Bayesian network structure learning in the presence of data noise

Yang Liu
School of Electronic Engineering and Computer Science
Queen Mary University of London

2023

Abstract

A Bayesian Network (BN) is a type of a probabilistic graphical model that captures conditional and marginal independencies between variables. These models are generally represented by a Directed Acyclic Graph (DAG), which is composed by nodes and arcs. The nodes represent variables and the absence of arcs represent conditional or marginal independencies. When BNs are applied to real-world problems, the structure of these models is often assumed to be causal (often referred to as a causal BN), and is often constructed from either expert knowledge and Randomised Controlled Trials (RCTs). However, these two approaches can be time-consuming and expensive, and it might not always be possible or ethical to perform RCTs. As a result, structure learning algorithms that recover graphical structures from observational data, which in turn could be used to inform causal structures, have received increasing attention over the past few decades.

To be able to guarantee the correctness of a structure learnt from data, a structure learning algorithm must rely on assumptions that may not hold in practice. One such crucial and commonly used assumption is that the observed data are independently and identically sampled from the underlying distribution, such that all statistical quantities of the distribution can be recovered with no bias from the observed data when sample size goes to infinite. While such assumptions are often needed to be able to devise theoretical guarantees, the impact of violating these assumptions when working with real data tends to be overlooked. Empirical investigations show that structure learning algorithms perform considerably worse on noisy data that violate many of their theoretical assumptions, relative to how they perform on clean synthetic data that do not violate any of their data-generating assumptions. However, there has been limited research on how to deal with these problems effectively and efficiently. This thesis investigates this research direction and primarily focuses on improving structure learning in the presence of measurement error and systematic missing data, which are two of the most common types of data noise present in real data sets.

Acknowledgements

First and foremost, I am grateful to my primary supervisor Dr Anthony Constantinou who always provided me with valuable and professional guidance to my research. He always supported me in difficult and challenging times and motivated me throughout. His immense knowledge and rich academic experience assisted and inspired me in all phases of my academic research and daily life. With his professional supervision, I could figure out research problems, deal with the challenges, and contribute to many publications. I would like to extend my sincere thanks to Prof Norman Fenton, Prof Martin Neil and Prof William Marsh for their invaluable comments and discussions at every stage of my PhD viva. Special thanks to my lab colleagues Kiattikun Chobtham, Dr Neville Kenneth Kitson and Dr Zhigao Guo for their insightful comments and kind help. I hope our friendships are long and amount to more collaborations in the future. I would like to thank Queen Mary University of London for giving me the opportunity to complete my PhD studies. Finally, I am also deeply grateful to my family and friends for their support, appreciation, encouragement and keen interest in my academic achievements.

Contents

1 Introduction **14**

1.1 Motivation . . . 14

1.2 Thesis structure and contributions . . . 15

1.3 Paper contributions . . . 16

2 Background **18**

2.1 Bayesian Networks . . . 18

2.2 Structure Learning . . . 21

2.2.1 Evaluation metrics . . . 22

2.2.2 Score-based structure learning . . . 23

2.2.3 Constraint-based structure learning . . . 28

2.2.4 Hybrid structure learning . . . 35

2.2.5 Structure learning for functional causal models . . . 36

3 The impact of data noise on structure learning, and learning structures from noisy data with model averaging **39**

3.1 Introduction . . . 39

3.2 Impact of data noise . . . 40

3.2.1 Case studies and synthetic data . . . 40

3.2.2 The investigated structure learning algorithms . . . 41

3.2.3 Evaluation . . . 42

3.2.4 Results and discussion . . . 43

3.3 Handling data noise by model averaging . . . 46

3.3.1 The Model Averaging Hill-Climbing algorithm . . . 46

3.3.2 Evaluation, results and discussion . . . 48

3.4 Concluding remarks . . . 50

4 Improving Structure Learning under Measurement Error **52**

4.1 Introduction . . . 52

4.2 Relevant works . . . 53

4.3 Preliminaries . . . 53

4.4 Impact of measurement error on structure learning . . . 55

4.5 The Spurious Edge Detection (SED) algorithm . . . 59

4.6 Empirical evaluation . . . 67

4.7 Conclusion . . . 74

5 Improving greedy search structure learning in the presence of systematic missing values 76

5.1 Introduction 76
5.2 Relevant works 77
5.3 Preliminaries 78
5.3.1 Hill Climbing algorithm 78
5.3.2 Missing data 78
5.4 Handling systematic missing data with Hill-Climbing 80
5.4.1 Hill-Climbing with pairwise deletion 80
5.4.2 Hill-Climbing with Inverse Probability Weighting 83
5.4.3 Hill-Climbing with adaptive Inverse Probability Weighting 87
5.5 Experiments 90
5.5.1 Generating synthetic data and missingness 90
5.5.2 Evaluation metrics 91
5.5.3 Results when the true DAG is sparse 91
5.5.4 Results when the true DAG is dense 93
5.5.5 Results when the true DAG is a real-world network 94
5.6 Conclusion 96

6 Using Markov blanket to improve data imputation in the presence of systematic missingness 99

6.1 Introduction and relevant works 99
6.2 Preliminaries 101
6.3 Markov Blanket based feature selection for imputation 101
6.4 Experiments 106
6.4.1 Synthetic case studies based on real-world BNs 106
6.4.2 Evaluation process 107
6.4.3 Results 107
6.4.4 Real-world case study 110
6.5 Conclusion 111

7 Conclusions and directions for future work 113

7.1 Conclusion 113
7.2 Future research 114

List of Figures

2.1 An example of a DAG that is unfaithful to its probability distribution. V_1 and V_3 are independent since $P(V_1, V_3) = P(V_1) \cdot P(V_3)$ for any value of V_1 and V_3; however, V_1 and V_3 are not d-separated by any variable set in the DAG. 19

2.2 (a) A hypothetical DAG and (b) its corresponding CPDAG. 20

2.3 Lattice explored by dynamic programming. Each directed edge explores a sink node and searches its optimal parent set. Each path from top to bottom represents a node ordering. 27

2.4 The output of fGES when the true DAG contains unobserved common cause. . . . 36

3.1 The overall decrease in accuracy over all algorithms for each noisy experiment, with respect to noise-free experiment. 46

3.2 The structure learning accuracy of the seven other algorithms, relative to the accuracy of MAHC. A negative green score indicates that a particular algorithm performed worse than MAHC, while a positive red score indicates that a specific algorithm outperformed MAHC. 49

4.1 A hypothetical graph that illustrates the relationship between the error-free variables $\boldsymbol{V}$ and the corresponding noisy variables $\boldsymbol{V}^*$ given the Independence rule, where a noisy variable V_i^* becomes independent of other variables in $\mathcal{G}^*$ given V_i. 54

4.2 An example to illustrate how measurement error is represented for a discrete variable V_i with four states. 54

4.3 Modelling the presence of measurement error on the two different causal equivalence classes. Case (a) represents the common-effect class, where V_1 and V_2 are d-connected conditional on either V_3 or V_3^*, and (b) represents the causal-chain class where V_1 and V_2 are d-separated conditional on V_3, whereas they are d-connected conditional on V_3^* (this also holds for the causal class of common-cause). 56

4.4 (a) A noisy graph that contains three error-free variables V_1, V_2 and V_3, and a noisy variable V_3^*. (b) The learnt graph that entails the same dependence and independence relationships derived from the observed variables. 57

4.5 (a) The true Asia network. (b) The CPDAG learnt by PC-stable given the error-free synthetic data set. (c) The CPDAG learnt by PC-stable given the same synthetic data set but with 5% measurement error on variable *bronc*. 58

4.6 The average number of 3-vertex cliques in the ground truth graphs, the graphs learnt from error-free data sets, and the graphs learnt from observed data sets with 10% measurement error on each variable. 59

4.7 An example of a graph that contains 3-vertex cliques 60

4.8 The three reconstructed graphs for clique $\{bronc, dysp, smoke\}$, based on the learnt graph in Figure 4.5c. Dotted nodes represent possible hidden error-free parents of the suspected noisy node under assessment. 61

4.9 (a) A noisy graph with four variables that incorporate measurement error. (b) The learnt graph with respect to (a). (c) The reconstructed graph that contains the conditional independence $V_1 \perp\!\!\!\perp V_3 \mid \{V_2, V_4\}$. 63

4.10 Left: The Asia graph learnt by the HC algorithm from a synthetic data set with 10,000 samples and 5% measurement error on each observed variable. Right: the candidate spurious edge-nodes pairs CSE of the learnt graph. 65

4.11 The average precision, recall and F_1 scores of the graphs produced by the five algorithms, where solid lines represent the results before SED modifications, dashed lines represent the results after SED modifications, red lines the results based on error-free data, green lines the results based on noisy data with 10% error rate, and blue lines the results based on noisy data with 20% error rate. 69

4.12 The average number of false and true 3-vertex cliques produced by the learnt graphs learnt from error-free data sets, the learnt graphs learnt from noisy data sets and the modified graphs learnt from noisy data sets. 70

4.13 The average re-scaled SHD scores and its three components produced by the four algorithms, where solid lines represent the results before SED modifies the graphs, dashed lines represent the results after SED modifies the graphs, green lines the results based on noisy data with 10% error rate, and blue lines the results based on noisy data with 20% error rate. 72

4.14 Average execution time needed to produce the learnt and modified graphs for the specified algorithms, across different error rate and sample size combinations. The execution time of SED is based on the time it takes to modify graphs. 74

4.15 An example where SED fails to recover the true CPDAG. 75

5.1 The three possible m-graphs assuming three observed variables with structure $V_1 \rightarrow V_2 \rightarrow V_3$. Shaded nodes represent partially observed variables. 80

5.2 A hill-climbing illustration of the DAG considered in Example 5.4.2, discussed in the main text. Shaded nodes represent partially observed variables. 84

5.3 Example of a searching step in HC-pairwise/HC-IPW. 88

5.4 Average F_1 and re-scaled SHD scores learnt by HC-pairwise, HC-IPW, HC-aIPW and Structural EM for sparse networks, under different assumptions of missingness and sample sizes. Each score represents the average score over 50 CPDAGs. Note the scores of HC-complete are based on complete data for benchmarking purposes; i.e., the same scores are superimposed in all three missingness cases as a dashed line. 92

5.5 Average F_1 and re-scaled SHD scores learnt by HC-pairwise, HC-IPW, HC-aIPW and Structural EM for dense networks, under different assumptions of missingness and sample sizes. Each score represents the average score over 50 CPDAGs. Note the scores of HC-complete are based on complete data for benchmarking purposes; i.e., the same scores are superimposed in all three missingness cases as a dashed line. 95

5.6 Average F_1 and re-scaled SHD scores learnt by HC-pairwise, HC-IPW, HC-aIPW and Structural EM for real-world networks, under different assumptions of missingness and sample sizes. Each score represents the average score over 50 CPDAGs. Note the scores of HC-complete are based on complete data for benchmarking purposes; i.e., the same scores are superimposed in all three missingness cases as a dashed line. 97

6.1 An example of m-graph. Shaded nodes represent partially observed variables. . . . 102

6.2 A hypothetical m-graph used to described the implications of violating Assumption 5.3.1 of MBFS. Shaded nodes represent partially observed variables. 105

6.3 Average RMSE between complete and imputed data produced by the different algorithms. A Lower score represents better performance. 108

6.4 Average F_1 scores of the graphs learnt by GES from data imputed by the different algorithms. A higher F_1 score represents better performance. The dashed line represents the performance of GES when applied to complete data. 109

6.5 Average execution time of MF and MBMF under different sample sizes, mechanisms of missingness, and rates of missingness. 110

List of Tables

3.1 The description of the six case studies. 40
3.2 Experiment codes used to represent the 16 specified configurations of noisy data investigated. 42
3.3 The 15 structure learning algorithms investigated, categorised by learning class. . . 42
3.4 The penalty weights assumed by the F_1 and SHD metrics. 43
3.5 Average and overall ranked performance of the algorithms over all case studies and sample sizes in experiment N (i.e., no noise), where the average rank is presented with its standard deviation. 44
3.6 Average and overall ranked performance for each algorithm over all case studies and sample sizes across all noisy experiments. The numbers in brackets represent the difference in performance with respect to the clean experiments N. Green and red text indicate increased and decreased relative ranked performance respectively. . . 45

4.1 The steps of the SED algorithm in modifying the Asia graph learnt by HC from synthetic data of sample size 10,000 with 5% measurement error on all variables. The red edges represent the edges classified as spurious in each iteration, whereas the blue edges represent the edges being pruned (i.e., no longer being considered as candidate spurious edges) after each iteration. 67
4.2 The number and percentage of modified graphs that are better, equal or worse than the learnt graphs in terms of graphical accuracy, for each algorithm, error-rate, and over different evaluation metrics. 73

5.1 Examples of necessary variables for each edge operation in HC, which we define as the variables with different parent-sets between the current best and neighbouring graphs, plus the parents that make up those parent-sets. 81
5.2 The properties of the six real-world BNs. 90
5.3 Mean and standard deviation of F_1 scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for sparse networks, under the different assumptions of missingness and sample sizes. 92
5.4 Mean and standard deviation of re-scaled SHD scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for sparse networks, under the different assumptions of missingness and sample sizes. 93
5.5 Mean and standard deviation of execution times produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for sparse networks and relative to HC when applied to complete data, under the different assumptions of missingness and sample sizes. 94

5.6 Mean and standard deviation of F_1 scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for dense networks, under the different assumptions of missingness and sample sizes. 95
5.7 Mean and standard deviation of re-scaled SHD scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for dense networks, under the different assumptions of missingness and sample sizes. 96
5.8 Mean and standard deviation of F_1 scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for real-world networks, under the different assumptions of missingness and sample sizes. 97
5.9 Mean and standard deviation of re-scaled SHD scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for real-world networks, under the different assumptions of missingness and sample sizes. 98

6.1 Summary of the real-world BNs used in Subsection 6.4.1. 106
6.2 Summary of the real-world data sets used in Subsection 6.4.4. 110
6.3 Average RMSE and PFC scores, and their standard deviations, for different imputation algorithms and real-world data combinations. Lower RMSE and PFC scores represent better performance. 111

List of abbreviations

BN	Bayesian Network
DAG	Directed Acyclic Graph
MEC	Markov Equivalence Class
CPDAG	Completed Partially Directed Acyclic Graph
RCTs	Randomised Controlled Trials
ML	Machine Learning
CI	Conditional Independence
FCMs	Functional Causal Models
SHD	Structural Hamming Distance
BSF	Balanced Scoring Function
KL	Kullback-Leibler
SED	Spurious Edge Detection
BIC	Bayesian Information Criterion
BDe	Bayesian Dirichlet equivalent
BDeu	Bayesian Dirichlet equivalent uniform
qNML	quotient Normalised Maximum Likelihood
LL	Log-likelihood
NML	Normalised Maximum Likelihood
HC	Hill-Climbing
GES	Greedy Equivalence Search
fGES	fast Greedy Equivalence Search
OBS	Ordering Based Search
ASOBS	Acyclic Selection OBS
WINASOBS	Window Acyclic Selection OBS
GOBNILP	Globally Optimal Bayesian Network learning using Integer Linear Programming
KCI-test	Kernel-based Conditional Independence test
RKHS	Reproducing Kernel Hilbert Space
FCI	Fast Causal Inference
RFCI	Really Fast Causal Inference
POIPG	Partially Orientated Inducing Path Graph
ASP	Answer Set Programming
GS	Grow-Shrink
MB	Markov Blanket
IAMB	Incremental Association Markov Blanket

MMHC	Max-Min Hill-Climbing
MMPC	Max-Min Parents Children
H2PC	Hybrid HPC
HPC	Hybrid Parents and Children
GFCI	Greedy Fast Causal Inference
BSC	Bayesian Scoring of Constraints
LiNGAM	Linear Non-Gaussian Acyclic Model
ICA	Independent Component Analysis
ANM	Additive Noise Model
GDS	Greedy DAG Search
HSIC	Hilbert Schmidt Independence Criterion
PNL	Post-NonLinear
TIN	Transformed Independent Noise
IN	Independent Noise
GIN	Generalised Independent Noise
SEMs	Structural Equation Models
FCI	Fast Causal Inference
ILP	Integer Linear Programming
GFCI	Greedy FCI
MAG	Maximal Ancestral Graph
PAG	Partial Acyclic Graph
FP	False Positive
CSE	Candidate Spurious Edge-nodes pair
EM	Expectation-Maximisation
MLE	Maximum Likelihood Estimation
MCAR	Missing Completely At Random
MAR	Missing At Random
MNAR	Missing Not At Random
NAL	Node-Averaging Likelihood
IPW	Inverse Probability Weighting
m-graph	missingness graph
MVNI	Multivariate Normal Imputation
MICE	Multiple Imputation by Chained Equations
ELM	Extreme Learning Machine
ML	Machine Learning
kNN	k-Nearest Neighbour
RF	Random Forest
GAIN	Generative Adversarial Imputation Nets
MI	Mutual Information
PCA	Principle Component Analysis
MBMF	Markov Blanket MissForest
MF	MissForest
CMB	Candidate Markov Blanket
MBFS	Markov Blanket-based Feature Selection
RMSE	Root Mean Squared Error

PFC	Proportion of Falsely Classified entries

Chapter 1

Introduction

1.1 Motivation

Associational *Machine Learning* (ML), and particularly deep learning, are found to be highly effective in areas with access to big data, such as in computer vision [Ho et al., 2020], natural language processing [Stiennon et al., 2020], and sound information processing [Purwins et al., 2019]. However, the success of deep learning has also revealed some of its drawbacks. A major drawback is that deep learning solutions are widely considered to be black-box solutions, in that they offer little interpretability in terms of how they arrive at a prediction outcome or explaining a recommended decision [Geiger et al., 2021]. While interpretability may not be necessary in all problems, it is necessary in many critical areas where decisions require justification. This means that, in certain domains, the ability to elucidate the relationship between outcomes and factors outweighs the importance of prediction accuracy. If we take healthcare as an example, understanding the causal relationships between symptoms, diseases, and treatments is vital for accurate diagnosis, effective treatment planning, and patient care. Uncovering accurate causal links would enable healthcare professionals to make informed decisions about available interventions, optimise treatment protocols, minimise potential risks or adverse effects, and formulate preventive measures to improve patient outcomes.

A causal *Bayesian Network* (BN) is a probabilistic graphical model that provides a formal framework for modelling and reasoning about causal relationships between variables. It allows for a systematic representation of cause-and-effect relationships, enabling insights into how changes in one variable propagate through the network affecting other variables, which enables the simulation of hypothetical interventions to estimate their effect. This capability is particularly valuable in domains where interpretability and transparency are crucial, such as in healthcare [McLachlan et al., 2020, Stallman et al., 2021], government policy [Hakhverdian, 2012, Wang et al., 2022], economics [Kragt et al., 2009, Tsagris, 2021], and education [Xenos, 2004, Almond et al., 2015].

An issue with utilising causal BNs is that they tend to require higher effort to develop compared to associational ML models. Depending on the application area, causal BNs may also require access to human expertise as part of their validation process in terms of capturing causal relationships. Historically, the construction of causal models relied on expert knowledge or randomised experiments and was associated with high costs, time constraints, and often feasibility challenges. Causal knowledge elicitation was common in the past where access to data were sparse. However, we nowadays have access to big and detailed data which diminishes the need for expert knowledge and encourages the development of objective causal models from data. This has led to the emergence of structure learning (or causal discovery) algorithms.

An important issue with structure learning algorithms is that it can be particularly difficult to discover accurate cause-and-effect relationships from real-world data. These algorithms are known to perform considerably better on clean synthetic experiments, where the input data satisfy the underlying assumptions of the algorithms, compared to how they perform on real-world data sets that tend to be noisy and to violate many of their underlying assumptions about the input data [Scheines and Ramsey, 2016, Constantinou et al., 2021]. The majority of structure learning algorithms do not consider the presence of noise in the data, and they tend to be evaluated with clean synthetic data sets, which are i.i.d. samples from the underlying distribution. This practice is known to overestimate their performance in real-world scenarios, which hinders the application of structure learning algorithms to real-world problems.

Two common types of data noise in real-world data sets are measurement error and missing data. Measurement error refers to the discrepancy between the true value of a variable and its measured or observed value. It arises due to various factors such as systematic biases in the measurement devices, mistakes in recording observations, and changes in environmental conditions. These errors can occur in both experimental and observational data collection. Measurement errors can have significant implications in scientific research and data analysis, since they could lead to biased estimates, incorrect conclusions, and reduced accuracy in statistical models. Therefore, it is essential to understand and account for measurement error to ensure the reliability and validity of research findings.

Missing data is another common type of data noise which refers to the absence or unavailability of certain values in a data set. It is a common issue that arises in data collection and can occur due to various reasons. For example, in surveys or questionnaires participants may not to answer certain questions, resulting in missing values. Besides, certain measurements may not always be applicable or feasible for all subjects or variables. For example, laboratory tests may not be applied to all participants, resulting in missing values. Ignoring the reasons behind why some values might be missing would introduce bias in observed values and distort the results of data analysis. Dealing with missing values is vital for ensuring data integrity, avoiding biased results, maximising sample size, and maintaining statistical assumptions.

The primary goal of this thesis is to enhance the effectiveness of structure learning when applied to real-world data. It aims to achieve this objective by improving the performance of structure learning algorithms in the face of data noise, with a particular focus on measurement error and systematic data missingness.

1.2 Thesis structure and contributions

This thesis investigates and provides improved solutions to the problem of recovering causal or conditional independence relationships from noisy, rather than noise-free, data that better reflect real data. Chapter 2 provides the necessary preliminary and background information, and Chapter 3 presents an empirical evaluation of structure learning in the presence of different types of data noise, aiming to provide a clear picture of the issues dealt with in the thesis. The subsequent chapters present new methods that provide improvements when learning primarily from data with measurement error and missing values. The thesis is structured as follows:

Chapter 2 provides background information on Bayesian networks and structure learning algorithms, and reviews important relevant works published in the academic literature. Some of the material presented in this chapter is based on a paper published in the *Artificial Intelligence Review*, where I contributed as a co-author [Kitson et al., 2023].

Chapter 3 investigates the impact of different types of data noise on structure learning. It also explores model-averaging strategies that could be combined with structure learning to improve learning in the presence of data noise. The main contributions of this chapter are a) a large-scale empirical evaluation of structure learning algorithms with noisy data, and b) a new algorithm that leverages model averaging to reduce sensitivity to data noise. The material presented in this chapter is based on two papers published in the *International Journal of Approximate Reasoning (IJAR)*, where I contributed as a co-author [Constantinou et al., 2021, 2022].

Chapter 4 focuses on structure learning in the presence of measurement error when the input data are discrete. The main contribution of this chapter is a new algorithm which acts as an add-on learning process for other structure learning algorithms, to identify and remove spurious edges that are potentially caused due to measurement error. The material presented in this chapter comes from a paper published in the *Journal of Machine Learning Research (JMLR)*, in which I am the leading author [Liu et al., 2022].

Chapter 5 focuses on structure learning from data that contain systematic missing values. The main contribution of this chapter is a novel solution that adopts the *Inverse Probability Weighting* (IPW) to maximally leverage the data samples used to train a greedy search structure learning algorithm, and successfully reduces or eliminates potential bias caused by systematic missingness. The material presented in this chapter comes from a paper published in the *Machine Learning* journal, in which I am the leading author [Liu and Constantinou, 2022].

Chapter 6 focuses on the problem of imputing missing data under all types of missingness. The main contribution of this chapter is a new imputation algorithm that employs the Markov blanket discovery algorithm *Grow-Shrink* (GS) to efficiently and effectively find the variable set that makes each variable in the data independent of the other variables given its Markov blanket, and then applies MissForest to each variable by only considering its Markov blanket to impute data. This process was found to improve imputation accuracy under all types of missingness. The material presented in this chapter comes from a paper presented at the *International Conference on Learning Representations (ICLR)*, in which I am the leading author [Liu and Constantinou, 2023].

Chapter 7 provides a summary of the findings and conclusions of this thesis, as well as outlines several potential directions for future research. This chapter also highlights some open problems in causal structure learning, some of which are based on a paper that is currently under review, in which I served as co-author [Constantinou et al., 2023].

1.3 Paper contributions

Part of the work presented in this thesis comes from the following publications:

1. **Liu, Y**., Constantinou, A.C. and Guo, Z., 2022. Improving Bayesian network structure learning in the presence of measurement error. *Journal of Machine Learning Research*, 23(324), pp.1-28.

2. **Liu, Y**. and Constantinou, A.C., 2022. Greedy structure learning from data that contain systematic missing values. *Machine Learning*, 111(10), pp.3867-3896.

3. **Liu, Y**. and Constantinou, A., 2023, February. Improving the imputation of missing data with Markov Blanket discovery. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR-2023).*

4. Constantinou, A.C., **Liu, Y**., Chobtham, K., Guo, Z. and Kitson, N.K., 2021. Large-scale empirical validation of Bayesian Network structure learning algorithms with noisy data. *International Journal of Approximate Reasoning*, 131, pp.151-188.

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

6. Kitson, N.K., Constantinou, A.C., Guo, Z., **Liu, Y**. and Chobtham, K., 2023. A survey of Bayesian Network structure learning. *Artificial Intelligence Review*, pp.1-94.

7. Constantinou, A.C., Kitson, N.K., **Liu, Y**., Chobtham, K., Hashemzadeh, A., Nanavati, P.A., Mbuvha, R. and Petrungaro, B., 2023. Open problems in causal structure learning: A case study of COVID-19 in the UK. *arXiv preprint arXiv:2305.03859.*

My personal contributions are as follows:

- Publications 1, 2 and 3: I led the development and the implementation of the new algorithms, the analysis and preparation of the results, as well as the writing of the first draft.

- Publications 4 and 7: I contributed to the implementation of the experiments and editing and reviewing the papers.

- Publication 5: I contributed to editing and reviewing the paper.

- Publication 6: I contributed to writing parts of the draft paper as well as reviewing other parts of the paper.

Chapter 2

Background

2.1 Bayesian Networks

A *Bayesian Network* (BN) is a probabilistic graphical model that captures conditional or causal independence relationships between variables, and is typically represented by a *Directed Acyclic Graph* (DAG). Each node in a BN graph represents a random variable, and the absence of an edge represents conditional or marginal independence. The conditional probability distribution associated with each node specify the probability distribution of that variable given the values of its parent variables. A causal BN assumes that the edges represent causal relationships.

A BN can be formally defined as a pair $\langle \mathcal{G}, P \rangle$ which consists a DAG $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ and a joint probability distribution P defined over $\boldsymbol{V}$, where $\boldsymbol{V} = \{V_1, \ldots, V_n\}$ represents a set of random variables and $\boldsymbol{E}$ represents a set of directed edges between pairs of variables. In a DAG $\mathcal{G}$, a variable V_i is the *parent* of V_j if there is a directed edge from V_i to V_j, and V_j is called the *child* of V_i. A variable is a *neighbour* of V_i if it is either a parent or a child of V_i. The *indegree* of a variable is the number of its parents and the *degree* of a variable is the number of its neighbours. A *path* is a sequence of distinct nodes in which every two consecutive nodes on the path are neighbours in the graph. A *directed path* denotes that every variable in the sequence is the parent of the subsequent variable. If there is a directed path from V_i to V_j, then V_j is a *descendant* of V_i, and V_i is an *ancestor* of V_j. Given a DAG G, a node V_i is a *collider* in a path p if the neighbours of V_i in p are both the parent of V_i. If the two parents are not neighbours, then such a node will be referred as an *unshielded collider* or *v-structure*, otherwise, it will be referred as a *shielded collider.* A triple of variables $\langle V_i, V_j, V_k \rangle$ is called *unshielded triple* in graph $\mathcal{G}$ if V_i, V_j and V_j, V_k are neighbours in $\mathcal{G}$, but V_i and V_k are not.

If a DAG is interpreted causally, it is called as a *causal DAG* in which a parent of a variable V_i is assumed to be a direct *cause* of V_i. The conditional independence and dependence relationships in a graph are defined by the following *d-separation* and *d-connection* criteria [Spirtes et al., 2000].

Definition 2.1.1 (D-separation)**.** *Given a graph $\mathcal{G}$, two variables V_i and V_j are d-separated given a variable set $\boldsymbol{S}$ if there is no path p between V_i and V_j such that (i) every collider in p belongs either to $\boldsymbol{S}$ or an ancestor of a node in $\boldsymbol{S}$, and (ii) no other variables on p are in $\boldsymbol{S}$.*

Definition 2.1.2 (D-connection)**.** *Given a graph $\mathcal{G}$, two variables V_i and V_j are d-connected given a variable set $\boldsymbol{S}$ if they are not d-separated by $\boldsymbol{S}$.*

If set $\boldsymbol{S}$ d-separates two variables V_i and V_j in a graph $\mathcal{G}$, then it is referred to as the *separating*

	$V_1 = 0$		$V_1 = 1$	
$P(V_3 \mid V_1, V_2)$	$V_2 = 0$	$V_2 = 1$	$V_2 = 0$	$V_2 = 1$
$V_3 = 0$	0.7	0.3	0.4	0.2
$V_3 = 1$	0.3	0.7	0.6	0.8

	$P(V_1)$
$V_1 = 0$	0.3
$V_1 = 1$	0.7

$$\begin{tikzcd}
V_1 \arrow[rr] \arrow[dr] & & V_3 \\
& V_2 \arrow[ur] &
\end{tikzcd}$$

$P(V_2 \mid V_1)$	$V_1 = 0$	$V_1 = 1$
$V_2 = 0$	0.2	0.9
$V_2 = 1$	0.8	0.1

Figure 2.1: An example of a DAG that is unfaithful to its probability distribution. V_1 and V_3 are independent since $P(V_1, V_3) = P(V_1) \cdot P(V_3)$ for any value of V_1 and V_3; however, V_1 and V_3 are not d-separated by any variable set in the DAG.

set of V_i and V_j. Note that the above definition is made with reference to a single variable. If there are two variable sets $\boldsymbol{U}$ and $\boldsymbol{V}$ such that every variable in $\boldsymbol{U}$ is d-separated or d-connected with every variable in $\boldsymbol{V}$ given a variable set $\boldsymbol{S}$, then $\boldsymbol{U}$ and $\boldsymbol{V}$ are also d-separated or d-connected given $\boldsymbol{S}$.

