaussian Graphical Models. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS’10, page 604–612, Red Hook, NY, USA, 2010. Curran Associates Inc.

N. Friedman. The Bayesian structural EM algorithm. In *Conference on Uncertainty in Artificial Intelligence*, 1998.

M. Gasse, A. Aussem, and H. Elghazel. A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications*, 41(15): 6755–6772, 2014. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2014.04.032.

M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider. 1 potassco: The potsdam answer set solving collection. *AI Commun*., 24:107–124, 01 2011. doi: 10.3233/AIC-2011-0491.

D. Geiger and D. Heckerman. Learning Gaussian networks. In R. L. de Mantaras and D. Poole, editors, *Uncertainty Proceedings* 1994, pages 235–243. Morgan Kaufmann, San Francisco (CA), 1994. ISBN 978-1-55860-332-5. doi: https://doi.org/10.1016/B978-1-55860-332-5.50035-3.

M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. InR. Kowalski, Bowen, and Kenneth, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

A. Gelman, J. Carlin, H. Stern, and D. Rubin. Bayesian Data Analysis, Second Edition. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2003. ISBN 9781420057294.

R. J. B. Goudie and S. Mukherjee. A Gibbs sampler for learning DAGs. *Journal of Machine Learning Research*, 17(30):1–39, 2016.

X. Guo, Y. Wang, X. Huang, S. Yang, and K. Yu. Bootstrap-based causal structure learning. In ”*Proceedings of the 31st ACM International Conference on Information and Knowledge Management*”, CIKM ’22, page 656–665, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392365. doi: 10.1145/3511808.3557249. URL https://doi.org/10.1145/3511808.3557249.

Z. Guo and A. C. Constantinou. Approximate learning of high dimensional Bayesian network structures via pruning of candidate parent sets. *Entropy*, 22(10), 2020. ISSN 1099-4300. doi:10.3390/e22101142.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444.

A. Hauser and P. Buhlmann. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs, J. Mach. Learn. Res., 13(1):2409–2464, Aug 2012. ISSN.

J. Hausser and K. Strimmer. Entropy inference and the james-stein estimator, with application to nonlinear gene association networks. *J. Mach. Learn. Res.*, 10:1469–1484, Dec 2009. ISSN 1532-4435.

D. Heckerman, E. Horvitz, and B. Nathwani. Toward normative expert systems: Part I the pathfinder project. *Methods of information in medicine*, 31:90–105, 07 1992. doi: 10.1055/s-0038-1634867.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. In R. L. de Mantaras and D. Poole, editors, *Uncertainty Proceedings* 1994, pages 293–301. Morgan Kaufmann, San Francisco (CA), 1994. ISBN 978-1-55860-332-5. doi: https://doi.org/10.1016/B978-1-55860-332-5.50042-0.

A. Hyttinen, F. Eberhardt, and M. Jarvisalo. Constraint-based causal discovery: Conflict resolution with answer set programming. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI’14, page 340–349, Arlington, Virginia, USA, 2014. AUAI Press. ISBN 9780974903910.

F. Jabbari and G. Cooper. An instance-specific algorithm for learning the structure of causal Bayesian networks containing latent variables. *Proceedings of the SIAM International Conference on Data Mining*, pages 433–441, 03 2020. doi: 10.1137/1.9781611976236.49.

F. Jabbari, J. Ramsey, P. Spirtes, and G. F. Cooper. Discovery of causal models that contain latent variables through Bayesian scoring of independence constraints. *Machine learning and knowledge discovery in databases : European Conference*, ECML PKDD, 2017:142–157, 2017.

W. James and C. Stein. Estimation with Quadratic Loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1: *Contributions to the Theory of Statistics*, pages 361–379, Berkeley, Calif., 1961. University of California Press.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An Introduction to Variational Methods for Graphical Models, page 105–161. MIT Press, Cambridge, MA, USA, 1999.

M. Kaiser and M. Sipos. Unsuitability of NOTEARS for causal graph discovery when dealing with dimensional quantities. *Neural Process. Lett.*, 54(3):1587–1595, jun 2022. ISSN 1370-4621. doi: 10.1007/s11063-021-10694-5. URL https://doi.org/10.1007/s11063-021-10694-5.

M. Kalisch, M. Machler, D. Colombo, M. H. Maathuis, and P. B ¨ uhlmann. Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software*, 47, 2012. doi: 10.18637/jss.v047.i11. URL http://CRAN.R-project.org/package=pcalg.

E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, M. Chelliah, W. Ebisuzaki, W. Higgins, J. Janowiak, K. C. Mo, C. Ropelewski, J. Wang, A. Leetmaa, R. Reynolds, R. Jenne, and D. Joseph. The ncep/ncar 40-year reanalysis project. *Bulletin of the American Meteorological Society*, 1996. URL https://psl.noaa.gov/data/gridded/data.ncep.reanalysis.html.

D. P. Kingma and M. Welling. Auto-encoding Variational Bayes, 2013. URL https://arxiv.org/abs/1312.6114.

N. K. Kitson and A. C. Constantinou. Learning Bayesian networks from demographic and health survey data. *Journal of Biomedical Informatics*, 113:103588, 2021. ISSN 1532-0464. doi: https://doi.org/10.1016/j.jbi.2020.103588.

N. K. Kitson, A. C. Constantinou, Z. Guo, Y. Liu, and K. Chobtham. A survey of Bayesian network structure learning. *Artificial Intelligence Review*, 2023. ISSN 1573-7462. doi: 10.1007/s10462-022-10351-w.

K. B. Korb, L. R. Hope, A. E. Nicholson, and K. Axnick. Varieties of causal intervention. In C. Zhang, H. W. Guesgen, and W.-K. Yeap, editors, *PRICAI 2004: Trends in Artificial Intelligence*, pages 322–331, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

J. Kuipers, P. Suter, and G. Moffa. Efficient sampling and structure learning of Bayesian networks. *Journal of Computational and Graphical Statistics*, 31(3):639–650, 2022. doi: 10.1080/10618600.2021.2020127.

E. Kummerfeld. A simple interpretation of undirected edges in essential graphs is wrong. *PLOS ONE*, 16(4):1–12, 04 2021. doi: 10.1371/journal.pone.0249415. URL https://doi.org/10.1371/journal.pone.0249415.

L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988.

H. Liu, K. Roeder, and L. Wasserman. Stability Approach to Regularization Selection (StARS) for high dimensional graphical models, In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 1432–1440, Red Hook, NY, USA, 2010. Curran Associates Inc.

C. Ma, S. Tschiatschek, J. M. Hernandez-Lobato, R. Turner, and C. Zhang. VAEM: A deep generative model for heterogeneous mixed type data, 2020.

URL https://arxiv.org/abs/2006.11941.

M. H. Maathuis, M. Kalisch, and P. Buhlmann. Estimating high-dimensional intervention effects from observational data. *The Annals of Statistics*, 37(6A):3133–3164, Dec 2009. ISSN 0090-5364. doi: 10.1214/09-aos685.

D. Madigan, S. A. Andersson, M. D. Perlman, and C. T. Volinsky. Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Communications in Statistics - Theory and Methods*, 25(11):2493–2519, 1996. doi: 10.1080/03610929608831853.

S. Magliacane, T. Claassen, and J. M. Mooij. Ancestral causal inference, In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4473–4481, Red Hook, NY, USA, 2016. Curran Associates Inc.

D. Malinsky and P. Spirtes. Estimating bounds on causal effects in high-dimensional and possibly confounded systems. *International Journal of Approximate Reasoning*, 88, 06 2017. doi: 10.1016/j.ijar.2017.06.005.