To associate the conditional independencies derived from distributions with those entailed by a DAG structure, we need to adopt a set of assumptions. One such fundamental assumption is *Markov assumption*:

Assumption 2.1.1 (The Markov assumption)**.** *Given a DAG G over a variable set $\boldsymbol{V}$, every variable in $\boldsymbol{V}$ is independent of its non-parental non-descendants conditional on its parents.*

A distribution $P(\boldsymbol{V})$ that follows the Markov assumption can be factorised as a product of a series of conditional probabilities as:

$$P(\boldsymbol{V}) = \prod_{V_i \in \boldsymbol{V}} P(V_i \mid \boldsymbol{Pa}_i), \tag{2.1}$$

where $\boldsymbol{Pa}_i$ is the parents of V_i.

Given the Markov assumption, any probability distribution represented by a DAG can have its conditional independences obtained by applying the d-separation criterion to the relevant DAG. However, the Markov assumption does not prohibit probability distributions from having additional conditional independences. For example, Figure 2.1 presents an example in which V_1 and V_3 are independent, i.e., $P(V_1, V_3) = P(V_1) \cdot P(V_3)$ for any value of V_1 and V_3, but are not d-separated by any variable set in the DAG. In this example, the effect of the direct path from V_1 to V_3 and the effect of the indirect path from V_1 to V_3 via V_2 cancel out the dependency between V_1 and V_3. Under such a scenario, the probability distribution is called *unfaithful* to its DAG structure. The *faithfulness assumption* assumes that the input data contain no such cases.

Assumption 2.1.2 (The faithfulness assumption)**.** *Given a DAG G over a variable set $\boldsymbol{V}$, a probability distribution $P(\boldsymbol{V})$ is faithful to G if and only if the conditional independence relationships in $P(\boldsymbol{V})$ are exactly the same as the independence relationships inferred by d-separation criterion from G.*

The faithfulness assumption implies that all marginal or conditional independences in the distribution are faithfully represented by the graphical structure via d-separation, rather than by

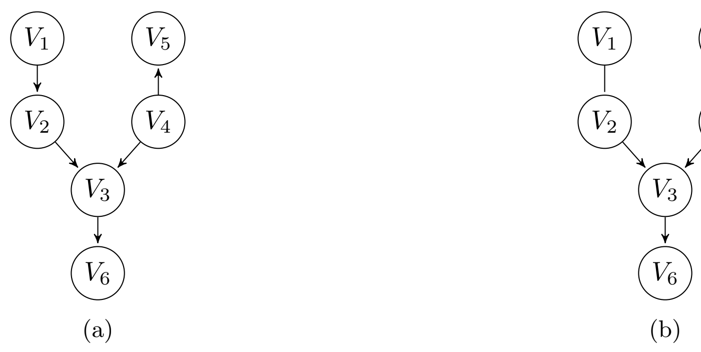


Figure 2.2: (a) A hypothetical DAG and (b) its corresponding CPDAG.

chance, as illustrated in Figure 2.1. Because the set of unfaithful distributions to a given DAG has Lebesgue measure zero [Uhler et al., 2013], it can be argued that the faithfulness assumption is a relatively weak and a reasonable assumption to make since almost all observed distributions are expected to satisfy it. However, although the possibility of an exact unfaithful distribution is infinitely small, the possibility of near unfaithful distributions is not [Weinberger, 2018]. In real-world scenarios, near unfaithful distributions can cause the same problem as exact unfaithful distributions.

Given the Markov and faithfulness assumptions, it is possible that multiple DAGs contain the same set of d-separation relationships. Such set of DAGs form a *Markov Equivalence Class* (MEC) which corresponds to a unique *Completed Partially Directed Acyclic Graph* (CPDAG) that contains both directed and undirected edges. We denote a fully connected undirected graph as $\mathcal{G}_c$, and an empty graph as $\mathcal{G}_\emptyset$. A directed edge in a CPDAG indicates that all Markov equivalent DAGs have this directed edge in their structure, whereas an undirected edge in a CPDAG indicates that this edge is present in all of the Markov equivalent DAGs but has inconsistent orientation. A CPDAG can be obtained from a DAG by a) preserving all its v-structures, b) preserving all the directed edges that would create a cycle or a new v-structure if reversed, and c) converting the residual directed edges to undirected edges. Figure 2.2 presents a hypothetical DAG along with its corresponding CPDAG. A more detailed conversion solution is described by Meek [1995].

Another commonly used assumption is *causal sufficiency*, which assumes that there are no unobserved common causes and selection bias in the causal DAG.

Assumption 2.1.3 (Causal sufficiency)**.** *There are no unmeasured variables acting as a common cause of any two or more observed variables.*

When the causal sufficiency assumption is violated, it implies that there is at least one unobserved common cause or selection bias in the input data. This would make it impossible to recover a DAG structure that exactly expresses the causal relationships between the observed variables. In this case, a *Maximal Ancestral Graph* (MAG) can be used to express all possible causal relationships between observed variables. A MAG is an extension of a DAG which may contain both directed and bi-directed edges between observed variables, where the directed edges represent direct or ancestral relationships between connected nodes, and the bi-directed edges indicate the presence of at least one latent confounder between the connected nodes. A *Partial Ancestral Graph* (PAG) represents the Markov equivalent class of a set of MAGs, and is analogous to the relationship between a CPDAG and its corresponding Markov equivalent DAGs. In addition to the tail and arrowhead, a PAG may contain another endpoint for edges marked as 'o'. An edge

$V_1 o \rightarrow V_2$ implies that both $V_1 \leftrightarrow V_2$ and $V_1 \rightarrow V_2$ are present in the Markov equivalent MAGs.

2.2 Structure Learning

BN models are constantly being published in the literature with application to diverse areas. The structure of these models is often derived from a combination of data, expert knowledge, and results from *Randomised Controlled Trails* (RCTs). However, knowledge elicitation can be expensive or time-consuming, and RCTs are not always possible or ethical. Therefore, learning causal BN structure from data, sometimes also referred to as *causal discovery*, has received increasing attention during the past few decades. A formal definition of BN structure learning is given as follows:

Definition 2.2.1 (Structure learning). *Given a data set D sampled from an unknown distribution P, structure learning aims to recover a graph G that entails the conditional independencies as those in P.*

It is well-known that discovering the optimal structure from observational data is an NP-hard problem even for a small number of variables [Chickering, 1996]. Therefore, structure learning algorithms tend to offer approximate solutions, with the potential to offer exact solutions in problems of lower dimensionality, such as when the number of variables is low or the graph tree-width is low. In general, there are three types of structure learning algorithms; score-based, constraint-based, and hybrid algorithms. Score-based algorithms search and compare different candidate graphs with an objective function. These algorithms search through the space of possible graphical structures by performing different edge configurations such as adding, removing or reversing edges, and assign a model-selection score to each graph visited. The objective function determines the model-selection score, where the aim is to find a balance between model fitting and model dimensionality. Different search strategies and objective functions may lead to different learning effectiveness and efficiency, and some algorithms are sensitive to the order of the variables read from data. Some of the most important objective functions and score-based algorithms are covered in Subsection 2.2.2.

In contrast, constraint-based structure learning infers the structure of a graphical model based on a series of *conditional independence* (CI) tests. Specifically, unlike score-based algorithms that optimise an objective function, constraint-based algorithms recover structures that are consistent with the CI tests derived from the input data. Constraint-based learning usually involves starting from a fully connected graph and removing edges that contradict the results obtained from CI tests. Then, conditional dependency tests and orientation rules that rely on the Markov, faithfulness and causal sufficiency assumptions are used to orientate some of the remaining edges, leading to a CPDAG output. We introduce several widely-used functions for CI and constraint-based algorithms in Subsection 2.2.3.

Furthermore, hybrid algorithms combine the strengths, but also the limitations, of both score-based and constraint-based learning. Hybrid learning typically involves two phases, where the first phase uses CI tests to restrict the search space of graphs, and the second phase applies score-based solutions to the restricted search space. Subsection 2.2.4 covers five classic hybrid structure learning algorithms.

Given the Markov, faithfulness, and causal sufficiency assumptions, the set of DAGs that belong to the same MEC represent the same marginal probability distribution which makes them statistically indistinguishable given the observational data. This means that traditional structure learning algorithms can identify structures only up to a Markov equivalence class, or the CPDAG of the true DAG, from observational data. However, algorithms based on the *Functional Causal Models* (FCM) assume a functional form of dependencies from observational data to enable them to identify a unique DAG. The functional equations or expressions used in FCMs describe how the value of one variable depends on the values of its parents, plus a stochastic noise term. These additional assumptions about the class of the dependency function and the noise term guide the FCMs-based algorithms towards recovering a unique DAG structure from observational data, which is normally unachievable for score-based and constraint-based algorithms that employ score-equivalent objective functions or CI tests. Subsection 2.2.5 describes some of the commonly used structure learning algorithms that learn FCMs.

2.2.1 Evaluation metrics

Different methods exist to evaluate structure learning algorithms. The two main types of evaluation are graph-based metrics and inference-based metrics. The former approach is generally used when evaluating structure learning algorithms with synthetic data, where the purpose is to investigate the capability of a structure learning algorithm in terms of recovering the underlying ground truth graph that generated the data. On the other hand, the latter approach is more commonly used to evaluate the performance of these algorithms when applied to real data where there is no access to the ground truth graph.

Commonly used graph-based metrics include the *Structural Hamming Distance* (SHD) [Tsamardinos et al., 2006] and the F_1 score. The SHD metric reflects the number of edge additions, edge removals and arc reversals required to move from the learnt graph to true graph. This means that the SHD score grows with the number of the variables, which means that graphs learnt from larger networks produce higher SHD scores. This scaling issue can be partly addressed by dividing the SHD score by the number of the variables; often referred to as the re-scaled SHD score. On the other hand, the F_1 score ranges between 0 and 1 by combining *Precision* and *Recall* in the following form:

$$F_1 = 2\frac{Precision \cdot Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}\,, \tag{2.2}$$

where TP is the number of edges that exist in both the learnt graph and true graph, FP is the number of edges that exist in the learnt graph but not in true graph, and FN is the number of edges that exist in the true graph but not in the learnt graph.

In contrast to the graphical metrics, inference-based metrics rely on measures of statistical distance or model selection functions to evaluate learnt structures. An example of a measure of statistical distance is the *Kullback-Leibler* (KL) divergence (or relative entropy), where a KL divergence score of 0 indicates that the two distributions contain equal information [Kullback and Leibler, 1951]. Model-selection functions include the Bayesian Information Criterion (BIC) and the Bayesian Dirichlet equivalent uniform (BDeu), which are also used as objective functions in score-based learning, and are covered in detail in Subsection 2.2.2. A limitation of inference-based metrics is that they are influenced by both structure learning and parameter learning. Therefore, evaluators such as the KL divergence and the BIC and BDeu scores judge the underlying model as a whole rather than its structure alone. In this thesis, most of the results are evaluated using the F_1 and SHD scores since the performance of the proposed solutions is investigated primarily with synthetic experiments, some of which are based on real-world graphical models and others on randomly generated networks.

2.2.2 Score-based structure learning

Objective function

Objective functions are used for model-selection by score-based structure learning algorithms that search for the optimal graph in the search-space of graphs. If an objective score can be written as the sum of the local scores of every variable given its parents, it is called a *decomposable* score. Most objective functions are decomposable, and include the BIC, the *Bayesian Dirichlet equivalent* (BDe) [Heckerman et al., 1995], the BDeu, and the *quotient Normalised Maximum Likelihood* (qNML) [Silander et al., 2018]. The BIC score of a DAG $\mathcal{G}$ given data D is defined as:

$$\begin{aligned} S_{BIC}(\mathcal{G}, D) &= \sum_i S_{BIC}(V_i \mid \boldsymbol{Pa}_i) \\ &= \sum_i S_{LL}\left(V_i \mid \boldsymbol{Pa}_i; \hat{\Theta}_i\right) - \frac{\log(N)}{2} \cdot |\hat{\Theta}_i|, \end{aligned} \tag{2.3}$$

where $S_{LL}\left(V_i \mid \boldsymbol{Pa}_i; \hat{\Theta}_i\right)$ is the maximised value of the Log-Likelihood (LL) of V_i given $\boldsymbol{Pa}_i$, $\hat{\Theta}_i$ is the maximum likelihood estimates of the parameters of V_i in $\mathcal{G}$, N is the sample size of D, and $|\hat{\Theta}_i|$ is the number of free parameters in $\hat{\Theta}_i$. If $\mathcal{G}$ is defined over a set of discrete multinomial variables $\boldsymbol{V} = \{V_1, \dots V_n\}$, then the BIC score has the following form:

$$S_{BIC}(\mathcal{G}, D) = \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \cdot \log \frac{N_{ijk}}{N_{ij}} - \frac{\log(N)}{2} \cdot (r_i - 1) q_i, \tag{2.4}$$

where N_{ijk} is the number of cases in data set D in which the variable V_i takes its k^{th} value and the parents of V_i take the j^{th} configuration. Similarly, N_{ij} is the number of cases in data set D where the parents of V_i take their j^{th} configuration and, therefore, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Lastly, r_i represents the number of distinct values of V_i and q_i represents the number of configurations of the parents of V_i. Haughton [1988] has shown that BIC is a *consistent* objective score which means it selects the true model with probability one as the sample size approaches infinity.

Definition 2.2.2 (Consistent score criterion)**.** *If data D contain n iid observations from distribution p, a score S is consistent if the following two conditions hold as the sample size $n \to \infty$.*

1. *For two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$, if there exists a parameterised BN $(\mathcal{G}_1, \boldsymbol{\theta}_1)$ that exactly represents p but no such parameterised BN exists for $\mathcal{G}_2$ to represent p, then $S(\mathcal{G}_1, D) > S(\mathcal{G}_2, D)$.*

2. *For two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$, if there exists parameterised BNs $(\mathcal{G}_1, \boldsymbol{\theta}_1)$ and $(\mathcal{G}_2, \boldsymbol{\theta}_2)$ that both exactly represent p, and if $\mathcal{G}_1$ contains fewer parameters than $\mathcal{G}_2$, then $S(\mathcal{G}_1, D) > S(\mathcal{G}_2, D)$.*

Unlike the BIC score, the BDe score is specifically designed for discrete BNs. If data set D contains only discrete values, we denote the ith column of D by D_i, and the columns of D that correspond to variable set $\boldsymbol{U}$ by $D_{\boldsymbol{U}}$. We also denote the entries, or rows, of column i by $D_{i,\boldsymbol{U}=j}$ when variable set $\boldsymbol{U}$ has configuration j. Assume the local distribution $P(V_i \mid \boldsymbol{Pa}_i, \Theta_i)$ is multinomial, and that each parameter Θ_i has a Dirichlet prior, the BDe score is:

$$\begin{aligned} S_{BDe}(\mathcal{G}, D) &= \log P(\mathcal{G}) + \log P(D \mid \mathcal{G}) \\ &= \log P(\mathcal{G}) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \log \int P\left(D_{i,\boldsymbol{Pa}_i=j} \mid \Theta_i\right) P(\Theta_i)\, d\Theta_i \\ &= \log P(\mathcal{G}) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left(\log \frac{\Gamma\left(N'_{ij}\right)}{\Gamma\left(N_{ij} + N'_{ij}\right)} + \sum_{k=1}^{r_i} \log \frac{\Gamma\left(N_{ijk} + N'_{ijk}\right)}{\Gamma\left(N'_{ijk}\right)} \right), \end{aligned} \tag{2.5}$$

where $P(\mathcal{G})$ is the prior probability of DAG $\mathcal{G}$ which is normally assumed to be uniform, $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$, $N'_{ijk} = N' \cdot P(V_i = k, \boldsymbol{Pa}_i = j \mid \mathcal{G})$ and N' is a hyperparameter called *equivalent sample size* which controls the strength of the belief on the prior parameters. If we assume $P(V_i, \boldsymbol{Pa}_i \mid \mathcal{G})$ is uniformly distributed, which means $P(V_i = k, \boldsymbol{Pa}_i = j \mid \mathcal{G}) = \frac{1}{r_i q_i}$, then we have the BDeu score as follow:

$$S_{BDeu}(\mathcal{G}, D) = \log P(\mathcal{G}) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left(\log \frac{\Gamma\left(\frac{N'}{q_i}\right)}{\Gamma\left(N_{ij} + \frac{N'}{q_i}\right)} + \sum_{k=1}^{r_i} \log \frac{\Gamma\left(N_{ijk} + \frac{N'}{r_i q_i}\right)}{\Gamma\left(\frac{N'}{r_i q_i}\right)} \right) \tag{2.6}$$

However, it has been shown that the BDe and BDeu scores are sensitive to the selection of the hyperparameter N' [Steck and Jaakkola, 2002, Silander et al., 2007]. Therefore, Silander et al. [2018] proposed to use the *Normalised Maximum Likelihood* (NML) criterion to address this problem, which is defined as:

$$P_{NML}\left(D_{i,\boldsymbol{Pa}_i=j}; \mathcal{G}\right) = \frac{P\left(D_{i,\boldsymbol{Pa}_i=j} \mid \hat{\Theta}_i\right)}{\sum_{D'} P\left(D'_{i,\boldsymbol{Pa}_i=j} \mid \hat{\Theta}_i(D')\right)}, \tag{2.7}$$

where D' is a data set with the same sample size and variables as in D, and $\hat{\Theta}_i(D')$ is the maximum likelihood estimates of the parameters obtained from D'. Then, the hyperparameter-free qNML score function is formally defined as:

$$S_{qNML}(\mathcal{G}, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \log \frac{P_{NML}\left(D_{i,\boldsymbol{Pa}_i=j}; \mathcal{G}\right)}{P_{NML}\left(D_{\boldsymbol{Pa}_i=j}; \mathcal{G}\right)} \tag{2.8}$$

Score-based algorithms

One of the earliest and simplest score-based algorithm is the *Hill-Climbing* (HC) algorithm [Heckerman et al., 1995], which performs greedy search over the DAG space. HC is a heuristic iterative algorithm which starts from a default graph (normally an empty graph). At each iteration, HC explores all the neighbouring DAGs of the given DAG by inserting, deleting or reversing one directed edge at a time. The pseudocode of the HC algorithm is provided in Algorithm 1. HC is a highly efficient algorithm, especially when paired with a decomposable objective function that would result in only one or two local scores re-computed at each iteration. However, HC is known to terminate at a local optimal solution, and its solution may be inadequate when the search space is large. Several techniques are proposed in the literature to improve its performance. For example, Heckerman et al. [1995] proposed *local restarts* by adding random perturbations on the returned

DAG and restarting HC from the perturbed DAG. Bouckaert [1994] proposed *tabu* that permits the algorithm to continue to explore graphical regions that minimally decrease the objective score, and keeping track of a *tabu list* to prevent the algorithm from returning to a recently visited DAG.

Algorithm 1 The Hill-Climbing structure learning algorithm

```
1: procedure HILL CLIMBING
       Input: data set $D$
       Output: learnt DAG $\mathcal{G}$
2:     $\mathcal{G} \leftarrow$ empty graph
3:     repeat
4:         $\delta \leftarrow 0$
5:         repeat
6:             construct a neighbouring DAG $\mathcal{G}_{nei}$ by adding, reversing or deleting an edge from $\mathcal{G}$
7:             if $S(\mathcal{G}_{nei} \mid D) - S(\mathcal{G} \mid D) > \delta$ then
8:                 $\delta \leftarrow S(\mathcal{G}_{nei} \mid D) - S(\mathcal{G} \mid D)$
9:                 $\mathcal{G}_{update} \leftarrow \mathcal{G}_{nei}$
10:            end if
11:        until all possible edge operations have been attempted
12:        if $\delta > 0$ then
13:            $\mathcal{G} \leftarrow \mathcal{G}_{update}$
14:        end if
15:    until $\delta = 0$
16:    return $\mathcal{G}$
17: end procedure
```

The *Greedy Equivalence Search* (GES) algorithm [Chickering, 2002] is a powerful greedy search algorithm that guarantees to return the CPDAG of the ground truth DAG given the causal sufficiency assumption, the causal faithfulness assumption, a consistent objective score, and when sample size goes to infinity. By searching over the CPDAG space rather than DAG space, GES significantly reduces the size of the search space and therefore prevents itself from becoming trapped in local optima. In order to explore all possible neighbouring CPDAGs of the current CPDAG, it implements the traditional insert, delete and reverse edge operations, but also some additional operators such as inserting and deleting undirected edge and making v-structure. The author derived an efficient method to compute the score change between the current best CPDAG and the CPDAG explored by each edge operator, such that GES recomputes a maximum of two local scores - for up to two nodes - at each iteration; similar to how other score-based algorithms compute the DAG score when paired with a score-equivalent function. Later, Ramsey et al. [2017] proposed the *fast GES* (fGES) algorithm that is a more efficient version of GES, aiming to expedite the structure learning process on large data sets with millions of variables. This acceleration is accomplished by leveraging parallel computing and reducing the search space. fGES achieves this improvements in efficiency by also assuming that no edge exists between two uncorrelated variables, and so it traverses a reduced search space of graphs. Unlike GES, when this assumption is violated, fGES would fail to recover the true CPDAG even when given infinite sample size.

The *Ordering Based Search* (OBS) algorithm [Teyssier and Koller, 2005] is another greedy search heuristic algorithm that searches over the node ordering space which has a size of $2^{O(n\log n)}$ and is significantly smaller than the size of DAG space $2^{\Omega(n^2)}$, where n is the number of variables. OBS starts from a random node ordering and attempts to swap every pair of adjacent nodes in the ordering at each iteration, and moves to the ordering with the highest score. The score of an ordering is defined by the highest score of the graph that is consistent with the ordering. By caching the visited ordering scores and pruning the parents set that are impossible to be in the optimal graph, OBS tends to converge faster than HC.

Scanagatta et al. [2015] developed a variant of OBS named *Acyclic Selection OBS* (ASOBS) which guarantees to return a graph that has a higher or equal score than the graph discovered by OBS. ASOBS uses a more time-efficient pruning strategy that prunes off the parent sets that are impossible to be optimal, without ever computing their score. Additionally, ASOBS allows the presence of back-arcs, which are directed edges from a lower ordering node to a higher ordering node, as long as it does not form cycles. Later, Scanagatta et al. [2017] developed another variant of ASOBS called *Window Acyclic Selection OBS* (WINASOBS) which employs a more powerful window operator that changes the position of a group of nodes in the node ordering, and uses the same relaxation as ASOBS when generating a DAG given the ordering.

Next, we describe several *exact* algorithms which guarantee to return the highest scoring graph from a given search space of graphs. One of the earliest learning strategies in this category is *dynamic programming* [Ott et al., 2003, Koivisto and Sood, 2004] which recursively divides structure learning into several simpler sub-problems. Singh and Moore [2005] observed that every DAG must have at least one *sink node*, i.e., node without children, thus the graph score $S(\mathcal{G})$ can be expressed as a recursive function:

$$S(\mathcal{G}, D) = S(V_s \mid \boldsymbol{Pa}_s) + S\left(\mathcal{G}_{\setminus s}, D\right) , \tag{2.9}$$

where V_s is a sink node in the graph $\mathcal{G}$, $\boldsymbol{Pa}_s$ is the parent set of V_s in $\mathcal{G}$ and $\mathcal{G}_{\setminus s}$ is a sub-graph of $\mathcal{G}$ that excludes V_s and all its connecting edges. They then developed a depth-first search algorithm to find the optimal DAG over the node ordering space. Figure 2.3 illustrates the lattice searched by dynamic programming, where each path from top to bottom represents a node ordering and each edge explores a sink node and its optimal parent set. For example, the red path represents the ordering $\{1, 2, 3, 4\}$ and the blue path represents the ordering $\{4, 1, 3, 2\}$. The optimal DAG can be constructed by identifying the path that maximises Equation 2.9.

However, dynamic programming is inefficient since it needs to explore all the possible parent sets for every variable and has complexity of $O(n2^n)$, where n is the number of variables. Yuan et al. [2011] formulate learning optimal DAG as a shortest path finding problem. They define the cost of an edge from $\boldsymbol{S}_1$ to $\boldsymbol{S}_2$ in Figure 2.3 by $BestMDL(X, \boldsymbol{S}_2)$, where X is the only node in S_1 but not in S_2, and $BestMDL(X, \boldsymbol{S})$ is defined as follows:

$$BestMDL(X, \boldsymbol{S}) = \max_{\boldsymbol{Pa}_X \subseteq \boldsymbol{S}} S_{BIC}(X \mid \boldsymbol{Pa}_X) \tag{2.10}$$

Then, they employ A* search to find the shortest path from the top layer to the bottom layer with the lowest cost. Moreover, they also propose several pruning rules to exclude those parent sets that are impossible to be optimal, so that A* would not explore them.

Another exact algorithm is the *Globally Optimal Bayesian Network learning using Integer Linear Programming* (GOBNILP) [Cussens, 2011] which encodes structure learning as the following

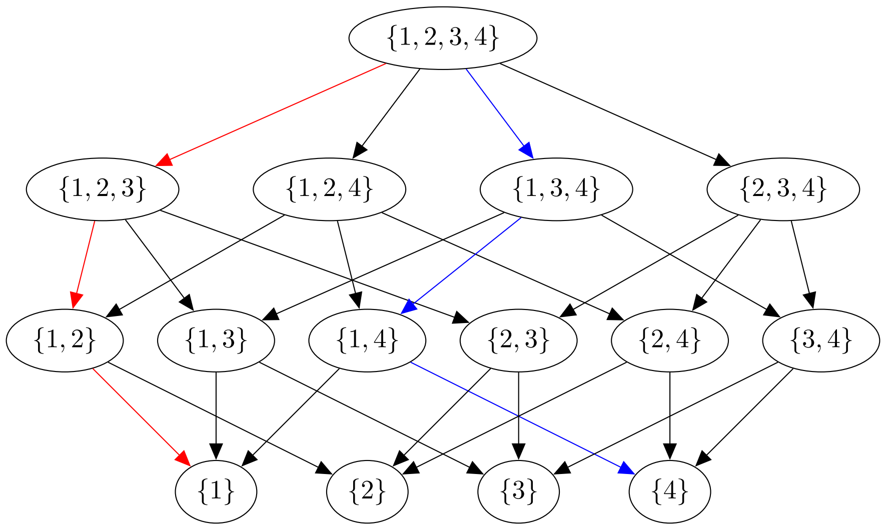


Figure 2.3: Lattice explored by dynamic programming. Each directed edge explores a sink node and searches its optimal parent set. Each path from top to bottom represents a node ordering.

constrained *integer programming* (IP) problem:

$$\text{Instantiate the } I\left(\boldsymbol{W} \to V_i\right)\text{s to maximise} \sum_{V_i \in \boldsymbol{V}} \sum_{\boldsymbol{W} \subseteq \boldsymbol{V} \setminus \{V_i\}} S\left(V_i \mid W\right) I\left(\boldsymbol{W} \to V_i\right),$$

$$\text{subject to } I\left(\boldsymbol{W} \to V_i\right)\text{s represent a DAG,}$$

where $I\left(\boldsymbol{W} \to V_i\right)$ is a binary variable with value 1 if and only if $\boldsymbol{W}$ is the parent of V_i in the optimal DAG. The constraint can be formulated into two linear constraints called the *convexity constraint* and the *cluster-based constraint*. The convexity constraint ensures that each variable has exactly one parent set, whereas the cluster-based constraint rules out graphs with cycles.

Definition 2.2.3 (Convexity constraint)**.**

$$\forall V_i \in \boldsymbol{V}: \sum_{\boldsymbol{W} \subseteq \boldsymbol{V} \setminus \{V_i\}} I\left(\boldsymbol{W} \to V_i\right) = 1$$

Definition 2.2.4 (Cluster-based constraint)**.**

$$\forall \boldsymbol{C} \subseteq \boldsymbol{V}, \forall k, 1 \leq k \leq |\boldsymbol{C}|: \sum_{V_i \in \boldsymbol{C}} \sum_{\boldsymbol{W}: \boldsymbol{W} \cap \boldsymbol{C} = \varnothing} I\left(\boldsymbol{W} \to V_i\right) \geq 1$$

Since the number of cluster-based constraints is exponential to the number of the variables, it is impossible to impose all constraints into the IP solver. GOBNILP constructs a sub-IP solver to choose the constraint with the highest efficacy in the corresponding iteration. Even so, GOBNILP still needs to place a limit on the number of parents for each node to be applicable in practice.

The aforementioned score-based algorithms are all based on combinatorial optimisation, i.e., finding a graph which optimises the objective function and satisfies the acyclic constraint. The NOTEARS Zheng et al. [2018] algorithm is the first to convert structure learning into a continuous optimisation problem. Assume the observed data is sampled from a linear model such that the value of V_i is a linear combination of the values of its parents $\boldsymbol{Pa}_i$ and its own disturbance ϵ_i:

$$V_i = \sum_{V_j \in \boldsymbol{Pa}_i} w_{ij} V_j + \epsilon_i. \tag{2.11}$$

Let W be the matrix composed by w_{ij}, the graph represented by W is acyclic if and only if $h(W) = Tr\left(e^{W \circ W}\right) - n = 0$, where $\circ$ is Hadamard (element-wise) product, Tr is the trace of matrix and n is the number of variables. This converts structure learning into the following continuous optimisation problem:

$$\min_{W \in \mathbb{R}^{d \times d}} \frac{1}{2N} \|\boldsymbol{V} - \boldsymbol{V}W\| + \lambda \|W\|_1, \text{ subject to } h(W) = 0, \tag{2.12}$$

where N is the number of samples. Equation 2.12 can then be executed using a standard augmented Lagrangian method [Fortin and Glowinski, 2000].

2.2.3 Constraint-based structure learning

Conditional independence tests

CI tests determine whether two variables are independent conditional on a variable set, including an empty set. The most commonly used CI test for discrete data is the G test which is statistically asymptotic to the χ^2 test. Like most of the CI tests, the G test assumes a null hypothesis where V_i and V_j are independent given $\boldsymbol{S}$. The G test calculates the *statistic* G and the *degree of freedom* df from data, and uses them to calculate the *p-value*, which is also known as *significance value*, based on the χ^2 distribution. If the p-value is below a predefined threshold, typically set to 0.01, 0.05 or 0.1, the null hypothesis is rejected and V_i and V_j are assumed to be dependent given $\boldsymbol{S}$. These are calculated as follows:

$$G = 2 \sum_{a \in V_i} \sum_{b \in V_j} \sum_{\boldsymbol{c} \in \boldsymbol{S}} N_{ab\boldsymbol{c}} \log \frac{N_{ab\boldsymbol{c}} N_{\boldsymbol{c}}}{N_{a\boldsymbol{c}} N_{b\boldsymbol{c}}}, \tag{2.13}$$

$$df = (|V_i| - 1)(|V_j| - 1) \prod_{S_k \in \boldsymbol{S}} |S_k| \tag{2.14}$$

where $N_{ab\boldsymbol{c}}$ is the number of cases with specific values $V_i = a, V_j = b$ and $\boldsymbol{S} = \boldsymbol{c}$. $N_{a\boldsymbol{c}}, N_{b\boldsymbol{c}}$ and $N_{\boldsymbol{c}}$ are defined analogously.

In the case of linear Gaussian data, Fisher's Z-test is commonly used which assumes the partial correlation coefficient is zero. The statistic of Fisher's Z-test is based on Fisher's Z-transformation which is defined as:

$$\hat{Z} = \frac{1}{2} \log \frac{1 + \hat{\rho}_{V_i V_j | \boldsymbol{S}}}{1 - \hat{\rho}_{V_i V_j | \boldsymbol{S}}}, \tag{2.15}$$

where $\hat{\rho}_{V_i V_j | \boldsymbol{S}}$ is the partial correlation coefficient between V_i and V_j given $\boldsymbol{S}$. The transformed partial correlation $\hat{Z}$ follows a normal distribution with mean $\mu = \frac{1}{2} \log \frac{1 + \hat{\rho}_{V_i V_j | \boldsymbol{S}}}{1 - \hat{\rho}_{V_i V_j | \boldsymbol{S}}}$ and standard deviation $\sigma = \frac{1}{\sqrt{N - q - 3}}$, where q is the number of variables in $\boldsymbol{S}$. Therefore, we can use the normal distribution Z, as described below, to compute the p-value given the null hypothesis of zero partial correlation ($Z_0 = 0$):

$$Z = \frac{\hat{Z} - Z_0}{\sigma} = \frac{1}{2} \sqrt{N - q - 3} \log \frac{1 + \hat{\rho}_{V_i V_j | \boldsymbol{S}}}{1 - \hat{\rho}_{V_i V_j | \boldsymbol{S}}} \tag{2.16}$$

If the functional form between variables is non-linear, the CI test based on zero partial correlation might lead to incorrect conclusion. Zhang et al. [2011] proposed the *Kernel-based Conditional Independence test* (KCI-test) which tests CI for continuous variables without assuming the functional form between variables. Let $(\mathcal{X}, \mathcal{B})$ be a measurable space, X a variable on $\mathcal{X}$, and $k_{\mathcal{X}}$ a positive definite kernel defined on $\mathcal{X}$. KCI-test constructs its test statistics using *characteristic* kernel. A kernel $k_{\mathcal{X}}$ is said to be characteristic if $E_{X\sim P}\left[f\left(X\right)\right] = E_{X\sim Q}\left[f\left(X\right)\right], \forall f \in \mathcal{H}_{\mathcal{X}}$[1] implies $P = Q$, where P and Q are two probability distributions of X [Fukumizu et al., 2007].

We denote $\ddot{X} = \left(X, Z\right), \boldsymbol{x} = \left\{x_1, \ldots, x_n\right\}, \boldsymbol{y} = \left\{y_1, \ldots, y_n\right\}, \boldsymbol{z} = \left\{z_1, \ldots, z_n\right\}$ the i.i.d. sample of X, Y and Z respectively. $\boldsymbol{K}_{\ddot{X}}$ is the kernel matrix of sample $\ddot{\boldsymbol{x}}$, and the corresponding centralised kernel matrix is $\widetilde{\boldsymbol{K}}_{\ddot{X}} = \boldsymbol{H}\boldsymbol{K}_{\ddot{X}}\boldsymbol{H}$, where $\boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$, $\boldsymbol{I}$ is the identity matrix and $\mathbf{1}$ is the vector of 1's. $\boldsymbol{K}_Y, \widetilde{\boldsymbol{K}}_Y, \boldsymbol{K}_Z$ and $\widetilde{\boldsymbol{K}}_Z$ are defined analogously. We define $\widetilde{\boldsymbol{K}}_{\ddot{X}|Z} = \boldsymbol{R}_Z\widetilde{\boldsymbol{K}}_{\ddot{X}}\boldsymbol{R}_Z$, where $\boldsymbol{R}_Z = \epsilon\left(\widetilde{\boldsymbol{K}}_Z + \epsilon\boldsymbol{I}\right)^{-1}$, ϵ is a small positive regularisation parameter [Schölkopf et al., 2002]. Furthermore, suppose we have the eigenvalue decomposition of $\widetilde{\boldsymbol{K}}_{\ddot{X}|Z}$ and $\widetilde{\boldsymbol{K}}_{Y|Z}$, i.e., $\widetilde{\boldsymbol{K}}_{\ddot{X}|Z} = \boldsymbol{V}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}}\boldsymbol{\Lambda}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}}\boldsymbol{V}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}}$ and $\widetilde{\boldsymbol{K}}_{Y|Z} = \boldsymbol{V}_{\boldsymbol{y}|\boldsymbol{z}}\boldsymbol{\Lambda}_{\boldsymbol{y}|\boldsymbol{z}}\boldsymbol{V}_{\boldsymbol{y}|\boldsymbol{z}}$, let $\boldsymbol{\psi}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}} = \left[\psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},1}\left(\ddot{\boldsymbol{x}}\right), \ldots, \psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},n}\left(\ddot{\boldsymbol{x}}\right)\right] = \boldsymbol{V}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}}\boldsymbol{\Lambda}^{1/2}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}}$ and $\boldsymbol{\phi}_{\boldsymbol{y}|\boldsymbol{z}} = \left[\phi_{\boldsymbol{y}|\boldsymbol{z},1}\left(\boldsymbol{y}\right), \ldots, \phi_{\boldsymbol{y}|\boldsymbol{z},n}\left(\boldsymbol{y}\right)\right] = \boldsymbol{V}_{\boldsymbol{y}|\boldsymbol{z}}\boldsymbol{\Lambda}^{1/2}_{\boldsymbol{y}|\boldsymbol{z}}$. Then, under the null hypothesis $(X \perp\!\!\!\perp Y \mid Z)$, the statistic of KCI-test between X and Y given Z is defined as

$$T_{CI} = \frac{1}{n}Tr\left(\widetilde{\boldsymbol{K}}_{\ddot{X}|Z}\widetilde{\boldsymbol{K}}_{Y|Z}\right) \tag{2.17}$$

which has the same asymptotic distribution as

$$\check{T}_{CI} = \frac{1}{n}\sum_{k=1}^{n^2}\lambda_k \cdot z_k^2\,, \tag{2.18}$$

where λ_k are the eigenvalues of $\check{\boldsymbol{w}}\check{\boldsymbol{w}}^T$, $\check{\boldsymbol{w}} = \left[\check{\boldsymbol{w}}_1, \ldots, \check{\boldsymbol{w}}_n\right]$ with the vector $\check{\boldsymbol{w}}_t$ obtained by stacking $\check{\boldsymbol{M}}_t = \left[\psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},1}\left(\ddot{x}_t\right), \ldots, \psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},1}\left(\ddot{x}_t\right)\right]^T \cdot \left[\phi_{\boldsymbol{y}|\boldsymbol{z},1}\left(y_t\right), \ldots, \psi_{\boldsymbol{y}|\boldsymbol{z},1}\left(y_t\right)\right]$, and z_k are i.i.d. χ^2_1-distributed variables. Therefore, we can obtain the Monte Carlo approximation of the null distribution $\check{T}_{CI}$ by generating n^2 samples from χ^2_1 distribution and summing them up with weights λ_k/n. Finally, the p-value of KCI-test can be estimated by the probability of samples from $\check{T}_{CI}$ exceeding T_{CI}.

Constraint-based algorithms

One of the earliest constraint-based algorithms is the SGS algorithm [Spirtes et al., 1990]. SGS is *sound* in that it guarantees to find the true CPDAG if the faithfulness and causal sufficiency assumptions hold, and while sample size goes to infinite. SGS relies on the two following rules about graph $\mathcal{G}$, which is defined over variables $\boldsymbol{V}$ given the faithfulness and causal sufficiency assumptions [Verma and Pearl, 1990]:

1. Two variables V_i and V_j are neighbours in graph $\mathcal{G}$ if and only if V_i and V_j are conditionally dependent given any subset $\boldsymbol{S} \subseteq \boldsymbol{V} \setminus \left\{V_i, V_j\right\}$.

2. In graph $\mathcal{G}$, if $\langle V_i, V_j, V_k\rangle$ is an unshielded triple, then V_i, V_j and V_k form a v-structure in $\mathcal{G}$, i.e., $V_i \rightarrow V_j \leftarrow V_k$, if and only if V_i and V_k are dependent given any subset $\boldsymbol{S} \subseteq \boldsymbol{V} \setminus \left\{V_i, V_k\right\}$ such that $V_j \in \boldsymbol{S}$.

[1] $\mathcal{H}_{\mathcal{X}}$ is the corresponding *Reproducing Kernel Hilbert Space* (RKHS) for $k_{\mathcal{X}}$

SGS initialises $\mathcal{G}$ as a fully connected undirected graph, and executes the following three learning phases:

1. *Adjacency phase*: Based on rule 1, for every pair of variables V_i and V_j, SGS performs CI tests between V_i and V_j conditional on every subset $\boldsymbol{S} \subseteq \boldsymbol{V} \setminus \{V_i, V_j\}$. If V_i and V_j are found to be independent given at least one conditioning set, SGS removes the undirected edge between V_i and V_j in graph $\mathcal{G}$. The output of the adjacency phase is referred as skeleton $\mathcal{C}$.

2. *V-structure phase*: Based on rule 2, for every unshielded triple $\langle V_i, V_j, V_k \rangle$ in skeleton $\mathcal{C}$, SGS performs CI tests between V_i and V_k conditional on every subset $\boldsymbol{S} \subseteq \boldsymbol{V} \setminus \{V_i, V_k\}$ that contains V_j. If V_i and V_k are dependent given any conditional set, then SGS orientates $V_i \to V_j$ and $V_j \leftarrow V_k$ in $\mathcal{G}$.

3. *Orientation propagation phase*: For every remaining undirected edge, if one of the orientations introduces additional v-structures or cycles, SGS orientates that undirected edge in the opposite direction in $\mathcal{G}$.

Algorithm 2 The SGS algorithm

```
1: procedure SGS($D$)
       Input: data $D$
       Output: CPDAG $\mathcal{G}$
       ▷ Adjacency phase
2:     $\mathcal{G} \leftarrow$ fully connected undirected graph
3:     for each variable pair $\langle V_i, V_j \rangle$ do
4:         if $\exists \boldsymbol{S} \subseteq \boldsymbol{V} \setminus \{V_i, V_j\}$, s.t. $V_i \perp\!\!\!\perp V_j \mid \boldsymbol{S}$ then
5:             remove $V_i - V_j$ from $\mathcal{G}$
6:         end if
7:     end for
       ▷ V-structure phase
8:     for each unshielded triple $\langle V_i, V_j, V_k \rangle \in \mathcal{G}$ do
9:         if $V_i \not\perp\!\!\!\perp V_k \mid \boldsymbol{S} \cup \{V_j\}, \forall \boldsymbol{S} \subseteq \boldsymbol{V} \setminus \{V_i, V_j, V_k\}$ then
10:            orientate $V_i \to V_j \leftarrow V_k$ in $\mathcal{G}$
11:        end if
12:    end for
       ▷ Orientation phase
13:    repeat
14:        for each undirected edge $V_i - V_j \in \mathcal{G}$ do
15:            if orientating $V_i \to V_j$ causes cycles or v-structures in $\mathcal{G}$ then
16:                orientate $V_i \leftarrow V_j$ in $\mathcal{G}$
17:            end if
18:        end for
19:    until $\mathcal{G}$ is unchanged
20:    return $\mathcal{G}$
21: end procedure
```

The pseudocode for SGS is shown in Algorithm 2. While SGS is sound in theory, it is time consuming due to the high number of conditional independence tests it performs and, therefore, inefficient. In the worst case, SGS requires $n(n-1)2^{n-3}$ CI tests in the adjacency phase, where n is the number of variables in $\boldsymbol{V}$. To improve learning efficiency, Spirtes and Glymour [1991] proposed the PC algorithm which requires fewer CI tests than SGS. PC also starts from a fully connected undirected graph, and adopts the same three phases as SGS but modifies the adjacency phase and v-structure phase as follows:

1. *Adjacency phase in PC*: For every ordered variable pair $\langle V_i, V_j \rangle$ adjacent in $\mathcal{G}$, PC performs CI tests conditional on the subsets $\boldsymbol{S} \subseteq Adj\,(\mathcal{G}, V_i) \setminus \{V_j\}$ with increasing set size, where $Adj\,(\mathcal{G}, V_i)$ represents the set of variables that are adjacent to V_i in the current graph $\mathcal{G}$. If V_i and V_j are found to be independent given such a conditioning set, PC removes the undirected edge between V_i and V_j from $\mathcal{G}$.

2. *V-structure phase in PC*: For every unshielded triple $\langle V_i, V_j, V_k \rangle$ in skeleton $\mathcal{C}$, if the variable set that makes V_i and V_k independent in the adjacency phase does not contain the variable V_j, then PC orientates $V_i \rightarrow V_j$ and $V_j \leftarrow V_k$ in $\mathcal{G}$.

PC modifies the adjacency phase based on the Markov assumption which hypothesises that a variable is independent of its non-descendants given its parents. Therefore, the minimal separating set of V_i and V_j must be a subset of either $Adj\,(\mathcal{G}, V_i)$ or $Adj\,(\mathcal{G}, V_j)$. The maximal number of CI tests required by PC during the adjacency phase is $\frac{n^2(n-1)^{k-1}}{(k-1)!}$, where k is the maximal degree of any variable. Although PC suffers the same complexity bound as SGS in the worst case, it is more efficient than SGS when the underlying true DAG is sparse. The pseudocode for PC is shown in Algorithm 3.

Algorithm 3 The PC algorithm

```
 1: procedure PC($D$)
        Input: data $D$
        Output: CPDAG $\mathcal{G}$
        ▷ Adjacency phase
 2:     $\mathcal{G} \leftarrow$ fully connected undirected graph
 3:     Sepset $\leftarrow \emptyset$
 4:     $l \leftarrow -1$
 5:     repeat
 6:         $l \leftarrow l + 1$
 7:         for each ordered variable pair $\langle V_i, V_j \rangle$ adjacent in $\mathcal{G}$ do
 8:             if $\exists \boldsymbol{S} \subseteq Adj\,(\mathcal{G}, V_i) \setminus \{V_j\}$, s.t. $V_i \perp\!\!\!\perp V_j \mid \boldsymbol{S}$ and $|\boldsymbol{S}| = l$ then
 9:                 remove $V_i - V_j$ from $\mathcal{G}$
10:                 Sepset$\,(V_i, V_j) =$ Sepset$\,(V_j, V_i) = \boldsymbol{S}$
11:             end if
12:         end for
13:     until Every variable $V_i$ satisfies $|Adj\,(\mathcal{G}, V_i)| \leq l$
        ▷ V-structure phase
14:     for each unshielded triple $\langle V_i, V_j, V_k \rangle \in \mathcal{G}$ do
15:         if $V_j \notin$ Sepset$\,(V_i, V_k)$ then
16:             orientate $V_i \rightarrow V_j \leftarrow V_k$ in $\mathcal{G}$
17:         end if
18:     end for
        ▷ Orientation phase
19:     repeat
20:         for each undirected edge $V_i - V_j \in \mathcal{G}$ do
21:             if orientating $V_i \rightarrow V_j$ causes cycles or v-structures in $\mathcal{G}$ then
22:                 orientate $V_i \leftarrow V_j$ in $\mathcal{G}$
23:             end if
24:         end for
25:     until $\mathcal{G}$ is unchanged
26:     return $\mathcal{G}$
27: end procedure
```

While the PC algorithm is theoretically sound and efficient, this theoretical strength does not always translate into practical effectiveness, and this applies to most algorithms. For example, PC is sensitive to the order of the variables as read from data and may return different results for different orderings. Moreover, as we will later show, PC is also sensitive to data noise, and any early mistakes PC might do will propagate to future independence or directionality decisions the algorithm might make. Next, we will present several modifications of the PC algorithm that are designed to enhance its performance in practical scenarios. Spirtes et al. [2000] proposed a variant of PC called the *Fast Causal Inference* (FCI) algorithm which does not assume causal sufficiency. FCI outputs a *Partially Orientated Inducing Path Graph* (POIPG) which is analogous to a PAG but contains slightly less information. FCI adopts the same adjacency phase and v-structure phase as PC but has another adjacency phase to discover the separating sets caused by unobserved confounders. This is because when there are unobserved confounders, the separating set for two observed variables may contain variables that are not neighbours of these two variables. Moreover, FCI uses a more complicated orientation phase to include the possible unobserved confounder in POIPG. Despite the presence of "Fast" in FCI's name, its adjacency phase is typically far more resource intensive than in the PC algorithm. *Really Fast Causal Inference* (RFCI) [Colombo et al., 2012] seeks to address this by reverting back to having just one adjacency phase instead of two as in FCI. The v-structure phase and one of the orientation rules are also modified to avoid orientation errors that might occur due to the fact that the PC adjacency phase is used rather than the more accurate FCI one.

Although PC is more efficient than SGS, its outcome is sensitive to the variable ordering read from data. Specifically, in the adjacency phase, when searching for the possible separating set for ordered variable pair $\langle V_i, V_j \rangle$, PC only explores the subsets of $Adj\left(\mathcal{G}, V_i\right)$. Therefore, an incorrect edge removal might lead to further errors in subsequent steps. Colombo and Maathuis [2014] found that the outcomes of all three phases in PC depend on the variable ordering, and proposed a variant of PC called PC-stable to resolve this issue. Similar to SGS and PC, PC-stable starts its learning phase from a fully connected graph $\mathcal{G}$ and uses the following three phases:

1. *Adjacency phase in PC-stable*: At each conditional set size l ranging from 0 to $n-1$, for every ordered variable pair $\langle V_i, V_j \rangle$, PC performs CI tests conditional on the subsets $\boldsymbol{S} \subseteq a\left(V_i\right) \setminus \{V_j\}$ with size l, where $a\left(V_i\right)$ is the set of adjacent variables of V_i in $\mathcal{G}$ which is only updated before processing all CI tests at conditional set size level l. If V_i and V_j are found to be independent given such a conditioning set, PC-stable removes the undirected edge between V_i and V_j in $\mathcal{G}$. This phase terminates if all $a\left(V_i\right)$ have size smaller than l.

2. *V-structure phase in PC-stable*: For every unshielded triple $\langle V_i, V_j, V_k \rangle$ in skeleton $\mathcal{C}$, PC-stable determines all subsets of $Adj\left(\mathcal{G}, V_i\right)$ or $Adj\left(\mathcal{G}, V_k\right)$ that d-separate V_i and V_k. If at least one such separating set is found and V_j is in none of those sets, PC-stable orientates $V_i \rightarrow V_j$ and $V_j \leftarrow V_k$ in $\mathcal{G}$. If $V_i \leftarrow V_j$ or $V_j \rightarrow V_k$ is orientated in a previous iteration, then PC-stable creates bi-directed edge $V_i \leftrightarrow V_j$ or $V_j \leftrightarrow V_k$ to ease conflicts.

3. *Orientation propagation phase in PC-stable*: For every remaining undirected edge, if one of the orientations introduces additional v-structures or cycles, PC-stable orientates that undirected edge in the opposite direction in $\mathcal{G}$. If both orientations cause either additional v-structures or cycles, PC-stable makes this edge bi-directional.

The pseudocode of PC-stable is shown in Algorithm 4. Ramsey [2016] proposed another variant of PC called PC-MAX which interprets the p-value as the reliability of CI tests, and uses the

Algorithm 4 The PC-stable algorithm

```
1: procedure PC-STABLE($D$)
       Input: data $D$
       Output: CPDAG $\mathcal{G}$
     ▷ Adjacency phase
2:   $\mathcal{G} \leftarrow$ fully connected undirected graph
3:   Sepset $\leftarrow \emptyset$
4:   $l \leftarrow -1$
5:   repeat
6:       $l \leftarrow l + 1$
7:       for each $V_i \in \mathcal{G}$ do
8:           $a(V_i) \leftarrow Adj(\mathcal{G}, V_i)$
9:       end for
10:      for each ordered variable pair $\langle V_i, V_j \rangle$ adjacent in $\mathcal{G}$ do
11:          if $\exists \boldsymbol{S} \subseteq a(V_i) \setminus \{V_j\}$, s.t. $V_i \perp\!\!\!\perp V_j \mid \boldsymbol{S}$ and $|\boldsymbol{S}| = l$ then
12:              remove $V_i - V_j$ from $\mathcal{G}$
13:              Sepset$(V_i, V_j)$ = Sepset$(V_j, V_i) = \boldsymbol{S}$
14:          end if
15:      end for
16:  until Every variable $V_i$ satisfies $|a(V_i) \setminus \{V_j\}| \leq l$
     ▷ V-structure phase
17:  for each unshielded triple $\langle V_i, V_j, V_k \rangle \in \mathcal{G}$ do
18:      if $\exists \boldsymbol{S} \subseteq Adj(\mathcal{G}, V_i)$ or $\boldsymbol{S} \subseteq Adj(\mathcal{G}, V_k)$, s.t. $V_i \perp\!\!\!\perp V_k \mid \boldsymbol{S}$ then
19:          if All such $\boldsymbol{S}$ do not contain $V_j$ then
20:              orientate $V_i * \rightarrow V_j \leftarrow * V_k$ in $\mathcal{G}$
21:          end if
22:      end if
23:  end for
     ▷ Orientation phase
24:  repeat
25:      for each undirected edge $V_i - V_j \in \mathcal{G}$ do
26:          if orientating $V_i \rightarrow V_j$ causes cycles or v-structures in $\mathcal{G}$ then
27:              if orientating $V_i \leftarrow V_j$ causes cycles or v-structures in $\mathcal{G}$ then
28:                  orientate $V_i \leftrightarrow V_j$ in $\mathcal{G}$
29:              else
30:                  orientate $V_i \leftarrow V_j$ in $\mathcal{G}$
31:              end if
32:          end if
33:      end for
34:  until $\mathcal{G}$ is unchanged
35:  return $\mathcal{G}$
36: end procedure
```

separating set with the highest p-value to determine if an unshielded triple is a v-structure in the v-structure phase. Similarly, Hyttinen et al. [2014] designed a weight scheme $w(k)$ to reflect the reliability for each conditional (in)dependence constraint k that is detected from data by CI tests. They define the optimal graph $\mathcal{G}^*$ as the graph that minimises the sum of weights of conditional (in)dependence constraints not implied by $\mathcal{G}^*$ as described in Equation 2.19, which can be solved by applying the off-the-shelf Answer Set Programming (ASP) algorithm [Gelfond and Lifschitz, 1988, Niemelä, 1999, Simons et al., 2002]:

$$\mathcal{G}^* = \arg\min_{\mathcal{G}} \sum_{k \in \boldsymbol{K} : k \notin \mathcal{G}} w(k), \tag{2.19}$$

where $\boldsymbol{K}$ represents the set of all conditional (in)dependence constraints that are testable from data. The weight $w(k)$ for constraint $k = V_i \perp\!\!\!\perp V_j \mid \boldsymbol{S}$ given a data set D is defined as:

$$w(k) = \log P(k \mid D) - \log(1 - P(k \mid D)) \tag{2.20}$$

$$P(k \mid D) = \frac{P(V_j \mid \boldsymbol{S})\,\alpha}{P(V_j \mid \boldsymbol{S})\,\alpha + P(V_j \mid V_i, \boldsymbol{S})(1-\alpha)}, \tag{2.21}$$

where α represents the prior of k, $P(V_j \mid \boldsymbol{S})$ and $P(V_j \mid V_i, S)$ correspond to the local scores used in score-based algorithms for V_j given parents $\boldsymbol{S}$ and $\{V_i\} \cup \boldsymbol{S}$ respectively.

Li et al. [2019] found that the separating set $\boldsymbol{S}$ of V_i and V_j detected by PC during the adjacency phase is likely not to be consistent with the final learnt graph, and they call this set *inconsistent separating set* in their paper. They propose another variant of PC that recursively executes a modified version of PC-stable to ensure the discovered separating sets are consistent with the final learnt graph. They modified the adjacency phase in PC-stable as follows:

1. Set the initial graph to $\mathcal{G}_1$.

2. Replace $a(V_i) \setminus \{V_j\}$ by $a(V_i) \setminus \{V_j\} \cap \text{Consist}(V_i, V_j \mid \mathcal{G}_2)$, where $\text{Consist}(V_i, V_j \mid \mathcal{G}_2)$ is defined as:

$$\begin{aligned} \text{Consist}(V_i, V_j \mid \mathcal{G}_2) = \{Z \in \mathit{Adj}(\mathcal{G}_2, V_i) \setminus \{V_j\} \mid & 1.\text{At least one path } \gamma^{Z}_{V_i V_j} \text{ exists in } \mathcal{G}_2; \\ & 2.Z \text{ is not a child of } V_i \text{ in } \mathcal{G}_2\}, \end{aligned} \tag{2.22}$$

   where $\gamma^{Z}_{V_i V_j}$ is a path between V_i and V_j through Z.

They name the above modified adjacency phase as $\text{NewStep1}(\mathcal{G}_1 \mid \mathcal{G}_2)$. Their modification first calls $\text{NewStep1}(\mathcal{G}_c \mid \mathcal{G}_\emptyset)$ to obtain $\mathcal{G}_0$, and then recursively calls $\text{NewStep1}(\mathcal{G}_0 \mid \mathcal{G}_{k-1})$ based on the graph $\mathcal{G}_{k-1}$ obtained in previous iteration $k-1$. Finally, they call the v-structure phase and orientation phase of PC-stable and do a final consistency check to ensure all separating sets are consistent with the final graph.

The Grow-Shrink (GS) algorithm [Margaritis and Thrun, 1999] is the first algorithm to utilise the concept of *Markov Blanket* (MB) to reduce the possible number of CI tests. GS contains two phases. In the first phase, the Grow phase, it dynamically adds variables to the candidate MB set that are conditionally dependent with the target variable. In the second phase, the Shrink phase, it removes the variables that do not belonging to the MB of the target variable. The pseudocode of the GS algorithm is presented in the Algorithm 5.

Algorithm 5 The Grow-Shrink (GS) algorithm

```
1: procedure GS($X, \boldsymbol{S}, D$)
       Input: target variable $X$, candidate variables set $\boldsymbol{S}$, data $D$
       Output: Candidate Markov Blanket $CMB$ of $X$
2:     $CMB \leftarrow \varnothing$
       ▷ Grow Phase
3:     repeat
4:         if $\exists S_i \in \boldsymbol{S}, s.t. X \not\perp\!\!\!\perp S_i \mid CMB$ then
5:             add $S_i$ to $CMB$
6:             remove $S_i$ from $\boldsymbol{S}$
7:         end if
8:     until $CMB$ stays unchanged
       ▷ Shrink Phase
9:     for each $Y \in CMB$ do
10:        if $X \perp\!\!\!\perp Y \mid CMB \setminus \{Y\}$ then
11:            remove $Y$ from $CMB$
12:        end if
13:    end for
14:    return $CMB$
15: end procedure
```

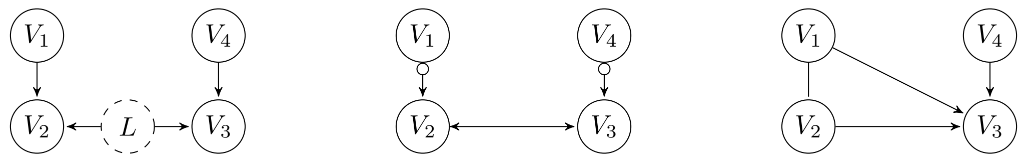
The *Incremental Association Markov Blanket* (IAMB) algorithm [Tsamardinos et al., 2003b] optimises the GS algorithm so that it can handle thousands of variables. The authors argue that GS's Grow phase is suboptimal because it is slow to discover spouses in the MB since these often have weak association. This, in turn, leads to more CI tests in the Grow and Shrink phases. Instead, they propose using conditional mutual information to determine the order in which a node is considered for inclusion into CMB during the Grow phase. They also propose a variant of IAMB, called *Inter-IAMB*, which interleaves the Grow and Shrink phases.

2.2.4 Hybrid structure learning

The *Max–Min Hill Climbing* (MMHC) algorithm proposed by Tsamardinos et al. [2006] is one of the earliest hybrid structure learning algorithms. It starts with the *Max–Min Parents Children* (MMPC) [Tsamardinos et al., 2003a] to construct the graph skeleton. MMPC is a variant of the constraint-based PC algorithm by prioritising the variable pair based on their strength of association. In the subsequent score-based phase, MMHC uses HC to produce the DAG, but is constrained to only consider edges that are present in the graph skeleton produced during the constraint-based phase.