D. Margaritis. Distribution-free learning of Bayesian network structure in continuous domains. In *Proceedings of the 20th National Conference on Artificial Intelligence* - Volume 2, AAAI'05, page 825–830. AAAI Press, 2005. ISBN 157735236x.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

Y. McLatchie, S. Rognvaldsson, F. Weber, and A. Vehtari. Robust and efficient projection predictive inference, 2023.

J. M. Mooij, S. Magliacane, and T. Claassen. Joint causal inference from multiple contexts, *J. Mach. Learn. Res.*, 21(1), Jan 2020. ISSN 1532-4435.

I. Niemela. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999. ISSN 1012-2443. doi: 10.1023/A:1018930122475.

C. Nowzohour, M. Maathuis, and P. Buhlmann. Structure learning with bow-free acyclic path diagrams, *Stat*, 1050:7, 2015.

J. M. Ogarrio, P. Spirtes, and J. Ramsey. A hybrid causal search algorithm for latent variable models. In A. Antonucci, G. Corani, and C. P. Campos, editors, *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, pages 368–379, 2016.

J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1988

J. Pearl. Causality: Models, reasoning, and inference, second edition. Causality, 29, 01 2000. doi:10.1017/CBO9780511803161.

J. Pearl and D. Mackenzie. The Book of Why: The New Science of Cause and Effect. Penguin Books Limited, 2018. ISBN 9780241242643.

K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900. doi: 10.1080/14786440009463897.

J. Peters and P. Buhlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 11 2013. ISSN 0006-3444. doi: 10.1093/biomet/ast043.

J. Piironen and A. Vehtari. Comparison of Bayesian predictive methods for model selection. *Statistics and Computing*, 27(3):711–735, Apr. 2016. ISSN 1573-1375. doi: 10.1007/s11222-016-9649-y.

J. Ramsey, P. Spirtes, and J. Zhang. Adjacency-faithfulness and conservative causal inference. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, page 401–408, Arlington, Virginia, USA, 2006. AUAI Press. ISBN 0974903922.

J. D. Ramsey. Scaling up greedy equivalence search for continuous variables. *CoRR*, abs/1507.07749, 2015.

K. Rantanen, A. Hyttinen, and M. Jarvisalo. Maximal ancestral graph structure learning via exact search. In C. de Campos and M. H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 1237–1247. PMLR, 27–30 Jul 2021.

T. Richardson and P. Spirtes. Ancestral graph Markov models. *Annals of Statistics*, 30, 11 2000. doi: 10.1214/aos/1031689015.

D. Rickles. Causality in complex interventions. *Medicine, Health Care and Philosophy*, 12:77–90, 2009.

F. Rodriguez-Sanchez. mpc-mixed library, 2021. URL https://github.com/ferjorosa/mpc-mixed.

F. Rodriguez-Sanchez, P. Larranaga, and C. Bielza. Incremental learning of latent forests. *IEEE Access*, 8:224420–224432, 2020. doi: 10.1109/ACCESS.2020.3027064.

F. Rodriguez-Sanchez, C. Bielza, and P. Larranaga. Multipartition clustering of mixed data with Bayesian networks. *International Journal of Intelligent Systems*, 37(3):2188–2218, 2022. doi: https://doi.org/10.1002/int.22770.

M. Scanagatta, G. Corani, and M. Zaffalon. Improved local search in Bayesian networks structure learning. In A. Hyttinen, J. Suzuki, and B. Malone, editors, *Proceedings of The* 3^{rd} *International Workshop on Advanced Methodologies for Bayesian Networks*, volume 73 of *Proceedings of Machine Learning Research*, pages 45–56. PMLR, 20–22 Sep 2017.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. ISSN 00905364. URL http://www.jstor.org/stable/2958889.

M. Scutari. An empirical-Bayes score for discrete Bayesian networks, In *European Workshop on Probabilistic Graphical Models*, 2016.

M. Scutari. Bnlearn dataset repository, 2019.

URL https://www.bnlearn.com/bnrepository.

M. Scutari and A. Brogini. Bayesian network structure learning with permutation tests. *Communications in Statistics - Theory and Methods*, 41(16-17):3233–3243, 2012.

M. Scutari, C. E. Graafland, and J. M. Gutierrez. Who Learns Better Bayesian Network Structures: Accuracy and Speed of Structure Learning Algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2019.10.003.