Gasse et al. [2014] introduced the *Hybrid HPC* (H2PC) algorithm which uses the *Hybrid Parents and Children* (HPC) algorithm to construct the graph skeleton in the constraint-based phase, with a specific focus on minimising false negative edges. HPC incorporates an ensemble of weak parent-and-children algorithms to achieve this objective. In the score-based phase, H2PC employs tabu search. Evaluations across 10 networks, ranging up to 1836 variables, demonstrate that H2PC outperforms MMHC in terms of both structural accuracy and data fitting. However, it should be noted that H2PC was shown to be considerably less efficient than MMHC, and approximately 10 times slower for larger sample sizes.

Ogarrio et al. [2016] proposed a hybrid algorithm called *Greedy Fast Causal Inference* (GFCI) to address the challenge of latent confounders and provide asymptotic guarantees of correctness.



(a) True DAG with an unobserved common cause L (b) True PAG over observed variables (c) Output of fGES

Figure 2.4: The output of fGES when the true DAG contains unobserved common cause.

They observed that constraint-based algorithms like FCI demonstrate asymptotic correctness when assuming causal sufficiency, but exhibit poorer performance when sample size is low. On the other hand, score-based algorithms like GES and fGES are asymptotically correct when assuming causal sufficiency, but tend to introduce additional adjacencies and incorrect orientations in the presence of latent variables [Ogarrio et al., 2016]. Figure 2.4 illustrates an example where fGES generates CPDAG given data sampled from a system with an unobserved common cause. In this example, fGES introduce extra adjacency, i.e., $V_1 \rightarrow V_3$, and incorrect orientations, i.e., $V_1 - V_2, V_2 \rightarrow V_3$, because it is impossible to find a CPDAG without hidden variables while exactly representing the conditional independencies among the observed variables in the true DAG. To overcome these limitations, GFCI starts by employing GES to generate a CPDAG, and then utilises CI tests to remove unnecessary adjacencies from the skeleton. Finally, modified FCI orientation rules are applied to produce a PAG. The authors evaluate GFCI using synthetic Gaussian BNs with varying numbers of variables (10^2 or 10^3) and different percentages (5% or 20%) of latent variables. The results show that GFCI generally achieves better recall and precision in terms of adjacencies and arrow end marks compared to FCI. In the cases where GFCI performed slightly worse, the difference was negligible.

Jabbari et al. [2017] introduced a hybrid version of the RFCI constraint-based algorithm known as RFCI-BSC. This variant incorporates a novel scoring approach called *Bayesian Scoring of Constraints* (BSC), which assigns a score to each CI test based on the probability of true conditional independence. In RFCI-BSC, the RFCI algorithm is adapted to make stochastic decisions on the truthfulness of each CI based on its corresponding BSC score. It is important to highlight that RFCI-BSC operates in a non-deterministic manner as it utilises a different random seed for each run.

Constantinou [2020] proposed the SaiyanH hybrid algorithm, which starts with constraint-based learning to produce a skeleton with no disjoint subgraphs, which can be viewed as a denser version of the maximum spanning tree. In the subsequent phase, it orientates all edges by combining CI tests with BIC and a function that maximises the impact of hypothetical interventions. It then applies tabu search to the resulting DAG, with the restriction not to eliminate edges that would result in disjoing subgraphs, to ensure full propagation of evidence when converting the learnt graph into a BN model.

2.2.5 Structure learning for functional causal models

This subsection describes algorithms that learn the structure of FCMs, which can be seen as a special case of continuous BN that expresses the relationship between causes $\boldsymbol{Pa}_i$ and effects V_i via the function:

$$V_i = f\left(\boldsymbol{Pa}_i, \epsilon_i\right), \tag{2.23}$$

where ϵ_i is the noise (disturbance) term of V_i which is assumed to be independent of $\boldsymbol{Pa}_i$. Given linear functions f and Gaussian noise terms ϵ_i, a FCM is equivalent to a general linear Gaussian BN. If the function f is nonlinear or the noise ϵ_i is non-Gaussian, then the underlying FCM is possible to be fully identified from observational data when the sample size moves to infitine [Shimizu et al., 2006, Hoyer et al., 2008, Zhang and Hyvärinen, 2009].

One of the earliest structure learning algorithms for FCMs is the Linear Non-Gaussian Acyclic Model (LiNGAM) [Shimizu et al., 2006] which assumes that the values of a variable are determined by a linear function of its parents with additive non-Gaussian noise. Given this assumption, the data generating process of V_i can be expressed as follows:

$$V_i = \sum_{V_j \in \boldsymbol{Pa}_i} b_{ij} V_j + \epsilon_i. \tag{2.24}$$

The Equation 2.24 can be rewritten in matrix form as follows:

$$\boldsymbol{V} = \boldsymbol{B}\boldsymbol{V} + \boldsymbol{E},$$

$$\boldsymbol{E} = (\boldsymbol{I} - \boldsymbol{B})\,\boldsymbol{V}, \tag{2.25}$$

where $\boldsymbol{B}$ is the matrix that contains the coefficients b_{ij} and $\boldsymbol{E}$ is the vector of independent noise terms ϵ_i. By assuming that $\boldsymbol{E}$ is composed by non-Gaussian components, LiNGAM estimates the coefficient matrix $\boldsymbol{W}$ from observed data in two steps. Firstly, LiNGAM applies standard Independent Component Analysis (ICA) [Comon, 1994] on data $\boldsymbol{V}$ to extract non-Gaussian components $\boldsymbol{S}$:

$$\boldsymbol{S} = \boldsymbol{W}\boldsymbol{V}. \tag{2.26}$$

However, the order of $\boldsymbol{S}$ retrieved by ICA is not fixed and may not correspond to the variable order in $\boldsymbol{V}$, i.e., the i-th component in $\boldsymbol{S}$ may not be the noise term for V_i, such that one cannot directly use $\boldsymbol{W}$ to reconstruct a BN. Therefore, LiNGAM permutes, normalises and re-scales $\boldsymbol{W}$ to be as close as possible to a lower triangular matrix which becomes an estimation of $\boldsymbol{B}$. Finally, one can leverage the estimated $\boldsymbol{B}$ to derive the causal relationship between each pair of variables and produce a learnt DAG.

Hoyer et al. [2008] generalises LiNGAM to nonlinear scenarios by assuming that causal relationships between variables might be nonlinear and that the noise term is additive, which is also known as Additive Noise Model (ANM). The relationship between parents and children in an ANM can be written as follows:

$$V_i = f\left(\boldsymbol{Pa}_i\right) + \epsilon_i, \tag{2.27}$$

where f represents the possible nonlinear relationhips between parents $\boldsymbol{Pa}_i$ and children V_i. They find that non-linearity of functions is also conducive to identification of causal directions which is analogous to non-Gaussianity. Their theoretical results indicate that when the true DAG is $V_1 \rightarrow V_2$ and the noise terms for V_1 and V_2 are both Gaussian, then the causal direction $V_1 \rightarrow V_2$ can be identified from observational data as long as the causal relationship between V_1 and V_2 is nonlinear. Hoyer et al. [2008] then propose an exhaustive algorithm to find the optimal DAG by searching all possible DAGs. For each DAG, they perform nonlinear regression for each variable given its parents, and test if the residuals are mutually independent. If an independence test fails, the corresponding DAG is rejected. However, this method is only feasible when the number of variables is low (roughly less than or equal to 7).

Peters et al. [2014] propose the *Greedy DAG Search* (GDS) algorithm for ANM intended for larger graphs. GDS assumes the noise term of a variable is independent with the noise terms of all other variables. Therefore, it defines an objective function based on the measure of independence between the residual of a variable and the residuals of all other variables, and then performs greedy search in the search-space of DAGs like HC. The objective score of GDS can be described as:

$$\sum_i \mathrm{DM}\left(res_i^{\mathcal{G}}, res_{-i}^{\mathcal{G}}\right) + \lambda \# \left(\mathrm{edges}\right), \tag{2.28}$$

where DM is a measure of dependence, $res_i^{\mathcal{G}}$ is the residual of V_i given its parents in $\mathcal{G}$, and $res_{-i}^{\mathcal{G}}$ is the residuals of all variables except V_i. GDS uses the minus logarithm of the p-value of *Hilbert Schmidt Independence Criterion* (HSIC) as the dependence measure, and the Gaussian Process regression as the regression model.

Zhang and Hyvärinen [2009] focus on another type of FCM, called *Post-NonLinear* (PNL) causal model, and explore the conditions under which PNL is identifiable. PNL is a broader class of FCM that includes both LiNGAM and ANM as two special cases. PNL expresses variable V_i as follows:

$$V_i = f_{i,2}\left(f_{i,1}\left(\boldsymbol{Pa}_i\right) + \epsilon_i\right), \tag{2.29}$$

where $f_{i,1}$ represents the possible nonlinear effect of $\boldsymbol{Pa}_i$ and $f_{i,2}$ represents the possible post-nonlinear distortion on V_i. It turns out that ANM is a special case of PNL when assigning $f_{i,2}$ a linear function. Zhang and Hyvärinen [2009] found that PNL is generally identifiable in theory except for five special situations including the well known linear Gaussian case.

Chapter 3

The impact of data noise on structure learning, and learning structures from noisy data with model averaging

3.1 Introduction

Over the past few decades, various structure learning algorithms have been proposed with different learning strategies and hypotheses. These algorithms typically assume that the observed data is sampled from the true underlying causal graph and that the statistics derived from the data are asymptotically equivalent to those obtained from the true distribution. However, these assumptions are violated when learning from real-world data, resulting in a performance gap between synthetic experiments and real-world applications. Nonetheless, it remains unclear to what extent this performance gap exists.

One of the main challenges in measuring the performance of structure learning algorithms is that the ground truth causal graph on which the data set is generated from is generally unknown in practice. This limitation prevents us from assessing the capability of these algorithms in reconstructing the true graph. There has been a growing interest in empirically evaluating structure learning algorithms. Relevant studies include Djordjilović et al. [2017] who examine the performance of PC, GOBNILP, and K2 in retrieving gene networks from transcription data, Scutari et al. [2019] who analyse the differences in accuracy and speed of various structure learning algorithms available in the bnlearn R package [Scutari et al., 2010], and Assaad et al. [2022] who evaluate structure learning algorithms for time series problems using both synthetic and real-world data sets.

In this Chapter, we broaden the scope of our investigation by examining a wider range of algorithms from multiple structure learning packages, and specifically in the presence of data noise. We simulate four different types of data noise, including measurement error, missing values, merging states and latent variables, to investigate how data noise affects structure learning performance. The aim is to better approximate the real-world performance of these algorithms by incorporating

Case study	Description
Asia	A BN that models the relationships between variables related to the diagnosis of lung cancer [Lauritzen and Spiegelhalter, 1988].
Alarm	A BN model for monitoring critically ill patients in intensive care units [Beinlich et al., 1989].
Pathfinder	A BN model designed to mimic the decision-making capabilities of a human expert in diagnosing lymph-node diseases [Heckerman et al., 1992].
Sports	A BN designed for modelling and predicting the outcomes of football matches [Constantinou, 2022].
ForMed	A BN model for risk assessment and risk management of violent reoffending amongst mentally ill prisoners [Constantinou et al., 2015].
Property	A BN designed for modelling the dynamics of the London buy-to-let property market [Constantinou and Fenton, 2017].

Table 3.1: The description of the six case studies.

various types and levels of noise in the data that are assumed to be common in real data. We evaluate the performance of algorithms that come from different classes of structure learning, with and without data noise, and provide an overview of how different types of data noise impact their performance. Once the impact of data noise is established, we then show that combining model averaging with pruning and greedy search leads to improved structure learning performance when the input data are noisy. This chapter is organised as follows: Section 3.2 presents an empirical investigation regarding the impact of data noise on structure learning algorithms, Section 3.3 describes a novel model averaging algorithm suitable for learning BN structures in the presence of data noise, and Section 3.4 concludes the chapter with our final remarks.

3.2 Impact of data noise

3.2.1 Case studies and synthetic data

We consider six discrete BNs to generate synthetic data. These are the widely-known Asia, Alarm and Pathfinder networks, and three relatively recent case studies that come from real-world BN applications. The case studies range from 8 to 109 variables. We provide a brief description of the six BNs in Table 3.1. Note that while the parameters in the Asia, Alarm, and Pathfinder networks are based on knowledge, the parameters in Sports and ForMed networks are learnt from real data, whereas those in the Property network are derived from clearly defined tax rules and regulating protocols.

We use the BN models described above to generate both clean (i.e., noise-free) and noisy synthetic data. Noisy data are generated by making different assumptions about the type and rate of noise in the data, resulting in multiple data sets for each case study. These categories are listed below, where for each category we generate data sets with sample sizes $10^2, 10^3, 10^4$, and 10^5. These data sets are classified under the category "No Noise" (N). Next, each of these N data sets is manipulated with at least one type of data noise, resulting in the other five data categories as follows:

- Noise-free (N): The traditional approach used in the structure learning literature to generate noise-free synthetic data .

- Missing values (M): There is a probability of 5% or 10% (both percentages are tested) for every data entry to be missing completely at random. Since most algorithms cannot handle data sets that have empty cells, we introduce a new state called ”missing” to represent the missing values.

- Incorrect values (I): There is a probability of 5% or 10% (both percentages are tested) for every data entry to be substituted by one of the other available values in the same variable. For example, if a variable has possible states $\{x, y, z\}$, a value x could be randomly changed into either y or z.

- Merged states[1] (S): There is a probability of 5% or 10% (both percentages are tested) for every variable to have two of its states merged into one. For example, a variable with states $\{a, b, c\}$ would have two random states, such as a and b, both combined into a new state ab.

- Latent variables (L): There is a probability of 5% or 10% (both percentages are tested) for every variable to be removed from data.

- Combo (C): Represents the case where two or more types of data noise occur simultaneously. Due to the high computational complexity associated with all the experiments, we restrict the rate of noise to 5% for each type of data noise considered in these experiments which assume multiple types of data noise[2].

The above categories lead to 16 different configurations of data noise, including the noise-free case, as described in Table 3.2. Each configuration of data noise is applied to each of the 6 case studies, and 5 different sample sizes are generated for each case, which leads to a maximum of 480 data sets for each algorithm to be evaluated. Note that some of these categories cannot be applied to all case studies. For example, experiment S could not have been applied to Boolean BNs, such as the Asia BN, because a binary variable already contains the minimum number of possible states it can have. Moreover, the Asia and Sports BNs incorporate eight and nine nodes respectively, which means experiments S and L, which involve manipulation of variables rather than data values, could be performed only at 10% level of randomisation, since manipulating just one variable corresponds to a rate of >10% in both cases.

3.2.2 The investigated structure learning algorithms

We consider 15 structure learning algorithms that come from different classes of learning, including constraint-based, score-based and hybrid learning algorithms. Some of the selected algorithms are old and well-established in the literature, while others come from recently published papers. A list of the selected algorithms is provided in Table 3.3. Please refer to Chapter 2 for the description of each of these algorithms.

We utilised the R package *r-causal* [Wongchokprasitti, 2019] to test PC-stable, FCI, FGES, GFCI and RFCI-BSC, the R package *bnlearn* [Scutari et al., 2010] to test GS, Inter-IAMB, HC,

[1]Missing values and merged states are sometimes regarded as a special case of ’coarsening’ in which the observations are not exact values of data but rather simplifications or summarisations of information [Heitjan and Rubin, 1991].

[2]The work in this section involved learning approximately 10,000 graphs with a total structure learning runtime of seven months.

		No Noise	Missing values		Incorrect values		Merged states		Latent variables	
			5%	10%	5%	10%	5%	10%	5%	10%
1	N	✓								
2	M5		✓							
3	M10			✓						
4	I5				✓					
5	I10					✓				
6	S5						✓			
7	S10							✓		
8	L5								✓	
9	L10									✓
10	cMI		✓		✓					
11	cMS		✓				✓			
12	cML		✓						✓	
13	cIS				✓		✓			
14	cIL				✓				✓	
15	cSL						✓		✓	
16	cMISL		✓		✓		✓		✓	

Table 3.2: Experiment codes used to represent the 16 specified configurations of noisy data investigated.

Category	Algorithm
Score-based	HC, TABU, FGES, GOBNILP, WINASOBS, NOTEARS
Constraint-based	PC-stable, FCI, GS, Inter-IAMB
Hybrid	MMHC, H2PC, GFCI, RFCI-BSC, SaiyanH

Table 3.3: The 15 structure learning algorithms investigated, categorised by learning class.

TABU, MMHC and H2PC, the *Bayesys* software [Constantinou, 2019] to test SaiyanH, the *GOBNILP* software [Cussens, 2011] to test GOBNILP, the *BLIP* software [Scanagatta, 2019] to test WINASOBS, whereas NOTEARS was tested using its original Python source code [Zheng et al., 2018]. While many of the implementations mentioned here provide the option for users to modify hyper-parameters that related to CI tests, objective functions, maximal in-degree, etc., there is no definitive guideline to assist users in selecting the optimal settings. Given the large number of experiments conducted in this Chapter, we opted to use the default settings for these hyper-parameters in each implementation to avoid introducing further variability, under the assumption that this is how many users would also apply these algorithms to real data.

3.2.3 Evaluation

In this study, we were interested in investigating the usefulness of these algorithms in real-world settings where we tend to require causal BNs to answer interventional or counterfactual queries. On this basis, the assessment is driven by how well the algorithms achieve this objective by recovering the true causal DAG that was used to generate the data. Therefore, we evaluate the performance of these algorithms in terms of recovering the true DAGs or MAGs, rather than recovering the true CPDAGs or PAGs. When there are hidden variables in the input data, the algorithms are evaluated with reference to the true MAG; otherwise, they are evaluated with reference to the true

True graph	Learnt graph	Weight	Reasoning
$V_i \rightarrow V_j$	$V_i \rightarrow V_j, V_i \text{o} \rightarrow V_j$	0	Complete match
$V_i \rightarrow V_j$	$V_i \leftrightarrow V_j, V_i - V_j, V_i \text{o} - \text{o}V_j, V_i \leftarrow V_j, V_i \leftarrow \text{o}V_j$	0.5	Partial match
$V_i \rightarrow V_j$	$V_i \perp\!\!\!\perp V_j$	1	No match
$V_i \leftrightarrow V_j$	Any edge	0	Latent confounder
$V_i \perp\!\!\!\perp V_j$	$V_i \perp\!\!\!\perp V_j$	0	Complete match
$V_i \perp\!\!\!\perp V_j$	Any edge	1	Incorrect dependency

Table 3.4: The penalty weights assumed by the F_1 and SHD metrics.

DAG. The differences between these two types of graphical representation can be found in Section 2.1.

We use the F_1 and SHD graphical metrics to evaluate the accuracy of the learnt graphs with reference to the true graphs. The description of these metrics can be found in Subsection 2.2.1. We use the penalty weights depicted in Table 3.4 to score each of the learnt graphs. Note that the original definition of SHD assumes that reversing an arc results in an increase of the SHD score by 1. However, in this Chapter, we halve the penalty under the assumption that the dependency has been correctly identified, albeit with the wrong orientation. This assumption is consistent with other SHD variants [de Jongh and Druzdzel, 2009].

3.2.4 Results and discussion

Because the results are collected from almost 10^5 individual runs, we introduce a runtime limit of six hours per run. Algorithms that fail to return a result within this specified time limit are assigned the lowest rank for that particular data set. For instance, if five out of the 15 algorithms fail to return a graph for a given data set, then those five algorithms would be given rank 11 for that particular data set. However, the GOBNILP algorithm was an exception to this rule, because it provides the option to users to stop the learning process within a given time and retrieve the best graph discovered up to that point. This feature of GOBNILP, which is not offered by other implementations, can be considered as an unfair advantage for GOBNILP in our results. Additionally, algorithms that fail to generate a graph due to errors are also assigned the lowest rank for that particular data set. While this was necessary to ensure the results remain unbiased across all experiments, this assumption does negatively influence the rankings of the algorithms that fail to produce a result. RFCI-BSC, FCI and PC-stable are the three algorithms that fail in most cases. As shown later, RFCI-BSC ranks lowest across all algorithms and over all experiments, whereas PC-stable and FCI rank in the middle of the ranking table, which means they are likely to perform better in experiments that they do produce a result.

To measure the impact of data noise, we separate the results into those with no synthetic noise (i.e., experiment N) which represents the typical approach to generating synthetic data, and those with different types of synthetic noise (i.e., all other experiments) based on the methodology described in Sections 2.2.2 to 2.2.4. This enables us to rank the algorithms over different case studies, types of data noise, sample size, and evaluation metrics.

Table 3.5 reports the average and overall ranks achieved by each algorithm in experiments N, as determined by each of the two scoring metrics. We report the overall rank rather than the overall score since not all algorithms produce a result in every single experiment. Averaging across scores that include missing scores would have biased the difference between scores, whereas assigning

Category	Algorithm	F_1 rank		SHD rank	
		Average	Overall	Average	Overall
Constraint-based	PC-stable	8.10 ± 3.83	11	6.83 ± 4.06	8
	FCI	7.70 ± 3.92	9	6.57 ± 3.96	7
	GS	11.87 ± 2.03	14	10.43 ± 3.36	13
	Inter-IAMB	10.00 ± 2.94	12	8.60 ± 3.31	12
Score-based	HC	3.63 ± 3.72	2	4.77 ± 4.06	2
	TABU	3.27 ± 3.07	1	4.43 ± 3.55	1
	FGES	7.50 ± 2.38	8	7.83 ± 2.84	10
	GOBNILP	4.80 ± 3.40	3	6.43 ± 4.55	5
	WINASOBS	6.30 ± 3.85	6	5.87 ± 3.58	4
	NOTEARS	12.00 ± 4.00	15	13.00 ± 2.00	15
Hybrid	GFCI	6.87 ± 2.28	7	6.87 ± 2.50	9
	RFCI-BSC	11.50 ± 2.73	13	10.90 ± 3.56	14
	MMHC	7.77 ± 2.74	10	6.47 ± 2.86	6
	H2PC	6.13 ± 3.79	5	5.10 ± 4.00	3
	SaiyanH	5.33 ± 2.26	4	8.00 ± 3.50	11

Table 3.5: Average and overall ranked performance of the algorithms over all case studies and sample sizes in experiment N (i.e., no noise), where the average rank is presented with its standard deviation.

ranks as discussed above avoids this bias. Overall, the results in Table 3.5 show that score-based algorithms perform better in experiment N, compared to constraint-based and hybrid algorithms.

Table 3.6 shows the average and overall ranked performance for each algorithm over all the 15 noisy experiments, and with reference to their ranked performance in clean experiments N. While in the clean experiments the TABU algorithm ranks first across both metrics, in the noisy experiments the HC algorithm ranks 1st in terms of F_1 and 2nd in terms of SHD score. This is unusual given that TABU returns a graph that is expected to be at least as good as the HC graph; although this is judged by the objective function. This observation suggests that data noise distorts model fitting which in turn has a negative effect on algorithms that optimise for fitting. On the other hand, the relative performance of SaiyanH - which also employs Tabu search- remains unaffected by data noise, and this suggests that hybrid algorithms that employ multiple learning strategies may be less sensitive to data noise.

It is worth noting that MMHC stands out as the only algorithm that has shown significant improvement in relative ranking with noisy experiments. Specifically, its overall ranking increases from 10th to 6th place in terms of F_1 score, and from 6th to 1st place in terms of SHD score. However, it is important to mention that the algorithm's high ranking in SHD score can be attributed to its tendency to produce fewer edges than other algorithms, and the SHD metric is known to favour sparse graphs. This partly explains why the F_1 rankings contradict the SHD rankings. In contrast, FCI experienced the largest decline in relative performance, despite the fact that it is designed to handle latent variables; i.e., one of the types of data noise simulated. Interestingly, the results for PC-stable highlight a contradiction between the two metrics. Specifically, while F_1 suggests that PC-stable has improved its relative performance under data noise, the SHD metric suggests otherwise. This illustrates how different evaluation metrics can lead to varying conclusions.

Finally, Figure 3.1 presents the overall decrease in accuracy over all algorithms for each noisy experiment, with respect to the clean experiments N. The impact of merging states and latent

Category	Algorithm	F_1 rank		SHD rank	
		Average	Overall	Average	Overall
Constraint-based	PC-stable	7.59 (+0.51)	10 (+1)	7.87 (-1.03)	10 (-2)
	FCI	8.67 (-0.97)	11 (-2)	8.67 (-2.11)	12 (-5)
	GS	11.74 (+0.12)	15 (-1)	9.54 (+0.89)	13 (+0)
	Inter-IAMB	9.79 (+0.21)	12 (+0)	7.82 (+0.78)	9 (+3)
Score-based	HC	3.60 (+0.03)	1 (+1)	4.92 (-0.15)	2 (+0)
	TABU	3.62 (-0.35)	2 (-1)	4.99 (-0.56)	4 (-3)
	FGES	7.15 (+0.35)	8 (+0)	7.12 (+0.71)	8 (+2)
	GOBNILP	5.17 (-0.37)	3 (+0)	6.72 (-0.28)	6 (-1)
	WINASOBS	6.54 (-0.24)	7 (-1)	5.49 (+0.38)	5 (-1)
	NOTEARS	11.65 (+0.35)	14 (+1)	12.83 (+0.17)	15 (+0)
Hybrid	GFCI	7.26 (-0.39)	9 (-2)	6.91 (-0.04)	7 (+2)
	RFCI-BSC	11.50 (+0.00)	13 (+0)	11.05 (-0.15)	14 (+0)
	MMHC	6.51 (+1.26)	6 (+4)	4.66 (+1.81)	1 (+5)
	H2PC	5.66 (+0.47)	5 (+0)	4.96 (+0.14)	3 (+0)
	SaiyanH	5.27 (+0.06)	4 (+0)	7.87 (+0.13)	11 (+0)

Table 3.6: Average and overall ranked performance for each algorithm over all case studies and sample sizes across all noisy experiments. The numbers in brackets represent the difference in performance with respect to the clean experiments N. Green and red text indicate increased and decreased relative ranked performance respectively.

variables is found to be minimal, resulting in a less than 10% decrease in overall accuracy for the algorithms evaluated. However, missing values and incorrect values have a much stronger impact on learning accuracy. For example, experiment I10 caused a decrease in learning accuracy of approximately 27% according to the F_1 metric, and a staggering 97% according to the SHD metric. The decrease in accuracy was more pronounced in the SHD results than in the F_1 results since the SHD metric represents error whereas the F_1 metric represents accuracy, and this led to some inconsistent conclusions. For example, SHD suggests that experiments I5 and cMI have a stronger negative impact than experiments cMISL which contain all types of noise. In contrast, the F_1 metric appears to have correctly identified that experiment cMISL causes the largest decrease in performance amongst the 15 experiments tested.

In general, the results suggest that score-based algorithms perform better than constraint-based and hybrid algorithms in both noise-free and noisy experiments, which is largely in agreement with the findings reported by Scutari et al. [2019]. Specifically, the HC, TABU and GOBNILP algorithms claimed the top three spots in terms of overall performance based on the F_1 metric, with or without data noise. However, other score-based algorithms, such as the FGES and WINASOBS algorithms, were positioned in the middle or bottom in terms of overall performance in both noise-free and noisy experiments.

The hybrid algorithms seemed to perform worse than score-based algorithms but better than constraint-based algorithms. SaiyanH is the highest-performing hybrid algorithm and ranked 4th in overall performance according to the F_1 metric, with and without data noise (although the SHD metric ranks it 11th). The rankings for H2PC, MMHC and GFCI are 5th, 6th to 10th and 7th to 9th respectively in terms of the F_1 score, with and without data noise, although they were ranked 3rd, 1st to 6th and 7th to 9th respectively in terms of SHD score. RFCI-BSC, despite being designed to account for latent variables during structure learning, performed poorly in the

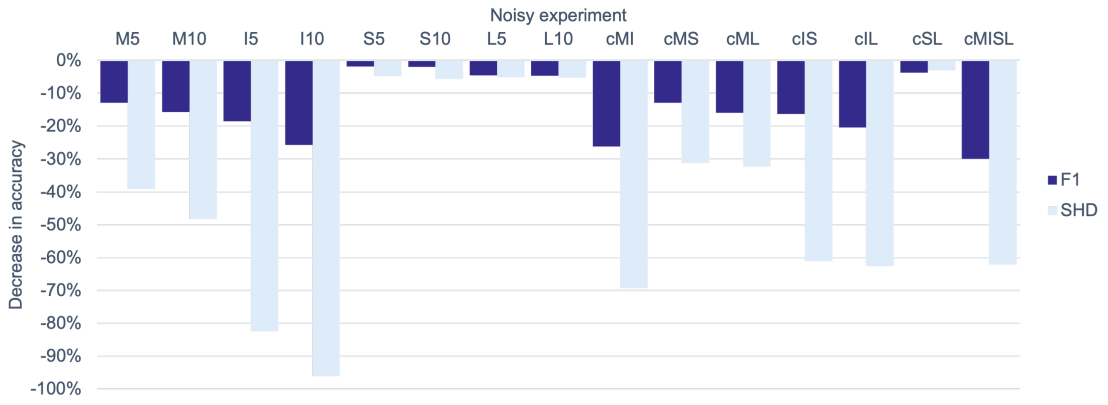


Figure 3.1: The overall decrease in accuracy over all algorithms for each noisy experiment, with respect to noise-free experiment.

experiments with and without data noise, and ranked 13th to 14th in terms of both metrics due to a high number of failures.

Lastly, constraint-based algorithms performed relatively poor overall. Even the best-performing constraint-based algorithms, PC-stable and FCI, ranked between 7th to 12th in all experiments based on both metrics, whereas the other constraint-based algorithms generally ranked lower than 10th. This could be attributed to the fact that we assume undirected edges that are true edges, to be equally wrong to true directed edges that are orientated in the wrong direction (i.e., a 0.5 penalty for both as indicated in Table 3.4), despite undirected edges having a 50% chance to be orientated correctly (not accounting for MEC considerations that could increase the success rate of the orientation).

3.3 Handling data noise by model averaging

In this section, we describe and evaluate the *Model Averaging Hill-Climbing* (MAHC) [Constantinou et al., 2022] which combines novel model averaging and pruning strategies with hill-climbing search to improve the reliability of structures learnt in the presence of data noise. Unlike conventional score-based algorithms that employ model selection functions and aim for the highest scoring graph, MAHC adopts a model averaging approach in which the learnt output reflects multiple candidate high-scoring graphs. While this is in direct contrast to exact structure learning algorithms like GOBNILP which guarantee to recover the highest scoring graph (albeit with restrictions), the motivation not to seek the highest scoring graph in the presence of data noise is supported by the results presented in the first part of this Chapter that highlight that maximising fitting in the presence of data noise is less desirable. This is because the highest scoring graph will be the one that best fits the data and the noise. Therefore, the hypothesis is that given the objective function becomes less useful in the presence of data noise, a model-averaging approach would be less sensitive to these errors.

3.3.1 The Model Averaging Hill-Climbing algorithm

The MAHC algorithm is an adaptation of the HC algorithm that incorporates two additional learning strategies. To begin with, the algorithm applies pruning to restrict the search space of

DAGs. The pruning phase results in a set of pruned arcs and pre-processed local scores which can be re-used during the subsequent phases of structure learning. The second extension involves the application of model averaging within the hill-climbing search space, which can be viewed as an extension of hill-climbing search where the depth level of the neighbouring graphs visited per hill-climbing iteration is increased by one level. Each neighbouring graph is assigned an average objective score, determined over the score of that graph and the scores of its valid neighbouring graphs. In other words, the model averaging phase optimises for the neighbouring graph that maximises the average score over a set of graphs, rather than the graph that has the highest score. This process, however, results in many more graphs visited, which in turn increases computational complexity substantially. This increase in computational complexity caused by the modified search method makes pruning necessary.

A crucial differentiation amongst pruning strategies lies in their soundness. A sound pruning strategy ensures that the pruned search space still includes the optimal graph. However, a sound pruning strategy would often remove a relatively small number of potential parent-sets, posing challenges for completing the model averaging process within a reasonable runtime. To address this issue, we propose a pruning strategy that can be viewed as an aggressive version of the pruning strategies that are typically applied to combinatorial optimisation structure learning problems, to significantly reduce the search space in exchange for a small decrease in the expected objective score of the learnt graph, which is less desired in the presence of data noise. Algorithm 6 outlines the pseudocode of the pruning strategy for a given variable V_i. MAHC would explore edges entering or leaving V_i consistent with those returned by Algorithm 6 for V_i.

Algorithm 6 The pruning strategy employed by MAHC

```
1: procedure PRUNING($V_i, D, mid$)
       Input: variable $V_i \in \boldsymbol{V}$, data set $D$, maximum in-degree for pre-processing $mid$
       Output: candidate parents set $CPS_i$ for $V_i$
2:     $CPS_i \leftarrow \boldsymbol{V} \setminus \{V_i\}$
3:     $C \leftarrow \emptyset$
4:     while $|C| < mid$ do
5:         for $V_j \in CPS_i \setminus C$ do
6:             if $S(V_i \mid C) > S(V_i \mid C \cup \{V_j\})$ then
7:                 $CPS \leftarrow CPS \setminus \{V_j\}$
8:             end if
9:         end for
10:        if $|CPS_i| > |C|$ then
11:            Add variable $V_l \in CPS_i \setminus C$ with the highest $S(V_i \mid V_l)$ score to $C$
12:        else
13:            break
14:        end if
15:    end while
16:    return $CPS_i$
17: end procedure
```

Once the pruning phase completes, MAHC moves to the model averaging phase. As discussed above, this phase involves visiting and evaluating valid neighbouring DAGs of a given valid neighbouring DAG. We can express this process through the objective function:

$$S_{MA}(\mathcal{G}, D) = \frac{1}{|nei(\mathcal{G})| + 1} \left(S(\mathcal{G}, D) + \sum_{\mathcal{G}_i \in nei(\mathcal{G})} S(\mathcal{G}_{\rangle}, D) \right), \tag{3.1}$$

where $nei(\mathcal{G})$ is the set of valid DAGs which can be reached by one edge modification from $\mathcal{G}$. Recall that edge modifications must adhere to the results obtained during the pruning phase. For example, if variable V_2 is pruned off from the candidate parent-set of V_1, then DAGs containing edge $V_2 \rightarrow V_1$ will not be explored or considered as a valid neighbouring DAG.

3.3.2 Evaluation, results and discussion

In this subsection, we assess the performance of the MAHC algorithm relative to various other structure learning algorithms across different case studies, sample sizes, data noises, and evaluation metrics. We set the parameter *mid* in the pruning phase to 3; implying that pruning decisions will be derived by exploring parent-sets up to size 3 (or maximum node in-degree of 3). We consider seven algorithms for comparison; the constraint-based FCI and PC-stable, the exact score-based learning GOBNILP, the approximate score-based learning HC and TABU, and the hybrid MMHC and SaiyanH algorithms. A description of these algorithms is given in the Sections 2.2.2 to 2.2.4. For data input, we consider both conventional clean data and the noisy dataset from experiment cMISL described in Subsection 3.2.1. This means that we employ the same experimental settings as those presented in Subsection 3.2.3. However, the evaluation in this section focuses on judging the proposed algorithm in terms of how well it recovers the true CPDAG, whereas in Section 3.2. we measured the impact of data noise in terms of recovering the true DAG structure.

Figure 3.2 presents the graphical accuracy as determined by F_1 and SHD, with and without data noise, across all case studies and sample sizes. The left charts focus on the results obtained from clean data and the right charts on the results obtained from noisy data. The percentage scores reflect the average relative difference in scores between the specified algorithm and MAHC across all case studies and sample sizes. The green negative percentage values indicate that the specified algorithm performed worse than MAHC. Conversely, the red positive percentage values indicate that the particular algorithm outperformed MAHC.

Figure 3.2 provides empirical evidence supporting the average superiority of MAHC over HC in terms of SHD and F_1 metrics, for both clean and noisy data. This finding demonstrates the advantages of model averaging in improving learning accuracy. Furthermore, the overall outcomes indicate that MAHC performs slightly better than TABU and MMHC across both metrics in both clean and noisy experiments. However, when compared to FCI, PC-stable and SaiyanH algorithms, MAHC only outperforms them in terms of the SHD metric in noisy data, while showing less favourable performance in other experimental settings. Finally, across both clean and noisy data, MAHC demonstrated inferior performance in comparison to the exact GOBNILP algorithm, as evaluated using both metrics.

Note that the results discussed in this subsection suggest that constraint-based learning performed better than score-based learning, contradicting the results presented in the previous subsections. This inconsistency occurs because here we compare the results between graphs for which both algorithms (MAHC and the competing algorithm) returned a result within the specified runtime limit, whereas in the previous subsections an algorithm that fails to produce a result within the specified runtime limit is assigned the lowest rank. This means that the results presented in this subsection are useful only in terms of comparisons with reference to MAHC, and not between learning classes or other algorithms.

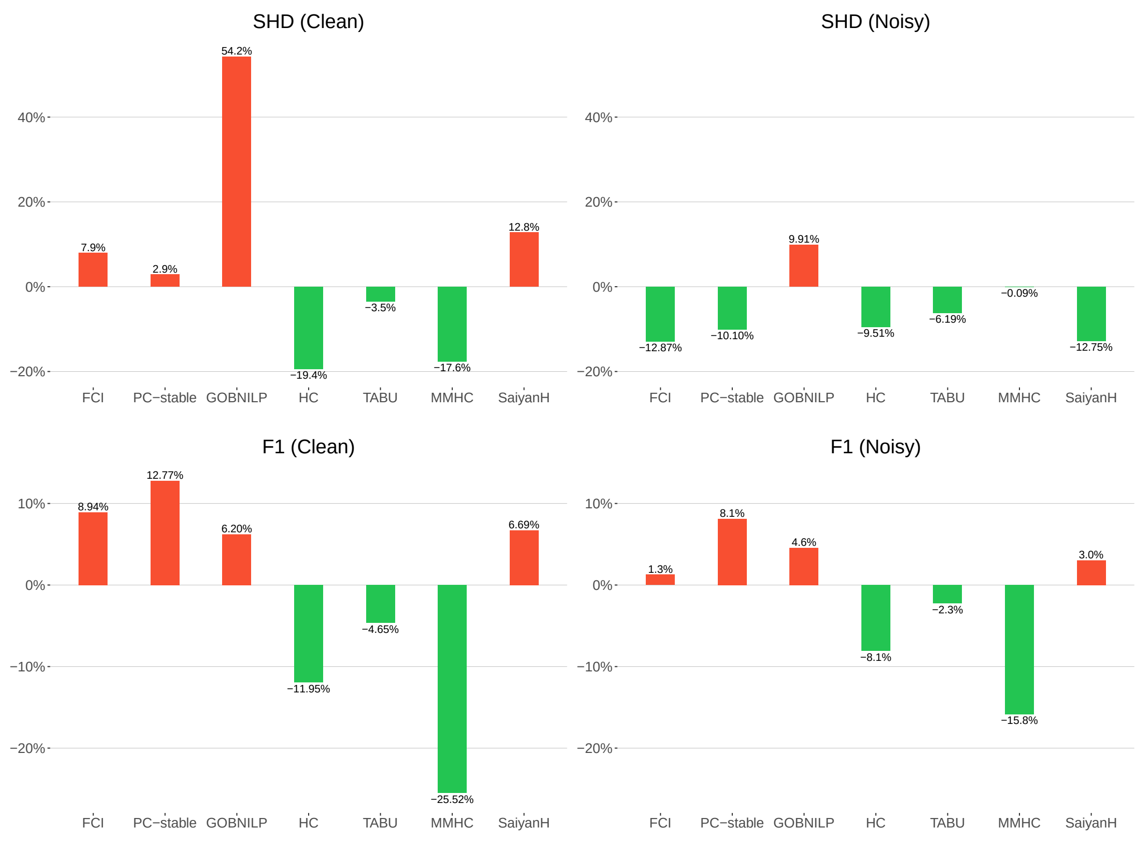


Figure 3.2: The structure learning accuracy of the seven other algorithms, relative to the accuracy of MAHC. A negative green score indicates that a particular algorithm performed worse than MAHC, while a positive red score indicates that a specific algorithm outperformed MAHC.

3.4 Concluding remarks

Structure learning algorithms are typically evaluated with synthetic data that satisfy the assumptions of the algorithms under evaluated. Some of these assumptions include causal sufficiency, causal faithfulness, and the completeness of the input data. Because these assumptions are unrealistic in practice, synthetic performance leads to an overestimation of the performance these algorithms might produce in practice. This Chapter investigates this research question.

According to the results depicted in Figure 3.1, the impact of latent variables (experiment L) and merging states (experiment S) on structure learning accuracy is relatively limited. In contrast, the presence of missing and incorrect data values (experiments M and I, respectively) have considerable impact on the performance of the structure learning algorithms. Specifically, when 5% or 10% of the data values are missing, the reduction in accuracy ranges from 13% to 18%, whereas for 5% or 10% of incorrect data values the reduction in accuracy ranges from 18% to 28%. When both types of data noise are combined into a single dataset, the decrease in accuracy ranges from 26% to 30%. When all four types of data noise are present in a single dataset, the decrease in accuracy ranges from 30% to 37%. These results imply that the BN structure learning accuracy reported in the literature, based on traditional synthetic data, overestimates real-world performance to a greater extent than previously assumed. Nevertheless, traditional noise-free synthetic experiments remain crucial in assessing BN structure learning algorithms under different hypothetical scenarios.

It is important to acknowledge that different algorithms exhibit varying degrees of sensitivity to different types of data noise. Initially, it was hypothesised that less complex or approximate algorithms would be more robust to data noise compared to exact or more complex algorithms. However, our findings (refer to Table 3.6) are mixed, with some results indicating that less complex algorithms like HC are more resilient to data noise compared to more sophisticated algorithms such as TABU, while other results indicate that data noise has a similar impact on the performance of exact GOBNILP and other non-exact algorithms. Noteworthy observations include H2PC and MMHC which appear to be less sensitive to data noise than other algorithms. Finally, while algorithms specifically designed to handle causal insufficiency such as FCI, GFCI, and RFCI-BSC perform well under noisy experiments that involve latent variables (i.e., experiments L), they do not perform as well as other algorithms under other types of data noise.

We also explored the efficacy of model averaging in mitigating the impact of data noise. To achieve this, we devised a novel objective function that involves averaging the scores of the evaluated graph and its neighbouring graphs. Subsequently, we introduced a variant version of the HC algorithm that utilises this objective function and incorporates a pruning phase to restrict the search space. Our results indicate that the integration of model averaging and pruning with hill-climbing search produce an algorithm that is less sensitive to data noise, and generates better results than algorithms that search for graphs with higher model selection scores.

Lastly, this study comes with some limitations. Firstly, some types of data noise could not be applied to all case studies. For example, *merging states* could not be applied to the Asia network that consists of boolean variables only. Secondly, due to the large number of experiments (those presented in Subsection 3.2.1 required a total structure learning runtime of seven months), all structure learning algorithms were tested with their default hyperparameters as provided by the software packages, under the assumption that this is how most users would employ them in practice. This means that we did not explore how hyperparameter sensitivity might influence these results.

Finally, the results of structure learning algorithms were evaluated against the corresponding true DAGs or MAGs. Since the true DAGs or MAGs are unlikely to be fully recoverable from pure observational data, some of the results produced by these algorithms might be sensitive to the order of the variables as read from data.

Chapter 4

Improving Structure Learning under Measurement Error

4.1 Introduction

Structure learning algorithms that learn the structure of a BN from observational data often do so by assuming the input data correctly reflect the true distribution of the variables. Most of the algorithms covered in Chapter 2 make this assumption. However, this assumption does not hold in the presence of measurement error, which can lead to spurious edges. This is one of the reasons why the synthetic performance of these algorithms often overestimates real-world performance.

The assumption of an underlying measurement error has only recently attracted attention in terms of its effect on BN structure learning. Scheines and Ramsey [2016] studied the effect of Gaussian measurement error on score-based FGES [Ramsey et al., 2017] and showed that even minor levels of measurement error can considerably deteriorate its accuracy. Traditionally, measurement error is generated and modelled under the assumption of Normally distributed and continuous data [Bollinger and van Hasselt, 2017]. In this Chapter, we assume the data are discrete and that the variables with measurement error are children of their underlying error-free version, and not parents of other variables. This assumption essentially makes them independent of other variables in the graph given their error-free version, which is desired.

Based on the above assumptions, we propose a heuristic score-based correction method, called the Spurious Edge Detection (SED) algorithm, for discrete BN structure learning. SED aims to identify and remove potential False Positive (FP) edges learnt by other structure learning algorithms, often in the presence of measurement error. SED can be added as an additional learning phase at the end of any structure learning algorithm, and serves as a correction learning phase that removes potential FP edges. The remainder of this Chapter is organised as follows: we discuss relevant works about structure learning in the presence of measurement error in Section 4.2, we provide the terminology and underlying assumptions relevant to this Chapter in Section 4.3, Section 4.4 illustrates the impact of measurement error on structure learning, Section 4.5 describes the proposed correction algorithm, Section 4.6 presents the results, and we provide our conclusions along with future research directions in Section 4.7.

4.2 Relevant works

Zhang et al. [2018] examined the conditions for identifying the underlying causal structure in the presence of measurement errors in observed data. They assumed that the error-free variables are generated by a LiNGAM model and discovered that while the complete recovery of the underlying LiNGAM was not feasible, a specific graphical structure called ordered group decomposition could always be identified from the observed data using overcomplete ICA.

Similarly, Dai et al. [2022] investigated the problem of measurement in LiNGAM and proposed the *Transformed Independent Noise* (TIN) condition to assess the independence between a linear transformation between two sets of measured variables. The TIN condition can be considered as a generalised version of the *Independent Noise* (IN) condition and the *Generalised Independent Noise* (GIN) condition. Dai et al. [2022] demonstrated that the ordered group decomposition could also be identified by employing an independence test based on the TIN condition.

Blom et al. [2018] proposed a method to estimate the upper bound of the variance of measurement error by assuming a linear Gaussian model as the true underlying model. They then utilised this estimated bound to determine the potential minimal and maximal values of partial correlations between variables, aiming to correct the results of CI tests. In their evaluation, they demonstrated that while there was a small difference between the baseline method and their correction approach, the corrections improved precision at lower recall levels in CI testing.

Yang et al. [2022] investigated the identifiability of linear Structural Equation Models (SEMs) when measurement error is present in data. They discovered a mapping between a linear SEM with variables containing measurement errors and a linear SEM with hidden variables that have no parents. Consequently, the identification of one model can be effectively applied to the other. Additionally, they demonstrated that, with the assumption of separability and a stronger version of the faithfulness assumption, it is possible to identify a linear SEM model from data even when the input data contain measurement errors.

4.3 Preliminaries

This section presents the preliminaries and the necessary terminology and assumptions used in this Chapter. We assume that the data are categorical and that every variable present in the data may be subject to measurement error. We refer to variables with measurement error as *noisy variables* and to variables without measurement error as *error-free variables*. Each noisy variable is assumed to be derived from its error-free version which is not present in the data. We denote the error-free variable as V_i where i represents the index of the error-free variable, and its corresponding noisy variable as V_i^*. When a variable is error-free, V_i is observed in the data, otherwise its noisy version V_i^* would be observed. We refer to *observed variable* as the one that is observed in the data. We use lowercase letters to represent the assignment of states where v_i^l denotes the lth state of variable V_i or its corresponding V_i^*. We define three types of graphs, i.e., error-free graph, noisy graph and learnt graph based on different involved variables and edges.

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

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

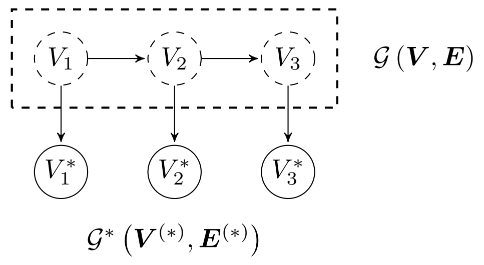
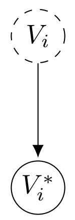

V_1
V_2
V_3
$\mathcal{G}(\boldsymbol{V}, \boldsymbol{E})$
V_1^*
V_2^*
V_3^*
$\mathcal{G}^*\left(\boldsymbol{V}^{(*)}, \boldsymbol{E}^{(*)}\right)$

Figure 4.1: A hypothetical graph that illustrates the relationship between the error-free variables $\boldsymbol{V}$ and the corresponding noisy variables $\boldsymbol{V}^*$ given the Independence rule, where a noisy variable V_i^* becomes independent of other variables in $\mathcal{G}^*$ given V_i.



$P(V_i^* \mid V_i)$	$V_i = 1$	$V_i = 2$	$V_i = 3$	$V_i = 4$
$V_i^* = 1$	0.9	0.1	0.1	0.01
$V_i^* = 2$	0.01	0.8	0.02	0.03
$V_i^* = 3$	0.03	0.08	0.85	0.01
$V_i^* = 4$	0.06	0.12	0.03	0.95

Figure 4.2: An example to illustrate how measurement error is represented for a discrete variable V_i with four states.

variables $\boldsymbol{V}^{(*)} = \boldsymbol{V} \cup \boldsymbol{V}^*$.

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

We assume that all graphs are DAGs and that the observed data are sampled from the observed variables that make up the noisy graph, independently and identically. Note that the observed data and learnt graphs are presented using the error-free variable names, since the algorithms are not given any information about which variables incorporate measurement error. We adopt the Markov, faithfulness and causal sufficiency assumptions introduced in Section 2.1. Additionally, since the variables are assumed to be discrete, we assume that a noisy variable V_i^* has only one parent, and that this parent represents its error-free version V_i[1]. This leads to the following *Independence rule*:

Assumption 4.3.1 (Independence rule)**.** *In the presence of measurement error, a noisy variable V_i^* is independent of other variables conditional on its error-free version V_i.*

Figure 4.1 presents a simple example that illustrates the relationship between error-free and noisy variables given the Independence rule, where each V_i^* becomes independent of the remaining nodes given its corresponding error-free parent V_i.

Therefore, we can use the conditional probabilistic distribution between V_i and V_i^* to represent the measurement error in data. Assume there are r_i states in V_i, then for each state $v_i^l \in V_i$, it has certain probability $P\left(V_i^* = v_i^k \mid V_i = v_i^l\right)$ to be recorded as another state v_i^k in V_i^*. Figure 4.2 presents an example to illustrate how the measurement error is represented for a discrete variable with four distinct states.

[1]In continuous cases, the noisy variable V_i^* is usually assumed to be dependent on two factors which are its error-free version V_i and an error term ϵ_i that represents the measurement error. Their relationship can be expressed as $V_i^* = f(V_i, \epsilon_i)$.

We denote the error rate α_i^l of state v_i^l for the noisy variable V_i^* as follows:

$$\alpha_i^l = 1 - P\left(V_i^* = v_i^l \mid V_i = v_i^l\right) \tag{4.1}$$

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

If the measurement error occurs randomly, we can assume the probability $P\left(V_i^* = v_i^k \mid V_i = v_i^l\right)$ is identical for all $l \neq k$. Besides, if the values of a variable are collected by the same measurement process, we can further assume the probability $P\left(V_i^* \mid V_i = v_i^l\right)$ is identical for different values v_i^l in V_i. Then, the conditional probabilistic distribution $P\left(V_i^* \mid V_i\right)$ can be simplified as follows:

$$P\left(V_i^* \mid V_i\right) = \begin{cases} \frac{\alpha_i}{r_i}, & \text{if } V_i^* \text{ has different value as } V_i \\ 1 - \alpha_i, & \text{if } V_i^* \text{ has the same value as } V_i \end{cases}, \tag{4.2}$$

where r_i represents the number of states in V_i.

4.4 Impact of measurement error on structure learning

This section illustrates that measurement error generally causes the structure learning algorithms to produce a higher number of spurious edges that tend to lead to a greater number of 3-vertex cliques, compared to the true number of such cliques in the ground truth graph. A clique is a set of nodes where each pair of nodes in the clique is adjacent. We first explain why this phenomenon occurs in theory, and then present the effect in practise by illustrating the empirical effect of measurement error on algorithms spanning all three classes of learning. Given the Causal Faithfulness assumption, the dependencies between variables are consistent with those entailed by applying d-separation rules on the BN. Therefore, we restrict the description about the effect of measurement error on d-connections and d-separations. For the unconditional (i.e., marginal (in)dependence) case, we derive Proposition 4.4.1.

Proposition 4.4.1. *The d-connection and d-separation relationships between two error-free variables V_1 and V_2 in a noisy graph $\mathcal{G}^*$ are consistent with the d-connection and d-separation relationships of their corresponding noisy versions V_1^* and V_2^*, given the Independence rule.*

Proof

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

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

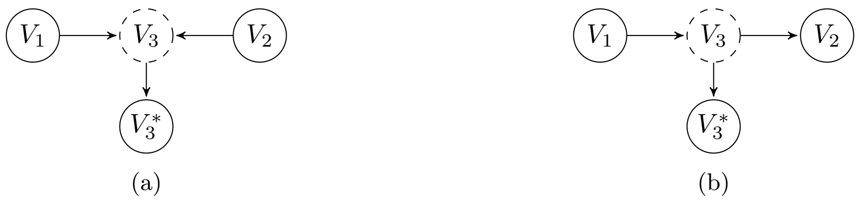


Figure 4.3: Modelling the presence of measurement error on the two different causal equivalence classes. Case (a) represents the common-effect class, where V_1 and V_2 are d-connected conditional on either V_3 or V_3^*, and (b) represents the causal-chain class where V_1 and V_2 are d-separated conditional on V_3, whereas they are d-connected conditional on V_3^* (this also holds for the causal class of common-cause).

■

According to Proposition 4.4.1, the unconditional (in)dependence relationship between noisy variables should be consistent with the unconditional (in)dependence relationship of their corresponding error-free variables given the Causal Faithfulness assumption. However, the conditional independence between error-free variables may not always hold for their corresponding noisy versions. Figure 4.3 illustrates two different causal classes with measurement error. Specifically, Figure 4.3a represents the causal class of common-effect where V_1 and V_2 are d-connected conditional on either V_3 or its noisy version V_3^*. Figure 4.3b represents the causal class of causal-chain where V_1 and V_2 are d-separated conditional on V_3, whereas they are d-connected conditional on V_3^* (this observation also holds for the causal class of common-cause).

These lead to Propositions 4.4.2 and 4.4.3 which state that although the conditional d-connection relations between noisy variables are consistent with those given by the error-free variables, it is likely that some conditional d-separations will no longer hold when the observed variables incorporate measurement error, as in the causal-chain example illustrated in Figure 4.4. In other words, under the large sample limit, the learnt graph will contain all the conditional dependence relationships that are entailed by the error-free graph, but may miss some conditional independence relationships that appear in the error-free graph.

Proposition 4.4.2. *In a noisy graph $\mathcal{G}^*$, if two error-free variables V_1 and V_2 are d-connected given a variable set $\boldsymbol{S}$, this d-connection will also hold for the noisy variables V_1^* and V_2^* conditional on the noisy variable set $\boldsymbol{S}^*$.*

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

According to Propositions 4.4.1 and 4.4.2, if two nodes V_i and V_j are neighbours in the noisy graph, they are d-connected conditional on any observed variable set in the noisy graph. Therefore, they will be neighbours in the learnt graph given the Causal Faithfulness assumption, and under large sample limit. However, some of the conditional independence relationships derived from the error-free graph might not hold in the observed data that contain measurement error and hence, would lead to spurious edges in the learnt graph.

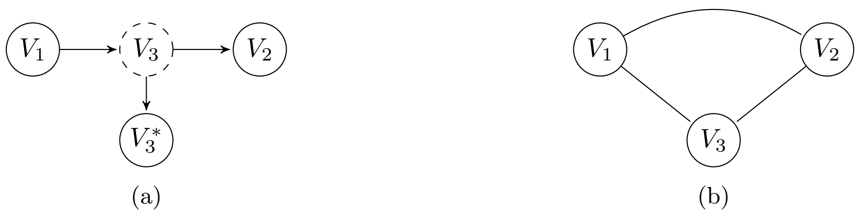


Figure 4.4: (a) A noisy graph that contains three error-free variables V_1, V_2 and V_3, and a noisy variable V_3^*. (b) The learnt graph that entails the same dependence and independence relationships derived from the observed variables.

Consider the simple noisy graph shown in Figure 4.4a which is composed by three error-free variables V_1, V_2 and V_3, and one noisy variable V_3^*. According to Propositions 4.4.1 and 4.4.2, and with reference to the example in Figure 4.4a, the unconditional and conditional dependencies between error-free variables V_1 and V_3 extent to their observed versions V_1 and V_3^*. Therefore structure learning algorithms tend to produce an edge between V_1 and V_3 in the graph learnt from observed data under large sample limit and similarly for V_2 and V_3. However, the only conditional independence relationship amongst the error-free variables, i.e., $V_1 \perp\!\!\!\perp V_2 \mid V_3$ would not hold when conditional on V_3^*. Therefore, we get $V_1 \not\perp\!\!\!\perp V_2 \mid V_3^*$ and the incorrect fully connected graph shown in Figure 4.4b as the learnt graph. In other words, the measurement error on an unshielded non-collider misleads structure learning algorithms towards producing a spurious edge between its neighbours, resulting in a 3-vertex clique. Note that structure learning can reconstruct up to the CPDAG $V_1 - V_3 - V_2$, or one of its corresponding DAGs, when the input data does not incorporate measurement error.

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

Proposition 4.4.3. *Given the causal faithfulness assumption and large sample limit, for any two non-adjacent error-free variables V_1 and V_2 in a noisy graph $\mathcal{G}^*$, if V_1 and V_2 are adjacent in the learnt graph $\mathcal{G}^l$, then there must be at least one connecting path p between V_1 and V_2 in $\mathcal{G}^*$. Besides, for each connecting path p, there is a 3-vertex clique $\{V_1, V_2, V_k\}$ in the learnt graph $\mathcal{G}^l$, where V_k is a variable in p.*

Proof Without loss of generality, we consider the situation where both V_1 and V_2 incorporate measurement error in the observed data. Since V_1 and V_2 are adjacent in the learnt graph $\mathcal{G}^l$ but not adjacent in the noisy graph $\mathcal{G}^*$, there must be at least one path p between V_1 and V_2 in $\mathcal{G}^*$ such that (i) no node in p is a collider, and (ii) all intermediate nodes in p incorporate measurement error. Otherwise, a set of observed variables could d-separate V_1^* and V_2^* in $\mathcal{G}^*$, and this would contradict with the adjacency between V_1 and V_2 in the learnt graph. If there is a connecting path $p = \{V_1, S_1, S_2, \ldots, S_n, V_2\}$, we can obtain another connecting path $p' = \{V_1, S_1, S_2, \ldots, S_n\}$ between V_1 and S_n. Thus, V_1^* and S_n^* would also be d-connected in the noisy graph given any observed variable set which leads to the presence of an edge between V_1 and S_n in the learnt graph. Besides, V_2 and S_n should still be adjacent in the learnt graph given the Proposition 4.4.1. Therefore, V_1, V_2 and S_n form a 3-vertex clique in the learnt graph. ■

We, therefore consider a 3-vertex clique as a sign for the presence of measurement error in at least one of the variables that make up the clique. When a learnt graph contains such a clique, we need

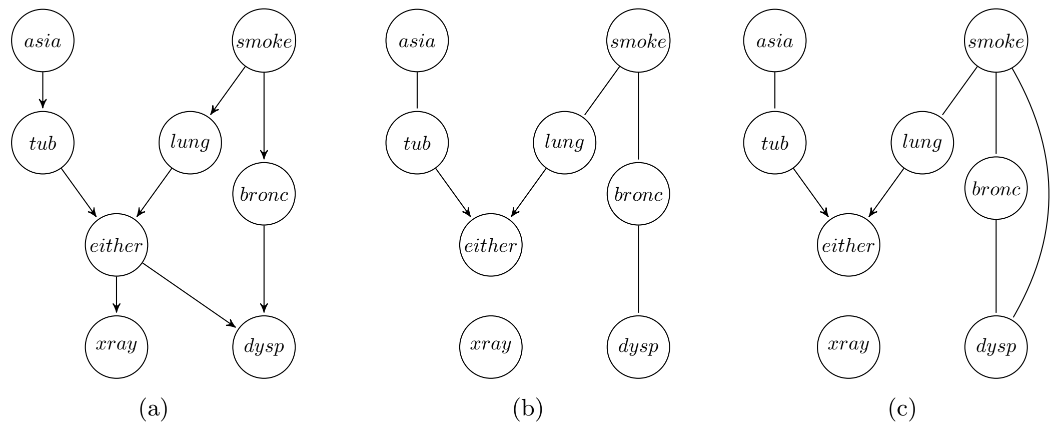


Figure 4.5: (a) The true Asia network. (b) The CPDAG learnt by PC-stable given the error-free synthetic data set. (c) The CPDAG learnt by PC-stable given the same synthetic data set but with 5% measurement error on variable *bronc*.

to determine whether the clique is in the error-free graph or the outcome of measurement error. If we could distinguish between these two possibilities, then we would be able to remove the spurious edges in the graph learnt from noisy data. This challenge can be viewed as a type of a hidden variable problem. In our case, a potential hidden variable represents the error-free parent of its corresponding noisy version.