M. Scutari, C. Vitolo, and A. Tucker. Learning Bayesian networks from big data with greedy search: computational complexity and efficient implementation. *Statistics and Computing*, pages 1–14, 2018.

T. Silander, P. Kontkanen, and P. Myllymaki. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. UAI'07, page 360–367, Arlington, Virginia, USA, 2007. AUAI Press. ISBN 0974903930.

R. Sokal and F. Rohlf. Biometry: The Principles and Practice of Statistics in Biological Research. W. H. Freeman, 1981. ISBN 9780716712541.

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

P. Spirtes, C. Glymour, and R. Scheines. Causation, Prediction, and Search, 2nd Edition, volume 1 of MIT Press Books. The MIT Press, August 2001.

C. Squires. causaldag Python library, 2018.

URL https://github.com/uhlerlab/causaldag.

H. Steck. Learning the Bayesian network structure: Dirichlet prior vs data. In D. A. McAllester and P. Myllymaki, editors, UAI 2008, *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, Helsinki, Finland, July 9-12, 2008, pages 511–518. AUAI Press, 2008.

P. Suter, J. Kuipers, G. Moffa, and N. Beerenwinkel. Bayesian structure learning and sampling of Bayesian networks with the r package BiDAG. *Journal of Statistical Software*, 105(1):1–31, 2023. doi: 10.18637/jss.v105.i09.

S. Thornley, R. Marshall, R. Jackson, D. Gentles, N. Dalbeth, S. Crengle, A. Kerr, and S. Wells. Is serum urate causally associated with incident cardiovascular disease? *Rheumatology (Oxford, England)*, 52, 10 2012. doi: 10.1093/rheumatology/kes269.

J. Tian and J. Pearl. Causal discovery from changes: a Bayesian approach. In *Conference on Uncertainty in Artificial Intelligence*, 2001.

S. Triantafillou and I. Tsamardinos. Constraint-based causal discovery from multiple interventions over overlapping variable sets. *Journal of Machine Learning Research*, 16(66):2147–2205, 2015.

S. Triantafillou and I. Tsamardinos. Score-based vs constraint-based causal learning in the presence of confounders. In *CFA@UAI*, 2016.

S. Triantafillou, K. Tsirlis, V. Lagani, and I. Tsamardinos. MATLAB library, 2019. URL https://github.com/mensxmachina/M3HC.

I. Tsamardinos, C. F. Aliferis, and A. 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*, KDD '03, page 673–678, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137370. doi: 10.1145/956750.956838.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 10, 2006. doi: 10.1007/s10994-006-6889-7.

K. Tsirlis, V. Lagani, S. Triantafillou, and I. Tsamardinos. On scoring maximal ancestral graphs with the max–min hill climbing algorithm. *International Journal of Approximate Reasoning*, 102, 08 2018. doi: 10.1016/j.ijar.2018.08.002.

M. Ueno. Robust learning Bayesian networks for prior belief. In F. G. Cozman and A. Pfeffer, editors, UAI 2011, *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, Barcelona, Spain, July 14-17, 2011, pages 698–707. AUAI Press, 2011.

H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1105–1112, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553515.

Y. Wang, L. Solus, K. D. Yang, and C. Uhler. Permutation-based causal inference algorithms with interventions, In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5824–5833, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

C. Wongchokprasitti. R-causal R Wrapper for Tetrad Library, v1.1.1, 2019. URL https://github.com/bd2kccd/r-causal.

K. Yamasaki, A. Gozolchiani, and S. Havlin. Climate networks around the globe are significantly affected by el nino. *Phys. Rev*. Lett., 100:228501, Jun 2008. doi: 10.1103/PhysRevLett.100.228501.

J. Zhang. Causal inference and reasoning in causally insufficient systems. Technical report, 2006.

J. Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172, 11 2008. doi: 10.1016/j.artint.2008.08.001.

X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing. DAG with no tears: Continuous optimization for structure learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018