Figure 4.5 presents an example based on the PC-stable algorithm and the classic Asia network[2]. Specifically, Figure 4.5a represents the true Asia network, Figure 4.5b represents the graph learnt from the error-free data set, and Figure 4.5c represents the graph learnt from the noisy data set with 5% measurement error on variable *bronc*, as defined by Equation 4.1, i.e., 5% of the value in *bronc* data are recorded by another valid but incorrect state. This relatively small rate of error has led to the spurious edge between *smoke* and *dysp*. This is because while *smoke* and *dysp* are independent conditional on the error-free variables *bronc* and *either*, this conditional independence is relaxed in the presence of measurement error on variable *bronc* and hence, the algorithm produces the additional FP edge. Moreover, this additional edge produces the 3-vertex clique $\{smoke, bronc, dysp\}$ that does not exist in the true Asia network nor in the graph learnt from the error-free data set.

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

Figure 4.6 compares the average number of 3-vertex cliques produced by each of the algorithms with and without measurement error, and with reference to the average number of 3-vertex cliques present in the ground truth graphs. These initial empirical results show that structure learning algorithms tend to produce more 3-vertex cliques in the graphs learnt from noisy data sets compared to both the true graph and the graphs learnt from error-free data sets. Moreover, score-

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

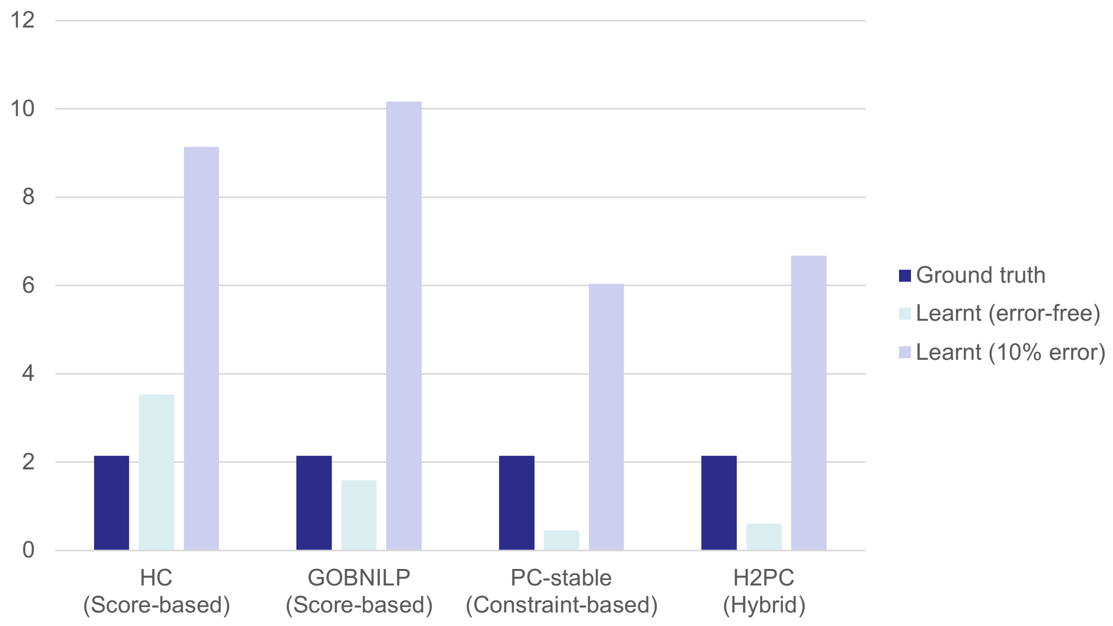


Figure 4.6: The average number of 3-vertex cliques in the ground truth graphs, the graphs learnt from error-free data sets, and the graphs learnt from observed data sets with 10% measurement error on each variable.

based learning seems to be more sensitive to the measurement error compared to constraint-based learning, although this observation is based on the default hyperparameters as implemented in the corresponding packages [Scutari et al., 2010, Wongchokprasitti, 2019, Cussens, 2011] that we have used to test the algorithms. These results support our hypothesis that a 3-vertex clique can be viewed as a sign for the presence of measurement error in the input data.

4.5 The Spurious Edge Detection (SED) algorithm

This section describes the Spurious Edge Detection (SED) algorithm which can be applied to the output graph produced by any other BN structure learning algorithm to discover and eliminate potential FP edges that tend to be the outcome of measurement error. The implementation of SED is available online [3]. Further to what has been discussed in Section 4.4, SED focuses its search for FP edges on the induced subgraph of 3-vertex cliques.

We consider every edge that connects two variables V_i and V_j in a 3-vertex clique $\{V_i, V_j, V_k\}$ in the learnt graph $\mathcal{G}^l$ to be a *candidate spurious edge*. According to Proposition 4.4.3, the node V_k is likely to be a variable on a connecting path between V_i and V_j in the underlying noisy graph $\mathcal{G}^*$, as long as V_i and V_j are not adjacent in $\mathcal{G}^*$. Therefore, the conditional independence between V_i and V_j is likely to be retrieved by introducing an error-free variable of V_k in the learnt graph, and treat the observed data of V_k as the observation of its noisy version. We define the Candidate Spurious Edge-nodes pair $CSE\,(E_i)$ for a candidate spurious edge E_i as the set of nodes that are

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

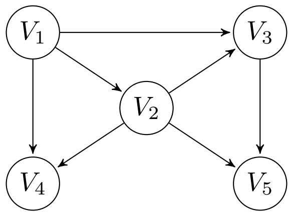


Figure 4.7: An example of a graph that contains 3-vertex cliques

adjacent to both the endpoints of E_i. For instance, the CSE for Figure 4.7 is:

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

Next, let us revisit the Asia network example in Figure 4.5c to investigate the possibility of a spurious edge in the presence of a single 3-vertex clique in the learnt graph. Recall that this is the graph learnt by PC-stable in the presence of 5% measurement error on variable *bronc*. Since the graph contains a single 3-vertex clique, the CSE for this graph is:

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

We then perform three graphical reconstructions as shown in Figure 4.8, one for each candidate spurious edge, given the Independence rule defined in Section 4.3, to identify and eliminate a spurious edge. During the graph reconstruction process, we build the reconstructed graphs that entail all the independences and conditional independences of the learnt graph, and test for an additional conditional independence relationship between the endpoints of the candidate spurious edge. For example, the graph in Figure 4.8a investigates the possibility of the edge $bronc - dysp$ being spurious and of the variable *smoke* incorporating measurement error[4], which is why it is replaced with a hidden unmeasured variable representing its error-free version, with the noisy version $smoke^*$ restructured as a child of the hidden variable. Moreover, the edge between *bronc* and *dysp* is removed such that the conditional independence $bronc \perp\!\!\!\perp dysp \mid smoke$ is introduced in Figure 4.8a, by assuming that the data for variable *smoke* are noisy, which could also explain the presence of clique $\{bronc, dysp, smoke\}$ in the learnt graph shown in Figure 4.5c. Similarly, Figures 4.8b and 4.8c repeat this process for the remaining two variables in clique $\{bronc, dysp, smoke\}$.

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

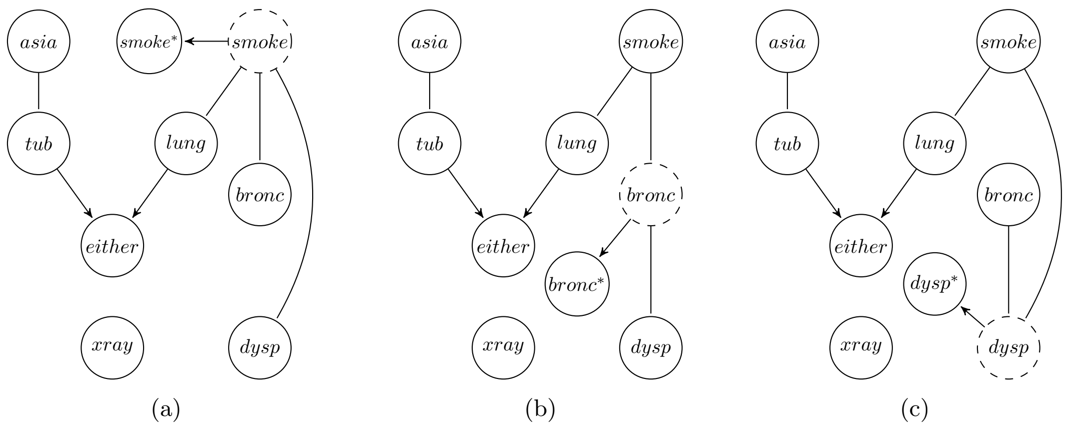


Figure 4.8: The three reconstructed graphs for clique $\{bronc, dysp, smoke\}$, based on the learnt graph in Figure 4.5c. Dotted nodes represent possible hidden error-free parents of the suspected noisy node under assessment.

Each reconstructed graph is then evaluated in terms of model selection between the learnt and observed distributions using the BIC score. While the true model is not present in the candidate collection of graphs, the BIC function should still select the model that converges with probability one to the quasi-true model as the sample size grows to infinite [Claeskens et al., 2008, Neath and Cavanaugh, 2012]. The quasi-true model in a candidate collection is the most parsimonious model that is closest to the true model, as measured by the Kullback-Leibler information. Therefore, when a reconstructed graph obtains a higher BIC score than the learnt graph, we consider the removal of that specific candidate spurious edge to produce a graph that is closer to the error-free graph. Because the reconstructed graphs include additional hidden variables, we adopt the Expectation-Maximization (EM) learning [Dempster et al., 1977] to compute the LL score of the BIC for each of the reconstructed graphs. Note that EM only guarantees to converge to a local optimum. This means that EM may perform arbitrarily poor in high dimensions and unable to find the best LL score estimation for the reconstructed graph. In other words, the estimated BIC score of the reconstructed graph might be lower than its true BIC score. If we assume that the measurement error on variable V_i occurs with equal probability for all its values, then the conditional distribution $P\left(V_i^* \mid V_i\right)$ contains only a single free parameter as shown in Equation 4.2, and this reduction in dimensionality increases the chance of EM converging to the global optimum. In the experiments carried out in this Chapter, we found that EM tends to converge to the global optimum when the initial value of the conditional probability table is set to $P\left(V_i^* \mid V_i\right)$ as identity matrix.

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

$$Q\left(\theta \mid \theta^t\right) = \sum_{m=1}^{N} \sum_{\boldsymbol{x}} P\left(\boldsymbol{X} = \boldsymbol{x} \mid D_m, \theta^t\right) \log P\left(\boldsymbol{X} = \boldsymbol{x}, D_m \mid \theta\right) , \tag{4.3}$$

where D_m represents the m-th case of data D. At the M step, the EM algorithm revises θ by maximising the expected LL:

$$\theta^{t+1} = \arg\max_{\theta} Q\left(\theta \mid \theta^{t}\right) \tag{4.4}$$

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

$$\log P\left(D \mid \theta^{t}\right) - \log P\left(D \mid \theta^{t-1}\right) < \epsilon\,, \tag{4.5}$$

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

Applying EM learning to a DAG requires that we compute:

$$\widetilde{N}_{ijk}^{t} = \sum_{m} P\left(V_i = k, \boldsymbol{Pa}_i = j \mid D_m, \theta^{t}\right) \tag{4.6}$$

for the E step, where $\widetilde{N}_{ijk}^{t}$ represents the expected count of number of records where the value of variable $V_i = k$ and its parents $\boldsymbol{Pa}_i = j$. For the M step, the solution of equation 4.4 has the following form:

$$\theta^{t+1} = \frac{\widetilde{N}_{ijk}^{t}}{\sum_k \widetilde{N}_{ijk}^{t}} \tag{4.7}$$

Once the parameters of the model are estimated, we use the expected LL obtained by the final M-step as the LL input in the BIC to measure the goodness-of-fit of a given reconstructed graph $\mathcal{G}_r$ with respect to the observed data. Specifically, the BIC score of a reconstructed graph $\mathcal{G}_r$ and data set D can be defined as:

$$S_{BIC}\left(\mathcal{G}_r, D\right) = Q\left(\theta^{t} \mid \theta^{t}\right) - \frac{1}{2}\log\left(N\right) d\,, \tag{4.8}$$

where N is the sample size of data set D and $d = \sum_{i} \left(r_i - 1\right) q_i$ estimates the number of free parameters in $\mathcal{G}_r$, considering both the hidden and observed variables, where r_i represents the number of states in variable V_i and q_i represents the number of configuration of the parents of V_i.

Note that the above equation for computing d represents the upper bound of the number of free parameters of graphs with hidden variables [Geiger et al., 1998, 2001]. Because there is no closed form solution to compute the number of free parameters of the reconstructed graphs that contain hidden variables, we assume the highest theoretical upper bound of the number of free parameters, and this implies high dimensionality penalty which in turn decreases the likelihood that the reconstructed graphs will outperform the learnt graphs in terms of the BIC score. Moreover, if the learnt graph or a reconstructed graph is a CPDAG, we will randomly select a DAG from the Markov equivalence class of the CPDAG and retrieve the BIC score of that DAG to represent the BIC score of the CPDAG, since the BIC score is equivalent for Markov equivalent DAGs.

When the endpoints of an edge in the learnt graph are present in multiple 3-vertex cliques simultaneously, it is possible that there is more than one connecting paths between them in the underlying noisy graph, such that we may need to import multiple hidden variables in the reconstructed graph to retrieve the missing conditional independence relationship. For example, in Figure 4.9a, V_1 and V_3 are conditionally independent given V_2 and V_4. However, this conditional independency does not hold in the corresponding observed data, and thus there is a spurious edge between V_1 and V_3 in the learnt graph shown in Figure 4.9b. In order to identify the true conditional independency between V_1 and V_3, i.e., $V_1 \perp\!\!\!\perp V_3 \mid \{V_2, V_4\}$, it requires that both the error-free nodes V_2 and V_4 are included in the reconstructed graph, as illustrated in Figure 4.9c.

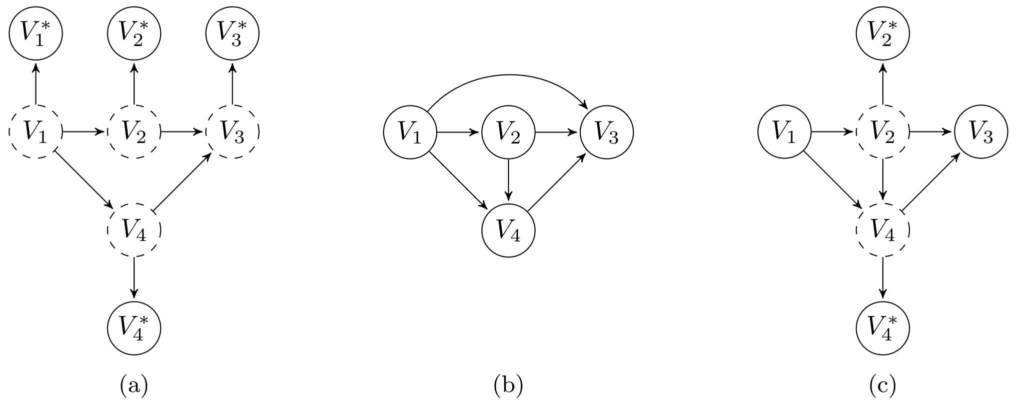


Figure 4.9: (a) A noisy graph with four variables that incorporate measurement error. (b) The learnt graph with respect to (a). (c) The reconstructed graph that contains the conditional independence $V_1 \perp\!\!\!\perp V_3 \mid \{V_2, V_4\}$.

The process we have used to reconstruct graphs is described in Algorithm 7, that takes as input a learnt graph $\mathcal{G}^l$, a set of noisy variables $\boldsymbol{V}$, a candidate spurious edge E, and a data set D. As described by Algorithm 7, a reconstructed graph is produced for each candidate spurious edge by replacing each involved noisy variable with an error-free variable, and adding the noisy variable as a child of its corresponding error-free variable. The hidden error-free variable has the same state space as the corresponding noisy variable. Moreover, the candidate spurious edge is removed from the reconstructed graph. Therefore, a reconstructed graph entails all the independences of the learnt graph, plus an additional independency corresponding to the endpoints of the candidate spurious edge. The output of Algorithm 7 represents the difference in BIC score between the reconstructed graph and the input learnt graph. If the input graph is a CPDAG, we compute the BIC score of that graph based on one of its valid DAGs, since all the DAGs that are part of the same Markov equivalence class would return the same BIC score. If the input graph is a PDAG which has no consistent DAG extensions, Algorithm 7 returns 0.

Algorithm 7 Graph reconstruction procedure

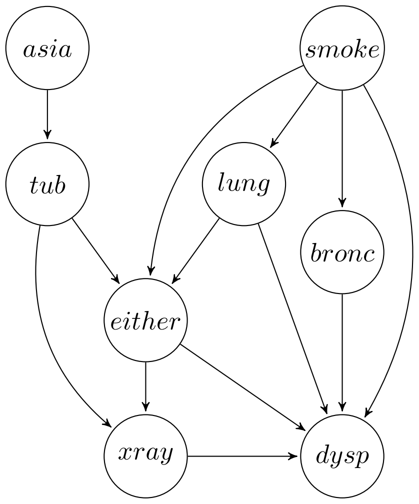
```
1: procedure RECONSTRUCTION($\mathcal{G}^l, E, \boldsymbol{V}, D$)
       Input: learnt graph $\mathcal{G}^l$, candidate spurious edge $E$, noisy variables $\boldsymbol{V}$, data $D$
       Output: difference in BIC score between reconstructed graph and input graph
2:     Compute the BIC score $score_o$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

Next, we introduce the complete SED algorithm, which represents an iterative process that searches for spurious edges by recursively executing the aforementioned Algorithm 7, and produces a modified graph that does not contain the edges identified as possible false positives. The pseudocode of the SED algorithm is described in Algorithm 8. Firstly, SED initialises the modified graph $\mathcal{G}_{mod}$ as a copy of the learnt graph $\mathcal{G}^l$, and generates the candidate spurious edge-nodes pairs CSE from $\mathcal{G}^l$. Then, SED recursively removes the candidate edges ordered by the highest positive output as determined by Algorithm 7 given the modified graph, and by assuming that there are l connecting paths between the endpoints of the candidate edge in the underlying noisy graph, where l is initially set to 1 and iteratively increased by 1 when no more edges can be identified as spurious give the current value of l. Therefore, SED is able to explore multiple connecting paths between the endpoints of every candidate spurious edge. The whole process is terminated when l is larger than the maximal size of $CSE\left(E_i\right)$. Note that when an edge between V_1 and V_2 in the learnt graph is detected as spurious by assuming that there is a connecting path between V_1 and V_2 via V_3 in the underlying noisy graph, it implies that V_1 and V_3 are either adjacent or connected through another connecting path that does not contain V_2. If V_1 and V_3 are adjacent in the noisy graph, it is not necessary to test whether the edge between them is spurious in the learnt graph. If there is a connecting path between V_1 and V_3 that does not contain V_2, then conditioning on the error-free variable V_2 will not d-separate V_1 and V_3 in the noisy graph. Therefore, SED will check for multiple connecting paths between V_1 and V_3.

Algorithm 8 Spurious Edge Detection (SED) algorithm

```
1: procedure SED($\mathcal{G}^l, D$)
       Input: learnt graph $\mathcal{G}^l$, data set $D$
       Output: modified graph $\mathcal{G}_{mod}$
2:     $\mathcal{G}_{mod} = \mathcal{G}^l$
3:     initialise $CSE$ from $\mathcal{G}^l$
4:     $l = 1$
5:     repeat
6:         $CSE_l = \{\}$
7:         for $E_i \in CSE$ do
8:             $CSE_l\left(E_i\right)$ = all subsets of $CSE\left(E_i\right)$ with length $l$
9:         end for
10:        while $\max_{E_i \in CSE, \boldsymbol{V}_j \in CSE_l(E_i)} Reconstruction\left(\mathcal{G}^l, E_i, \boldsymbol{V}_j, D\right) > 0$ do
11:            $E_m, \boldsymbol{V_m} = \arg\max_{E_i \in CSE, \boldsymbol{V}_j \in CSE_l(E_i)} Reconstruction\left(\mathcal{G}^l, E_i, \boldsymbol{V}_j, D\right)$
12:            remove $E_m$ from $\mathcal{G}_{mod}$
13:            remove $E_m$ from $CSE$
14:            if $l == 1$ then
15:                prune the edges between each endpoint of $E_m$ and $V_m$ from $CSE_l$
16:            end if
17:        end while
18:        $l = l + 1$
19:    until $l > \max_{E_i \in CSE} \left|CSE\left(E_i\right)\right|$
20:    return $\mathcal{G}_{mod}$
21: end procedure
```

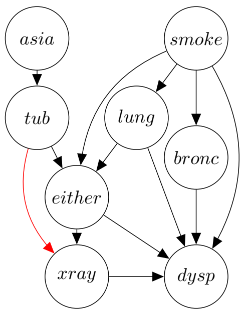
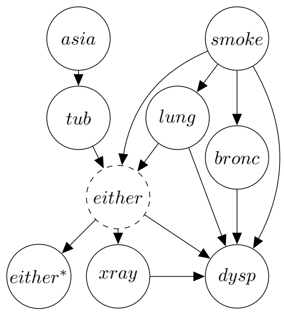
Table 4.1 presents a worked example that illustrates the different steps of SED when applied to the graph shown in Figure 4.10. This experiment is based on the Asia network learnt by the HC algorithm from a synthetic data set with 10,000 samples and 5% measurement error on each observed variable. In Table 4.1, the modified graph represents the state of modified graph at the given iteration, the optimal reconstructed graph represents the reconstructed graph with the highest positive output, and CSE represents the candidate spurious edge set that contains edges that continue to be tested for false positives. The red edges depicted in Table 4.1 represent the



$$CSE = \left\{ \begin{aligned} & tub \rightarrow either : \{xray\}, \\ & tub \rightarrow xray : \{either\}, \\ & smoke \rightarrow lung : \{either, dysp\}, \\ & smoke \rightarrow either : \{lung, dysp\}, \\ & smoke \rightarrow bronc : \{dysp\}, \\ & smoke \rightarrow dysp : \{lung, either, bronc\}, \\ & lung \rightarrow either : \{smoke, dysp\}, \\ & lung \rightarrow dysp : \{smoke, either\}, \\ & bronc \rightarrow dysp : \{smoke\}, \\ & either \rightarrow xray : \{tub, dysp\}, \\ & either \rightarrow dysp : \{smoke, lung, xray\}, \\ & xray \rightarrow dysp : \{either\} \end{aligned} \right\}$$

Figure 4.10: Left: The Asia graph learnt by the HC algorithm from a synthetic data set with 10,000 samples and 5% measurement error on each observed variable. Right: the candidate spurious edge-nodes pairs CSE of the learnt graph.

edges classified as spurious by SED, whereas the blue edges represent the edges pruned (i.e., no longer being considered as candidate spurious edges) after each iteration. A candidate spurious edge is pruned when no valid reconstructed graph is found to have a score that is higher than the score of the learnt graph.

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

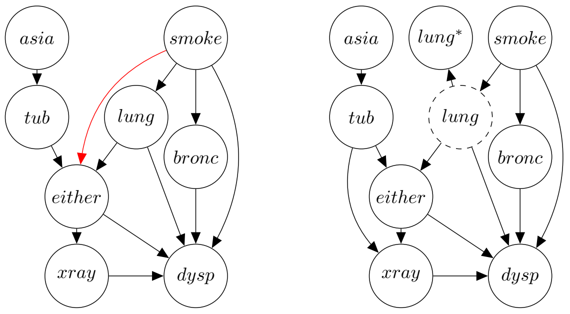
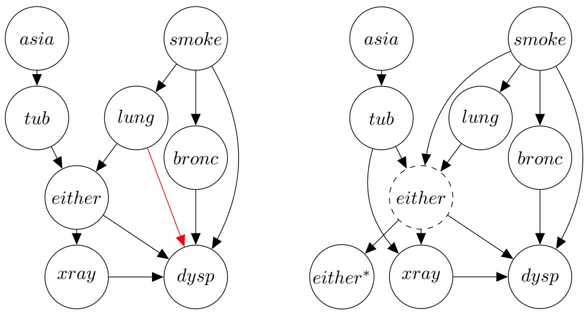
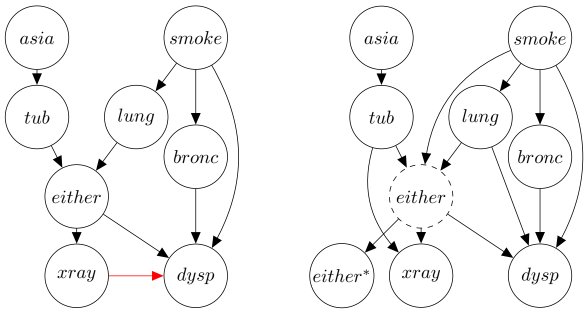
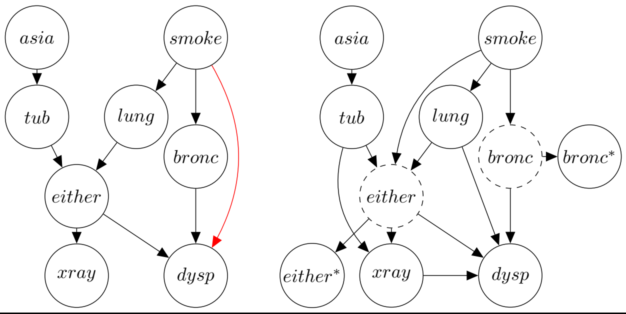
2		$smoke \rightarrow lung : \{either, dysp\}$ $smoke \rightarrow either : \{lung, dysp\}$ $smoke \rightarrow bronc : \{dysp\}$ $smoke \rightarrow dysp : \{lung, either, bronc\}$ $lung \rightarrow either : \{smoke, dysp\}$ $lung \rightarrow dysp : \{smoke, either\}$ $bronc \rightarrow dysp : \{smoke\}$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

Table 4.1: The steps of the SED algorithm in modifying the Asia graph learnt by HC from synthetic data of sample size 10,000 with 5% measurement error on all variables. The red edges represent the edges classified as spurious in each iteration, whereas the blue edges represent the edges being pruned (i.e., no longer being considered as candidate spurious edges) after each iteration.

The illustration in Table 4.1 starts by initialising the modified graph as a copy of the learnt graph shown in Figure 4.10, and determining the CSE based on that graph. Then, SED iterates over the candidate spurious edges by assuming that there is one connecting path between the endpoints of the candidate edges, i.e., assuming one variable which is adjacent to the endpoints of the candidate spurious edge as noisy, and importing its error-free parent in the reconstructed graph. During the iterative process, SED first identifies $tub \rightarrow xray$ as spurious since the reconstructed graph that does not contain $tub \rightarrow xray$ returns the highest score. SED then removes $tub \rightarrow xray$ from the modified graph and further prunes $tub \rightarrow either$ from CSE (i.e., it is no longer considered as a candidate spurious edge). This is because SED finds that there is a connecting path between *tub* and *xray* via *either*, which in turn implies that *tub* and *either* are either adjacent or connected through a connecting path that does not contain *xray* in the true noisy graph. Therefore, it is not necessary to examine whether $tub \rightarrow either$ is spurious under the assumption of a single connecting path between *tub* and *either* via *xray*.

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

4.6 Empirical evaluation

We validate the effectiveness of the SED algorithm, which can be viewed as a structure learning addon, by applying it to five well-established structure learning algorithms spanning different classes of learning. These are the score-based HC, GES and GOBNILP, the constraint-based PC-stable and the hybrid H2PC. We use the bnlearn R package [Scutari et al., 2010] to test the effect on HC and H2PC, the rcausal R package [Wongchokprasitti, 2019] for PC-stable and GES, and the pygobnilp python package [Cussens, 2011] for GOBNILP. The evaluation does not consider the algorithms mentioned in Section 4.2 because those algorithms assume continuous data, whereas here we focus on categorical data.

We use the BIC score as the objective function for the three score-based HC, GES and GOBNILP algorithms, and for the score-based phase of H2PC. For the constraint-based algorithm PC-stable, including the constraint-based phase in H2PC, we use the G-square test as the statistical test and set the threshold for rejecting the null hypothesis at 0.05. Lastly, GOBNILP's maximum in-degree is set to 3 (default hyperparameter). Because BIC is a score-equivalent objective function, HC, GES, GOBNILP and H2PC produce a DAG from a Markov Equivalent Class of DAGs, and which we convert into the corresponding CPDAGs to be used as the input of the SED algorithm; i.e., input graph G in Algorithm 8. Recall that when the constraint-based PC-stable returns a PDAG that cannot be converted into a DAG, SED makes no modifications since the BIC score cannot score that PDAG and hence, Algorithm 7 returns 0 in this case. We employ two metrics to evaluate the learnt CPDAGs which are the F_1 score and the re-scaled SHD score.

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

$$P\left(V_i^* \mid V_i = v_i^l\right) = \begin{cases} \alpha_{i1}^l, & V_i^* = v_i^1 \\ \alpha_{i2}^l, & V_i^* = v_i^2 \\ \vdots & \vdots \\ 1-\alpha_i^l, & V_i^* = v_i^l \\ \vdots & \vdots \\ \alpha_{ir_i}^l, & V_i^* = v_i^{r_i} \end{cases} \tag{4.9}$$

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

We explore the performance of the SED algorithm on both error-free and noisy data sets. Figure 4.11 presents the F_1, precision and recall scores produced by the five algorithms averaged across all seven networks, on both the error-free and noisy data sets, with and without SED modifications. Note that in the case of error-free data sets, there is no visible difference in the precision, recall and F_1 scores between the learnt graphs and the graphs modified by SED. From this, we can conclude that SED performs largely insignificant modifications to the graphs learnt from error-free data sets. On the other hand, the modifications made on graphs learnt from noisy data have led to noticeable improvements in terms of the precision and F_1 metrics, and particularly in cases where data have higher sample size. This can be explained by the fact that the structure learning algorithms generally tend to produce more edges when the input data contain higher samples, such that more false positive 3-vertex cliques that could be detected and corrected by SED. We present the results of 3-vertex cliques in Figure 4.12. Another explanation is that the EM learning used by SED is less effective when the sample size of the input data is low.

Specifically, the results show that SED improves the average precision and F_1 scores across all the five algorithms, when the data contain measurement error (both 10% or 20%) and when the sample size is larger than 1,000. This suggests that the edges removed by SED are mainly false

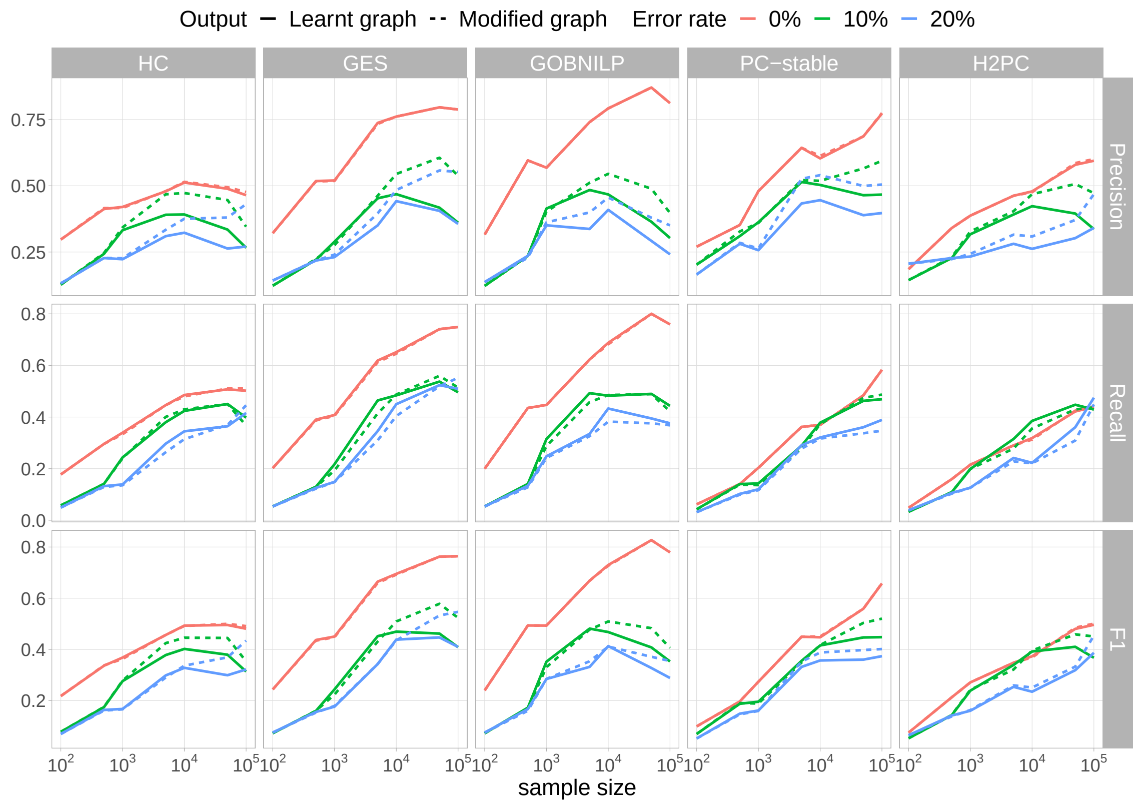


Figure 4.11: The average precision, recall and F_1 scores of the graphs produced by the five algorithms, where solid lines represent the results before SED modifications, dashed lines represent the results after SED modifications, red lines the results based on error-free data, green lines the results based on noisy data with 10% error rate, and blue lines the results based on noisy data with 20% error rate.

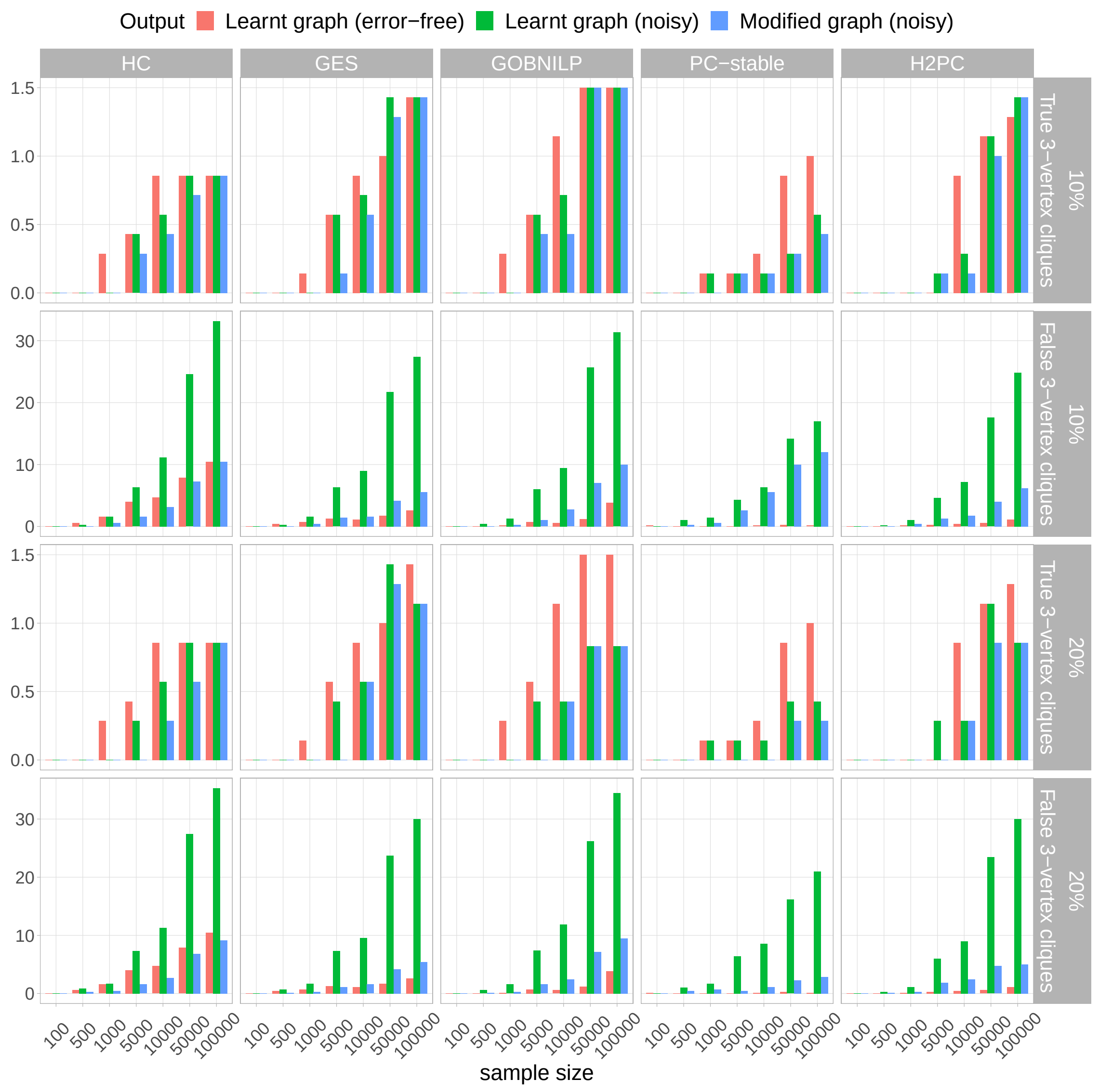


Figure 4.12: The average number of false and true 3-vertex cliques produced by the learnt graphs learnt from error-free data sets, the learnt graphs learnt from noisy data sets and the modified graphs learnt from noisy data sets.

positives, even though SED would occasionally also remove some true positive edges which causes recall to drop slightly for some of the experiments. When the sample size is less than 1,000, the learnt graphs tend to contain a small number of 3-vertex cliques (refer to Figure 4.12), and this gives little to no opportunity to SED to make modifications. The results also show that there is no apparent difference in the gain in score over the two different levels of measurement error tested. Interestingly, the improvements on graphs produced by score-based HC, GES and GOBNILP are somewhat stronger compared to the improvements on graphs produced by PC-stable and H2PC, based on the hyperparameter defaults of these algorithms. These results are consistent with the empirical analysis presented in Figure 4.6, which shows that score-based learning is more sensitive to measurement error compared to constraint-based learning.

Figure 4.13 repeats these results for the re-scaled SHD score. While the results are largely consistent with those based on the F_1 score, the improvements appear to be major and more consistent in terms of the SHD score. Specifically, Figure 4.13 shows that after applying SED to the graphs learnt from noisy data by all five structure learning algorithms, the number of removed and re-orientated edges required to convert the learnt graphs to true graphs is significantly reduced, and this reinforces the observation that SED is effective at removing false positive edges. Figure 4.13 also shows that the number of edges produced by the algorithms increases slightly with measurement error as expected. This implies that SED is likely to have more opportunities to remove edges as the rate of measurement error increases. Overall, the SHD results suggest that the SED algorithm improves the graphs learnt by the structure learning algorithms by successfully eliminating many more false positive, rather than true positive, edges.

Table 4.2 presents the number and percentage of modified graphs that are better, equal or worse than the learnt graphs, under different experimental settings and evaluation metrics. The results show that when no measurement error exists in the input data (i.e., error-free case), the SED algorithm preserves the learnt graph in around 90% of the cases, and improves or decreases the performance approximately in equal proportions for the remaining 9% of cases. Specifically, without measurement error in the data, the modifications increased the F_1 score in 11 (5%) graphs and decreased it in 10 (4%) graphs. Similarly, the modifications increased the SHD score in 10 (5%) graphs and decreased it in 9 (4%) graphs.

On the other hand, when measurement error exists in the input data (i.e., noisy cases), the SED modifications improve considerably more graphs than the graphs worsened. According to the SHD score, the SED modifications improve 120 (49%) and 130 (53%) learnt graphs and only worsen just 9 (4%) and 10 (4%) learnt graphs when the error rate is 10% and 20% respectively. In terms of the F_1 score, the SED modifications improve 93 (38%) and 109 (44%) learnt graphs and worsen 51 (21%) and 48 (20%) learnt graphs when the error rate is 10% and 20% respectively. These percentages are generally consistent across all five algorithms irrespective of their class of learning.

Lastly, Figure 4.14 presents the average execution time needed to produce the learnt and the modified graphs, across the different algorithms and over different error rates and sample sizes. The execution time for the modified graphs refers to the time it takes SED to modify the learnt graphs. The five structure learning algorithms used to produce the learnt graphs spent similar time learning the graphs from error-free data and noisy data with 10% error rate, and were slightly faster (in general) when learning from noisy data with 20% error rate. When it comes to SED, because the number of 3-vertex cliques tends to increase with both the measurement and the sample size of the input data (see Figure 4.12), it naturally spends more time to process learnt graphs produced

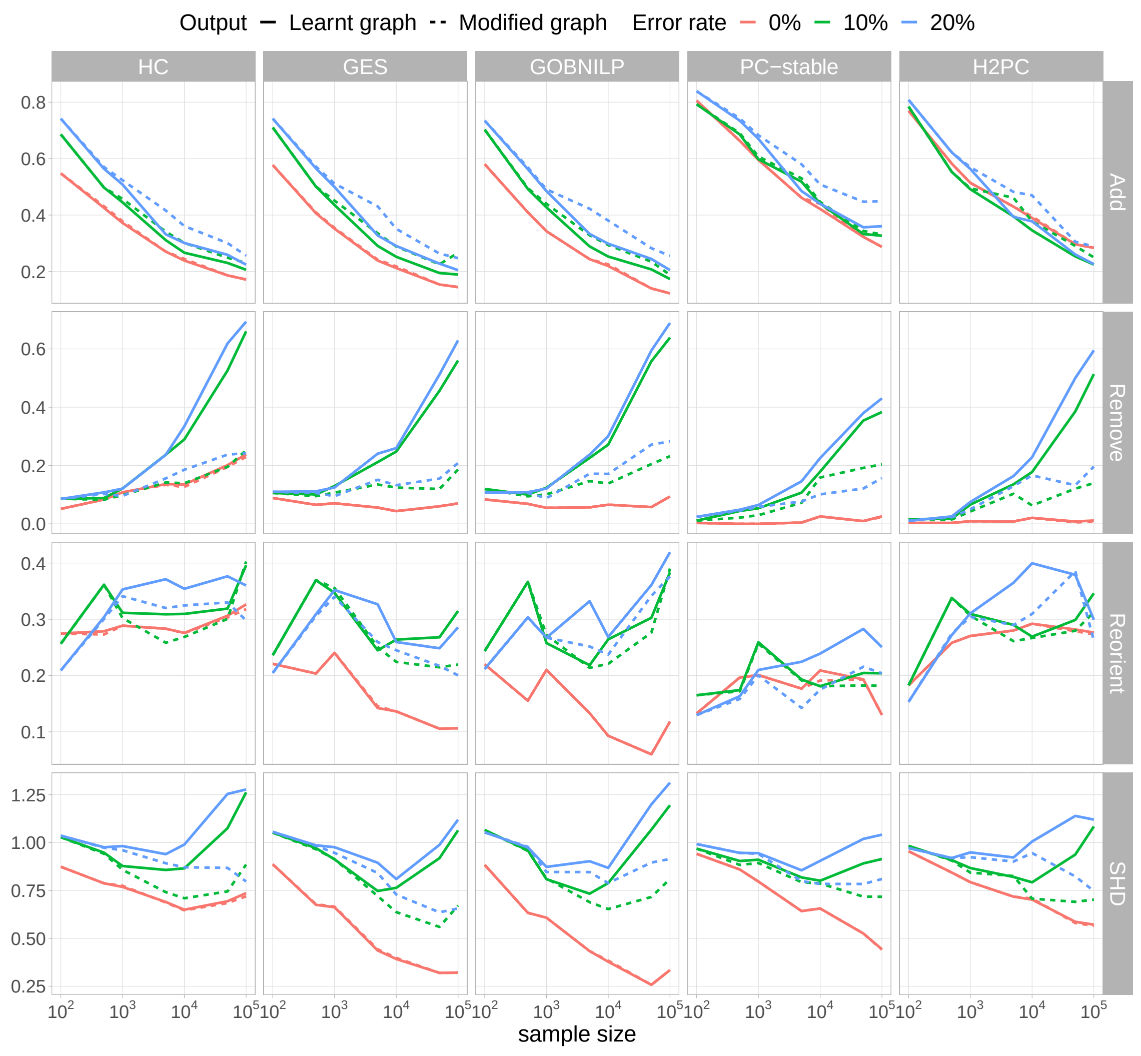


Figure 4.13: The average re-scaled SHD scores and its three components produced by the four algorithms, where solid lines represent the results before SED modifies the graphs, dashed lines represent the results after SED modifies the graphs, green lines the results based on noisy data with 10% error rate, and blue lines the results based on noisy data with 20% error rate.

Algorithm	Modified graph vs learnt graph	Error rate					
		0%		10%		20%	
		F_1	SHD	F_1	SHD	F_1	SHD
HC	Better	7/14%	7/14%	25/51%	30/61%	25/51%	28/57%
	Equal	38/78%	39/80%	16/33%	18/37%	18/37%	19/39%
	Worse	4/8%	3/6%	8/16%	1/2%	6/12%	2/4%
GES	Better	0/0%	0/0%	20/41%	26/53%	21/43%	27/55%
	Equal	45/92%	45/92%	17/35%	21/43%	18/37%	20/41%
	Worse	4/8%	4/8%	12/24%	2/4%	10/20%	2/4%
GOBNILP	Better	0/0%	0/0%	14/28%	23/47%	18/37%	25/51%
	Equal	48/98%	48/98%	19/39%	25/51%	19/39%	22/45%
	Worse	1/2%	1/2%	16/33%	1/2%	12/24%	2/4%
PC-stable	Better	2/4%	1/2%	13/27%	16/33%	25/51%	26/53%
	Equal	47/96%	48/98%	30/61%	29/59%	15/31%	21/43%
	Worse	0/0%	0/0%	6/12%	4/8%	9/18%	2/4%
H2PC	Better	2/4%	2/4%	21/43%	25/51%	20/41%	24/49%
	Equal	46/94%	46/94%	19/39%	23/47%	18/37%	23/47%
	Worse	1/2%	1/2%	9/18%	1/2%	11/22%	2/4%
Overall	Better	11/5%	10/4%	93/38%	120/49%	109/44%	130/53%
	Equal	224/91%	226/92%	101/41%	116/47%	88/36%	105/43%
	Worse	10/4%	9/4%	51/21%	9/4%	48/20%	10/4%

Table 4.2: The number and percentage of modified graphs that are better, equal or worse than the learnt graphs in terms of graphical accuracy, for each algorithm, error-rate, and over different evaluation metrics.

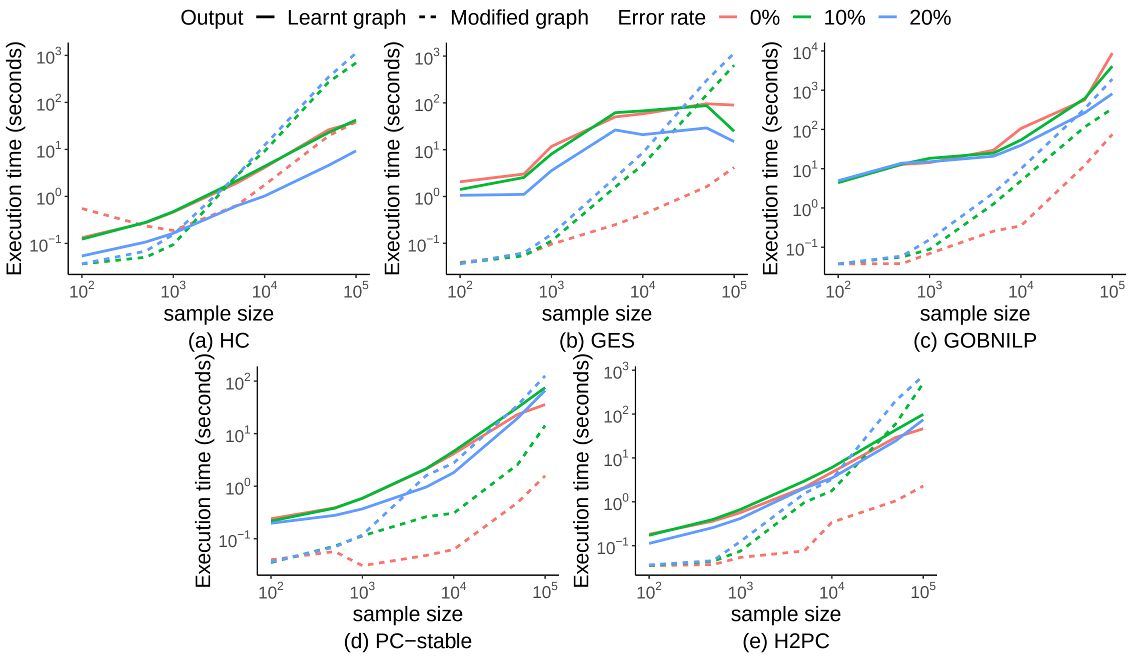


Figure 4.14: Average execution time needed to produce the learnt and modified graphs for the specified algorithms, across different error rate and sample size combinations. The execution time of SED is based on the time it takes to modify graphs.

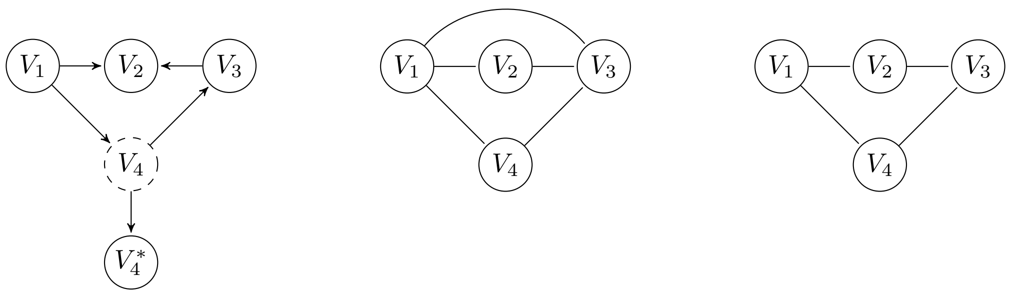
with higher error rate and/or sample size. When the sample size is less than or equal to 10,000, the execution time spent by SED is generally less than the time spent by the structure learning algorithms, whereas the execution time is similar to that of the structure learning algorithms when the sample size is larger than 10,000.

4.7 Conclusion

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

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

A limitation of our work is that SED is restricted to discrete data. While extending SED to continuous data might be a sensible direction for future work, it should be noted that working with continuous data is likely to introduce further challenges, and this is because the computational complexity of EM learning tends to increase substantially when applied to continuous data.



(a) True DAG with measurement error on V_4 (b) Learnt CPDAG (c) Recovered CPDAG by SED

Figure 4.15: An example where SED fails to recover the true CPDAG.

Another limitation is that the proposed algorithm relies on the assumption that a noisy variable is independent of other variables in the network conditional on its error-free version, and this assumption is often considered to be too strong in some fields [Hu, 2008]. For example, a survey on unemployment data by Bound et al. [2001] shows that unemployment rate is underestimated, and the underestimation error appears to be dependent on the demographic characteristics of the respondent, such as age and sex.

The spurious edges will often cause true unshielded colliders to become shielded colliders in the learnt CPDAG. Therefore, in order to recover the underlying true CPDAG, we have to not only eliminate the spurious edges, but also to reorientate some of the (un)directed edges to be able to recover the v-structures distorted by spurious edges caused by measurement error. Figure 4.15 presents an example where SED fails to recover the true v-structure $V_1 \rightarrow V_2 \leftarrow V_3$. This happens because, after eliminating the spurious edge $V_1 - V_3$, SED preserves the remaining edges and converts the graph into a CPDAG without checking the possibility of v-structures in the unshielded triple $\langle V_1, V_2, V_3 \rangle$. Future research works could include this additional learning phase in SED to ensure the true CPDAG is recoverable under this scenario.

Moreover, since the problem of measurement error can be viewed as a special case of a hidden variable problem, future work could extend the application of this approach to structure learning algorithms designed to learn graphical structures under the assumptions of causal insufficiency [Zhang, 2008, Ogarrio et al., 2016].

Chapter 5

Improving greedy search structure learning in the presence of systematic missing values

5.1 Introduction

Learning from data that contain missing values represents a common phenomenon in many domains. While numerous BN structure learning algorithms have been proposed in the literature over the past few decades, relatively few of them account for missing data, and those that do tend to rely on standard approaches that assume missing data are missing at random, such as the Expectation-Maximisation algorithm. This hinders the application of structure learning to real-world problems, since missing data represents a common issue in most applied areas including medicine and healthcare [Constantinou et al., 2016], clinical epidemiology [Pedersen et al., 2017], traffic flow prediction [Tian et al., 2018], anomaly detection [Zemicheal and Dietterich, 2019], and financial analysis [John et al., 2019]. Therefore, there is a greater need for structure learning algorithms that account for potential data bias due to systematic missing values, without having significant impact on the computational efficiency of structure learning.

According to Rubin [1976], missing data problems can be categorised into three classes. These are the Missing Completely At Random (MCAR), the Missing At Random (MAR) and the Missing Not At Random (MNAR). Specifically, MCAR denotes that the observed variables are marginally independent of the indicator of observability. Here, the indicator of observability is an auxiliary variable to represent the missingness of partially observed variables. This type of missingness is usually caused by technical error that would not bias the analysis. The definition of MAR, on the other hand, is somewhat counterintuitive in its name and assumes that missing data are independent of indicators of observability given observed data. For example, in an investigation between age and frequency of smoking, missing data are MAR if younger respondents are more likely to not disclose their smoking frequency. Lastly, data missingness are said to be MNAR if it is not MAR. In the above example, the missingness are MNAR if data on respondent's age also contains missing values.

Methods that deal with missing data typically include naïve approaches such as the complete case analysis (a.k.a list-wise deletion) and multiple imputation [Rubin, 2004]. Complete case analysis involves removing the data cases that contain missing values and hence, restricting learning to complete data cases. Clearly, while this approach is easy to implement, it can be sample inefficient and may yield bias when missingness are not MCAR [Graham, 2009]. Multiple imputation, on the other hand, fills - rather than ignoring - the missing values and takes the uncertainty of imputation into consideration by repeating imputation over different possible values [Azur et al., 2011]. However, multiple imputation is built under the assumption of MAR which means it may also produce biased outcomes when data are MNAR.

In this Chapter, we propose three variants of the greedy search Hill-Climbing algorithm to investigate how they handle missing data values under different assumptions of missingness. These variants can be viewed as fusions between greedy search score-based learning, and the pairwise deletion and IPW methods discussed above that have been previously applied to constraint-based learning. The contribution of this Chapter is a novel structure learning algorithm suitable for structural learning from data that contain systematic missingness, and is organised as follows: Section 5.2 reviews relevant works about structure learning under the presence of missing values, Section 5.3 provides necessary preliminary information that includes notation and background information, Section 5.4 describes the proposed algorithm, Section 5.5 presents the results, and we provide our concluding remarks in Section 5.6.

5.2 Relevant works

One of the earliest advanced approaches for dealing with missing data is the EM algorithm, which was also later adopted by the structure learning community. The Structural EM algorithm [Friedman et al., 1997] is an iterative process which consists of two steps: the Expectation (E) step and the Maximisation (M) step. In E step, Structural EM makes inferences on the missing values and computes the expected sufficient statistics based on the graph learnt in previous iteration. The M step follows where the current state of the learnt graph is revised based on the sufficient statistics obtained at step E. An advantage of Structural EM is that it can be combined with different structure learning algorithms. A disadvantage, however, is that it is computationally inefficient due to the inference process that takes place at step E. Therefore, in practice, the E step of the Structural EM algorithm is usually implemented with single imputation, i.e., imputing the expectation of the missing values derived from the observed values. Ruggieri et al. [2020] compared the performance of the original Structural EM to that of the imputed-based Structural EM, and found that the latter achieves better performance in most of the simulation scenarios.

An increasing number of algorithms are recently proposed to improve structure learning from data containing missing values. In the case of score-based learning, two model selection methods have been proposed based on the likelihood function called Node-Average Likelihood (NAL) for discrete [Balov et al., 2013] and conditional Gaussian BNs [Bodewes and Scutari, 2021]. NAL is a decomposable function which computes local likelihood for each variable based on the locally complete data set. The formal definition of NAL can be written as:

$$S_{NAL}\left(\mathcal{G}, D\right) = \sum_{i} \frac{1}{|D_i|} S_{LL}\left(V_i \mid \boldsymbol{Pa}_i; \hat{\Theta}_i\left(D_i\right)\right), \tag{5.1}$$

where D_i is the locally complete data set for V_i in which V_i and its parents $\boldsymbol{Pa}_i$ are all observed, and $\hat{\theta}_i\left(D_i\right)$ is the maximum likelihood estimates of the parameters obtained from D_i. Note that although the NAL score is consistent under MCAR, it is not consistent under MAR or MNAR.

In the case of constraint-based learning, Strobl et al. [2018] treated missing values as a type of selection bias and showed that performing test-wise deletion during CI tests represents a sound solution for the FCI algorithm [Spirtes et al., 2000]. In the context of constraint-based learning, test-wise deletion is a process that deletes the data cases with missing values amongst the variables involved in a given CI test. Gain and Shpitser [2018] later show that replacing the standard CI test in PC with an Inverse Probability Weighting (IPW) [Horvitz and Thompson, 1952] based CI test, enables PC to be applied to data sets which contain systematic missing values without loss of consistency. IPW is an approach to alleviate bias in data distributions by reweighting the data cases which we will describe in detail in Section 5.4. However, IPW based CI testing assumes sufficient information of missingness, such as information about the parents of missingness and the total ordering of the missing indicators, which is unlikely to be known in practise. Tu et al. [2019] tried to address this issue by making additional assumptions about the type of missingness, to find the parents of missingness from the restricted search space. They first predict the parents of missingness using a constraint-based learning process, for every observed variable that contained missing values, and then apply the IPW based CI tests using the sufficient information obtained during the constraint-based learning phase. In this chapter, we adopt the same assumptions (i.e., Assumptions 5.3.1 and 5.3.2) as those used in [Tu et al., 2019], but introduce a new solution for greedy search score-based algorithms to handle systematic missingness.

5.3 Preliminaries

In this Chapter, we consider discrete variables which we denote with uppercase letters (e.g., U, V), and the assignment of variable states with lowercase letters (e.g., u, v). We denote a set of variables with bold uppercase letters (e.g., $\boldsymbol{U}, \boldsymbol{V}$), and the assignment of a set of variable states with bold lowercase letters (e.g., $\boldsymbol{u}, \boldsymbol{v}$). We also adopt the Markov assumption, the faithfulness assumption and the causal sufficiency assumption previously introduced in Section 2.1.

5.3.1 Hill Climbing algorithm

For simplicity, we focus on the Hill-Climbing (HC) structure learning algorithm [Heckerman et al., 1995] described in Subsection 2.2.2. As with most other structure learning algorithms, HC is usually paired with a decomposable score function to evaluate each graph explored relative to the input data. A score function $S(\mathcal{G}, D)$ is decomposable if it can be written as the sum over a set of local scores, each of which corresponds to a variable and its parents in $\mathcal{G}$. While all score-based algorithms can use a decomposable score, this property is particular efficient in the case of HC search since it explores one or two graphical modifications at a time; i.e., one in case of edge addition or removal, and two in the case of edge reversal. Therefore, the objective score for each neighbouring graph $\mathcal{G}_{nei}$ can be obtained efficiently by only recomputing the local scores of up to two nodes whose parent-set has changed, and obtaining the local scores of the remaining nodes whose parent-set remains intact from the current best graph $\mathcal{G}$.

5.3.2 Missing data

Given the definition from Rubin [1976], missing data can be categorised into three classes based on their randomness.

1. Missing Completely At Random (MCAR): MCAR refers to the situation where the missingness of data is unrelated to both observed and unobserved variables. In other words, the missingness occurs randomly and has no systematic relationship with the data.

2. Missing At Random (MAR): MAR occurs when the missingness can be explained by the observed variables in the data set but not by the unobserved variables. The missingness is related to the observed data, but not to the missing values themselves. Theoretically, when data is MAR, the missing values can be estimated without bias using the observed variables, ensuring that no information is lost in the data.

3. Missing Not At Random (MNAR): MNAR refers to a situation where the missingness in a data set depends on the unobserved variables or the missing values themselves. In other words, the probability of missingness is related to the values that are missing and cannot be explained by the observed variables alone. MNAR is the most intricate yet realistic assumption regarding missing data, as it implies that the missing values cannot be accurately identified from the existing data. Mishandling such missing values can lead to erroneous outcomes and misleading results.

Recently, Mohan et al. [2013] proposed a graphical model called *missingness graph* (or m-graph) to capture both the data and missing value generation mechanisms simultaneously. Daniel et al. [2012], Martel García [2013], Thoemmes and Rose [2014] also propose graphical models for missing data. In this thesis, we adopt the m-graph model since the marginal distribution represented by a m-graph is identifiable under certain conditions. The m-graph contains both fully observed variables and partially observed variables, and use the connections between fully observed variables and partially observed variables to imply different missingness scenarios.

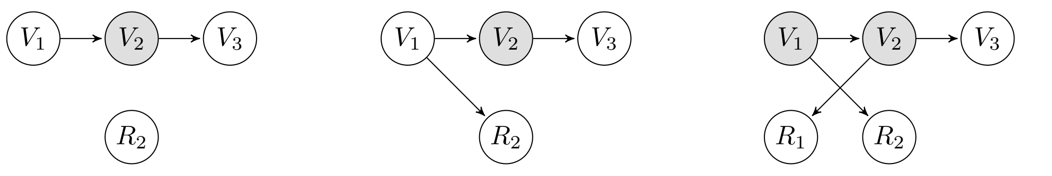
We denote the set of fully observed variables (i.e, variables without missing values) as $\boldsymbol{V}^o$ and the set of partially observed variables (i.e., variables with at least one missing values) as $\boldsymbol{V}^m$. For every partially observed variable $V_i \in \boldsymbol{V}^m$, we define an auxiliary variable R_i called missing indicator to reflect the missingness in V_i, where R_i takes the value of 0 when V_i is recorded and the value of 1 when V_i is missing. Then an m-graph $\mathcal{G}(\mathbb{V}, \boldsymbol{E})$ is composed by variables $\mathbb{V} = \boldsymbol{V}^o \cup \boldsymbol{V}^m \cup \boldsymbol{R}$ and edges $\boldsymbol{E}$ between $\mathbb{V}$. A m-graph assumes that the observed variable V cannot be the child of any missing indicator R. Given the m-graph, we can redefine the three missing assumptions as follows:

1. MCAR: variables in $\boldsymbol{R}$ are not influenced by any observed variables in the m-graph, such that $\boldsymbol{R} \perp\!\!\!\perp \boldsymbol{V}^o \cup \boldsymbol{V}^m$.

2. MAR: variables in $\boldsymbol{R}$ could only be influenced by the fully observed variables but not by the partially observed variables in the m-graph, such that $\boldsymbol{R} \perp\!\!\!\perp \boldsymbol{V}^m \mid \boldsymbol{V}^o$.

3. MNAR: variables in $\boldsymbol{R}$ could be influenced by both fully observed variables and partially observed variables in the m-graph.

Figure 5.1 presents the three possible m-graphs assuming three observed variables with structure $V_1 \to V_2 \to V_3$, depicting the MCAR, MAR and MNAR assumptions respectively. Note that the missingness assumptions defined by the m-graph approach are stronger than the assumptions defined by Rubin [1976]. This is because the m-graph approach considers missingness based on variables rather than individual values. However, the m-graph approach offers a more explainable and transparent method for encoding missing patterns, which is why it is employed in this thesis.



(a) MCAR m-graph (b) MAR m-graph (c) MNAR m-graph

Figure 5.1: The three possible m-graphs assuming three observed variables with structure $V_1 \rightarrow V_2 \rightarrow V_3$. Shaded nodes represent partially observed variables.

Mohan et al. [2013] show that, given the causal sufficiency assumption and Assumption 5.3.1, the population distributions are identifiable from the observed data if and only if there is no variable that is also a parent of its own missingness indicator. We, therefore, adopt the following two assumptions needed to ensure the identifiability of the underlying data population.[1]

Assumption 5.3.1. *There are no edges between variables in* $\boldsymbol{R}$.

Assumption 5.3.2. *No partially observed variable can be the parent of its own missingness indicator.*

Although Mohan et al. [2013] show that systematic missingness can be handled effectively by IPW in theory, it is not convenient to apply IPW to the entire observed data in practice since this will significantly reduce the sample size, which in turn will negatively affect the results (this is further discussed in Subsection 5.4.3). Studies that apply IPW to constraint-based learning do so by restricting the application of IPW to the parts of data involved in the CI test. However, this concept cannot be extended to score-based learning, where no related solution exists. The subsequent subsections introduce three variants of the HC algorithm, and describe how to incorporate IPW to score-based greedy search algorithms, while aiming to maximally leverage observed data.

5.4 Handling systematic missing data with Hill-Climbing

This section describes the three HC variants that we explore in extending the learning process towards dealing with systematic missing data. Specifically, subsection 5.4.1 describes HC with pairwise deletion which we call HC-pairwise, subsection 5.4.2 describes HC with both pairwise deletion and Inverse Probability Weighting which we call HC-IPW, and subsection 5.4.3 describes an improved version of HC-IPW, the HC-aIPW, that prunes off less data samples compared to HC-IPW. The first two HC-variants can be viewed as sub-versions of HC-aIPW, but are important in their own in illustrating the successive improvements in learning accuracy.

5.4.1 Hill-Climbing with pairwise deletion

Recall that, at each iteration, HC moves to the neighbouring graph that maximally improves the objective score, and that performing HC search with a decomposable scoring function means that there is no need to recompute the local score of variables whose parent-set remains unchanged across graphs. Therefore, an efficient (but not necessarily effective) way of applying HC to missing data is to ignore data cases that contain missing values in variables that form part of the set of variables considered when exploring local score changes to a DAG. We refer to this process as

[1] We do not check whether these assumptions hold because they are not testable in the presence of missing values.

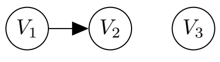
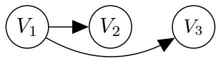
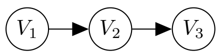
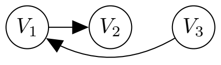
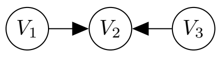
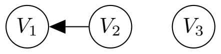
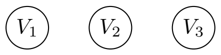
Current DAG state $\mathcal{G}$	Edge operation	Neighbouring DAG $\mathcal{G}_{nei}$	Necessary variables
	add $V_1 \rightarrow V_3$		$\{V_1, V_3\}$
	add $V_2 \rightarrow V_3$		$\{V_2, V_3\}$
	add $V_3 \rightarrow V_1$		$\{V_1, V_3\}$
	add $V_3 \rightarrow V_2$		$\{V_1, V_2, V_3\}$
	reverse $V_1 \rightarrow V_2$		$\{V_1, V_2\}$
	delete $V_1 \rightarrow V_2$		$\{V_1, V_2\}$

Table 5.1: Examples of necessary variables for each edge operation in HC, which we define as the variables with different parent-sets between the current best and neighbouring graphs, plus the parents that make up those parent-sets.

pairwise deletion, where "pair" refers to the current pair of candidate DAGs (the current best DAG and neighbouring DAG), and this deletion process may involve more than two variables. When comparing the current best DAG against a neighbouring DAG, the *necessary variables* would be the nodes with unequal parent-sets between the two graphs, plus the parents of those nodes in the two graphs. Formally, when exploring a neighbouring DAG $\mathcal{G}_{nei}$ from the current best DAG $\mathcal{G}$, the set of necessary variables $\boldsymbol{W}$ between $\mathcal{G}$ and $\mathcal{G}_{nei}$ can be described as:

$$\boldsymbol{W} = \cup_{V_i \in \boldsymbol{V}_d} \left\{V_i, \boldsymbol{Pa}_i, \boldsymbol{Pa}_i^{nei}\right\}, \tag{5.2}$$

where $\boldsymbol{V}_d$ is the set of variables that have different parent-sets between $\mathcal{G}$ and $\mathcal{G}_{nei}$, and $\boldsymbol{Pa}_i$ and $\boldsymbol{Pa}_i^{nei}$ are the parent-sets of V_i in $\mathcal{G}$ and $\mathcal{G}_{nei}$ respectively. For simplicity, we refer to the data set obtained after applying pairwise deletion as the pairwise deleted data set.

Example 5.4.1. Assume that, during HC, the current state of DAG $\mathcal{G}$ is a graph containing three variables $\{V_1, V_2, V_3\}$ and the edge $V_1 \rightarrow V_2$, as illustrated in Table 5.1. Given DAG $\mathcal{G}$, there are six possible edge operations each of which produces a neighbouring graph $\mathcal{G}_{nei}$. Operation add $V_1 \rightarrow V_3$, for example, can be evaluated by assessing the change in the local score of V_3, i.e., $S(V_3 \mid V_1) - S(V_3)$, since V_3 is the only variable with different parents between $\mathcal{G}$ and $\mathcal{G}_{nei}$. When the data set contains missing values, we can apply pairwise deletion to data given $\{V_1, V_3\}$ in order to obtain a complete data set that will enable us to assess the neighbouring graph resulting from this edge operation. However, there is a risk that this action may lead to biased estimates when missingness is not MCAR.

Because pairwise deletion leads to edge operations that are assessed based on different subsets of the data, it is possible to get stuck in an infinite loop where previous neighbouring graphs are constantly revisited and re-selected as a higher scoring graph. This can happen when, for example, DAG $\mathcal{G}_2$ returns a higher score than $\mathcal{G}_1$ based on pairwise deleted data set D_1, $\mathcal{G}_3$ returns a higher score than $\mathcal{G}_2$ based on pairwise deleted data set D_2, and $\mathcal{G}_1$ returns a higher score than $\mathcal{G}_3$ based on pairwise deleted data set D_3. In this example, HC with pairwise deletion would identify the graphical scores as $\mathcal{G}_1 < \mathcal{G}_2 < \mathcal{G}_3 < \mathcal{G}_1$ and never converge to a maximal solution. We address this issue by restricting HC search to neighbours not previously identified as the optimal graph. We call this variant of HC as HC-pairwise, and present its pseudo-code in Algorithm 9.

Algorithm 9 The HC-pairwise algorithm

```
 1: procedure HC-PAIRWISE
        Input: data set $D$
        Output: learnt DAG $\mathcal{G}$
 2:     $\mathcal{G} \leftarrow$ empty graph
 3:     $\mathcal{G}_{record} \leftarrow \{\mathcal{G}\}$
 4:     repeat
 5:         $\delta \leftarrow 0$
 6:         repeat
 7:             construct a neighbouring DAG $\mathcal{G}_{nei}$ by adding, reversing or deleting an edge from
                $\mathcal{G}$
 8:             if $\mathcal{G}_{nei} \notin \mathcal{G}_{record}$ then
 9:                 construct $D_{pw}$ by pairwise deleting $D$ given the necessary variables $\boldsymbol{W}$
10:                 if $S(\mathcal{G}_{nei} \mid D_{pw}) - S(\mathcal{G} \mid D_{pw}) > \delta$ then
11:                     $\delta \leftarrow S(\mathcal{G}_{nei} \mid D_{pw}) - S(\mathcal{G} \mid D_{pw})$
12:                     $\mathcal{G}_{update} \leftarrow \mathcal{G}_{nei}$
13:                 end if
14:             end if
15:         until all possible edge operations have been attempted
16:         if $\delta > 0$ then
17:             $\mathcal{G} \leftarrow \mathcal{G}_{update}$
18:             $\mathcal{G}_{record} = \mathcal{G}_{record} \cup \{\mathcal{G}\}$
19:         end if
20:     until $\delta = 0$
21:     return $\mathcal{G}$
22: end procedure
```

When data are MCAR on the basis of $\boldsymbol{R} \perp\!\!\!\perp \boldsymbol{V}$, the distribution entailed by any pairwise deleted data set is an unbiased estimate of the underlying true distribution:

$$P\left(V_i \mid \boldsymbol{Pa}_i, \boldsymbol{R}_s = \boldsymbol{0}\right) = P\left(V_i \mid \boldsymbol{Pa}_i\right), \tag{5.3}$$

where $\boldsymbol{R}_s$ can be any subset of $\boldsymbol{R}$.

From this, we derive Proposition 5.4.1, which states that, when the missingness is MCAR, the DAG learnt by HC-pairwise is a local maximum graph, at least when BIC is used as the objective function. We define the local maximum graph as the graph with an objective score not lower than the scores of all its valid neighbouring graphs, when these scores are derived from the fully observed data set; i.e., it is independent of missingness generated.

Proposition 5.4.1. *Assume data D is MCAR and sample size $N \to \infty$, for any DAG $\mathcal{G}$ and one of its neighbouring DAG $\mathcal{G}_{nei}$*

$$S_{BIC}\left(\mathcal{G}_{nei} \mid D_{pw}\right) > S_{BIC}\left(\mathcal{G} \mid D_{pw}\right), \textit{ iff } S_{BIC}\left(\mathcal{G}_{nei} \mid D_f\right) > S_{BIC}\left(\mathcal{G} \mid D_f\right),$$

where D_{pw} is the pairwise deleted data set which is derived from D by removing the data cases with missing values amongst the necessary variables $\boldsymbol{W}$, and D_f is the corresponding fully observed data set.

Proof We define the variables used in proof as follows: $\boldsymbol{V_d}$ is the set of variables with different parent-sets between a given DAG $\mathcal{G}$ and its neighbouring DAG $\mathcal{G}_{nei}$, $\boldsymbol{W}$ is a set of the necessary variables as defined in Equation 5.2, and N and N_{pw} are the sample sizes of the partially observed data set D and pairwise deleted data set D_{pw} respectively. Then, we have

$$\begin{aligned}
&S_{BIC}\left(\mathcal{G}_{nei} \mid D_f\right) - S_{BIC}\left(\mathcal{G} \mid D_f\right) \\
&\quad= \sum_{i=1}^{n} \big(S_{BIC}(V_i \mid \boldsymbol{Pa}_i^{nei}) - S_{BIC}(V_i \mid \boldsymbol{Pa}_i)\big) \\
&\quad= \sum_{i:V_i \in \boldsymbol{V}_d} \big(S_{BIC}(V_i \mid \boldsymbol{Pa}_i^{nei}) - S_{BIC}(V_i \mid \boldsymbol{Pa}_i)\big) \\
&\quad= \sum_{i:V_i \in \boldsymbol{V}_d} \left(\sum_{D_f} \big(\log P(V_i \mid \boldsymbol{Pa}_i^{nei}) - \log P(V_i \mid \boldsymbol{Pa}_i)\big) + \frac{\log(N)}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right) \right) \\
&\quad= \frac{N}{N_{pw}} \sum_{i:V_i \in \boldsymbol{V}_d} \left(\sum_{D_{pw}} \big(\log P(V_i \mid \boldsymbol{Pa}_i^{nei}, \boldsymbol{R_W} = \boldsymbol{0}) - \log P(V_i \mid \boldsymbol{Pa}_i, \boldsymbol{R_W} = \boldsymbol{0})\big) \right. \\
&\qquad\quad \left. + \frac{\log(N_{pw})}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right) + \frac{\log(N/N_{pw})}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right) \right) \qquad (5.4) \\
&\quad= \frac{N}{N_{pw}} \left(S_{BIC}\left(\mathcal{G}_{nei} \mid D_{pw}\right) - S_{BIC}\left(\mathcal{G} \mid D_{pw}\right) + \frac{\log(N/N_{pw})}{2} \sum_{i:V_i \in \boldsymbol{V}_d} \left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right) \right) \\
&\quad\propto S_{BIC}\left(\mathcal{G}_{nei} \mid D_{pw}\right) - S_{BIC}\left(\mathcal{G} \mid D_{pw}\right) + O(1). \qquad (5.5)
\end{aligned}$$

Equation 5.4 follows from Equation 5.3 given the MCAR assumption and large sample limit. Equation 5.5 is due to the missing rate of data D, i.e., N_{pw}/N, does not relate to the sample size N and remains constant with the increase of N. ■

5.4.2 Hill-Climbing with Inverse Probability Weighting

Although HC-pairwise will progressively learn a better DAG after each iteration when missingness is MCAR, this property does not necessarily hold when missingness is MAR or MNAR, since systematic bias in the data might produce

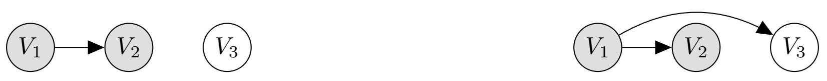
$$P\left(V_i \mid \boldsymbol{Pa}_i, \boldsymbol{R} = \boldsymbol{0}\right) \neq P\left(V_i \mid \boldsymbol{Pa}_i\right). \qquad (5.6)$$

To diminish data biases caused by potential dependencies between missing and observed data, we further explore applying the IPW method to the pairwise deleted data set.

According to Mohan et al. [2013, Theorem 2] and Tu et al. [2019], when the causal sufficiency assumption and Assumptions 5.3.1 and 5.3.2 hold, the joint distribution of variables $\boldsymbol{V}$ can be fully identified from the observed part of the data set (i,e., the data after applying pairwise deletion) by

$$P\left(\boldsymbol{V}\right) = \frac{P\left(\boldsymbol{V}, \boldsymbol{R} = \boldsymbol{0}\right)}{\prod_{R_i \in \boldsymbol{R}} P\left(R_i = 0 \mid \boldsymbol{Pa}_{R_i}, \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}} = \boldsymbol{0}\right)}, \qquad (5.7)$$

where $\boldsymbol{Pa}_{R_i}$ is the set of parents of missing indicator R_i, and $\boldsymbol{R}_{\boldsymbol{Pa}_{R_i}}$ is the set of missing indicator



(a) Current best DAG (b) Neighbouring DAG

Figure 5.2: A hill-climbing illustration of the DAG considered in Example 5.4.2, discussed in the main text. Shaded nodes represent partially observed variables.

of the partially observed variables in $\boldsymbol{Pa}_{R_i}$. Then, we have

$$
\begin{aligned}
P(\boldsymbol{V}) &= \frac{P(\boldsymbol{V}, \boldsymbol{R}=\mathbf{0})}{\prod_{R_i \in \boldsymbol{R}} P\left(R_i = 0 \mid \boldsymbol{Pa}_{R_i}, \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}} = \mathbf{0}\right)} \\
&= \frac{P(\boldsymbol{V} \mid \boldsymbol{R}=\mathbf{0}) P(\boldsymbol{R}=\mathbf{0})}{\prod_{R_i \in \boldsymbol{R}} P\left(R_i = 0 \mid \boldsymbol{Pa}_{R_i}, \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}} = \mathbf{0}\right)} \\
&= P(\boldsymbol{V} \mid \boldsymbol{R}=\mathbf{0}) \cdot \frac{P(\boldsymbol{R}=\mathbf{0})}{\prod_{R_i \in \boldsymbol{R}} \frac{P\left(\boldsymbol{Pa}_{R_i} \mid R_i=0, \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}}=\mathbf{0}\right) P\left(R_i=0 \mid \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}}=\mathbf{0}\right)}{P\left(\boldsymbol{Pa}_{R_i} \mid \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}}=\mathbf{0}\right)}} \\
&= P(\boldsymbol{V} \mid \boldsymbol{R}=\mathbf{0}) \cdot \\
&\qquad \underbrace{\frac{P(\boldsymbol{R}=\mathbf{0})}{\prod_{R_i \in \boldsymbol{R}} P\left(R_i = 0 \mid \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}} = \mathbf{0}\right)}}_{c} \prod_{R_i \in \boldsymbol{R}} \underbrace{\frac{P\left(\boldsymbol{Pa}_{R_i} \mid \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}} = \mathbf{0}\right)}{P\left(\boldsymbol{Pa}_{R_i} \mid R_i = 0, \boldsymbol{R}_{\boldsymbol{Pa}_{R_i}} = \mathbf{0}\right)}}_{\beta_{R_i}}. \qquad (5.8)
\end{aligned}
$$

Since the term c in Equation 5.8 represents a constant value, we can apply pairwise deletion to the missing data cases of variables $\boldsymbol{V}$ and weight the pairwise deleted data set by $\prod_{Ri \in R} \beta_{R_i}$. This will produce a weighted data set that approximates the unbiased distribution $P(\boldsymbol{V})$. We call this HC variant HC-IPW, and can be viewed as an extension of HC-pairwise that incorporates both the pairwise deletion and IPW methods. Unlike HC-pairwise, the HC-IPW algorithm can be used under the assumption the input data are MAR or MNAR, in addition to MCAR, to diminish data bias caused by systematic missing values.

It should be noted that when $\boldsymbol{Pa}_{R_i}$ contains partially observed variables, Equation 5.8 implies that $\boldsymbol{Pa}_{R_i} \subseteq \boldsymbol{V}$; otherwise, the columns of $\boldsymbol{Pa}_{R_i}$ in the pairwise deleted data set may contain missing values that will render the calculation of β_{R_i} invalid. The following example shows that it might be impossible to recover the underlying true distribution if any $\boldsymbol{Pa}_{R_i} \nsubseteq \boldsymbol{V}$.

Example 5.4.2. Consider that Figure 5.1c is the true m-graph, the current best DAG $\mathcal{G}$ in HC search is the one shown in Figure 5.2a, and Figure 5.2b presents one of its neighbouring DAGs, $\mathcal{G}_{nei}$. Since the difference in score between $\mathcal{G}_{nei}$ and $\mathcal{G}$ is $S(V_3 \mid V_1) - S(V_3)$, we need to ensure that missingness does not bias the estimate of distribution $P(V_1, V_3)$ when computing distributional score difference. If we apply pairwise deletion directly on the necessary variables $\{V_1, V_3\}$ and use Equation 5.8 to recover $P(V_1, V_3)$. This will result in the following equation:

$$
P(V_1, V_3) = P(V_1, V_3 \mid R_1 = 0) \frac{P(R_1 = 0)}{P(R_1 = 0)} \underbrace{\frac{P(V_2 \mid R_2 = 0)}{P(V_2 \mid R_1 = 0, R_2 = 0)}}_{\beta_{R_1}}.
$$

However, the problem in the above equation is that we cannot compute the weight term β_{R_1} for data cases that contain missing values in V_2.

To avoid this, when assessing the edge operations from $\mathcal{G}$ to $\mathcal{G}_{nei}$ in HC-IPW, the pairwise deletion for Equation 5.8 should be performed on *sufficient variables* $\boldsymbol{U}$, which is a variable set that contains the necessary variables $\boldsymbol{W}$ plus the parents of missing indicators of all variables in $\boldsymbol{U}$:

$$\boldsymbol{U} = \cup_{V_i \in \boldsymbol{V}_d} \left\{V_i, \boldsymbol{Pa}_i, \boldsymbol{Pa}_i^{nei}\right\} \cup \boldsymbol{Pa}_{\boldsymbol{R_U}}, \tag{5.9}$$

where $\boldsymbol{V}_d$ is the set of variables that have different parent-sets between $\mathcal{G}$ and $\mathcal{G}_{nei}$, and $\boldsymbol{Pa}_i$ and $\boldsymbol{Pa}_i^{nei}$ are the parent-sets of V_i in $\mathcal{G}$ and $\mathcal{G}_{nei}$ respectively. It is worth noting that Equation 5.9 represents a recursive process that iterates over the parents of missing indicators for all involved variables, i.e., not only $\boldsymbol{W}$ but also $\boldsymbol{U} \backslash \boldsymbol{W}$ should be included in $\boldsymbol{U}$ in order to resolve the issue illustrated in Example 5.4.2.

Another potential issue with Equation 5.8 is that the parents $\boldsymbol{Pa}_{R_i}$ of each missing indicator R_i are generally unknown. Tu et al. [2019] used constraint-based learning to discover the parents of each missing indicator, and this approach has been proven to be sound when the causal sufficiency assumption and Assumptions 5.3.1 and 5.3.2 hold. We have, therefore, adopted the constraint-based approach proposed by Tu et al. [2019] to discover the parents of the missing indicators in applying HC-IPW. The intention here is that this approach can be used to exclude variable V_j as the parent of R_i, if R_i is found to be independent of V_j given any variable set $\boldsymbol{S}$, given the pairwise deleted data set for $\{V_j\} \cup \boldsymbol{S}$. Algorithm 10 provides the pseudo-code.

Algorithm 10 Discovering the parents of the missing indicators using constraint-based learning

```
1: procedure DETECTING PARENTS OF MISSING INDICATORS
       Input: data set $D$
       Output: the parents of missing indicators $\boldsymbol{Pa_R}$
2:     for each $V_i \in \boldsymbol{V}_m$ do
3:         $\boldsymbol{Pa}_{R_i} \leftarrow \boldsymbol{V} \backslash V_i$
4:         for each $V_j \in \boldsymbol{V} \backslash V_i$ do
5:             remove $V_j$ from $\boldsymbol{Pa}_{R_i}$ if $R_i \perp\!\!\!\perp V_j \mid \boldsymbol{S}, R_j = 0, \boldsymbol{R_S} = \boldsymbol{0}$, for any $\boldsymbol{S} \subseteq \boldsymbol{Pa}_{R_i}$
6:         end for
7:     end for
8:     return $\boldsymbol{Pa_R}$
9: end procedure
```

Algorithm 11 describes the HC-IPW algorithm, where lines coloured in blue represent the difference in pseudo-code between HC-IPW and HC-pairwise. Note that when computing the objective score for HC-IPW, the weighted statistics $\widetilde{N}_{ijk}, \widetilde{N}_{ij}$ are used instead of the standard N_{ijk}, N_{ij} used in HC, and which are defined as follows:

$$\widetilde{N}_{ijk} = \sum_{s=1}^{|D_{pw}|} 1_{ijk}\left(d^s\right) \cdot \beta^s, \tag{5.10}$$

$$\widetilde{N}_{ij} = \sum_{k=1}^{r_i} \widetilde{N}_{ijk}, \tag{5.11}$$

where 1_{ijk} is the indicator function of the event $(V_i = k, \boldsymbol{Pa}_i = j)$ which returns 1 when the combination of $V_i = k, \boldsymbol{Pa}_i = j$ appears in the input data case, and returns 0 otherwise, d^s is the s^{th} record in pairwise deleted data set D_{pw}, and β^s is the weight corresponding to d^s. Therefore,

Algorithm 11 The HC-IPW algorithm

```
1: procedure HC-IPW
       Input: data set $D$
       Output: learnt DAG $\mathcal{G}$
2:     $\mathcal{G} \leftarrow$ empty graph
3:     $\mathcal{G}_{record} \leftarrow \{\mathcal{G}\}$
4:     retrieve the parents of missing indicators via Algorithm 10
5:     repeat
6:         $\delta \leftarrow 0$
7:         repeat
8:             construct a neighbouring DAG $\mathcal{G}_{nei}$ by adding, reversing or deleting an edge from $\mathcal{G}$
9:             if $\mathcal{G}_{nei} \notin \mathcal{G}_{record}$ then
10:                construct $D_{pw}$ by pairwise deleting $D$ given the sufficient variables $\boldsymbol{U}$
11:                compute weight $\beta$ by Equation 5.8 for $D_{pw}$
12:                if $S(\mathcal{G}_{nei} \mid D_{pw}, \beta) - S(\mathcal{G} \mid D_{pw}, \beta) > \delta$ then
13:                    $\delta \leftarrow S(\mathcal{G}_{nei} \mid D_{pw}, \beta) - S(\mathcal{G} \mid D_{pw}, \beta)$
14:                    $\mathcal{G}_{update} \leftarrow \mathcal{G}_{nei}$
15:                end if
16:            end if
17:        until all possible edge operations have been attempted
18:        if $\delta > 0$ then
19:            $\mathcal{G} \leftarrow \mathcal{G}_{update}$
20:            $\mathcal{G}_{record} = \mathcal{G}_{record} \cup \{\mathcal{G}\}$
21:        end if
22:    until $\delta = 0$
23: end procedure
```

we define the BIC score for pairwise deleted data set D_{pw} given β as follows:

$$S_{BIC}\left(\mathcal{G} \mid D_{pw}, \beta\right) = \sum_{i=1}^{n}\left(\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} \widetilde{N}_{ijk} \cdot \log\frac{\widetilde{N}_{ijk}}{\widetilde{N}_{ij}} - \frac{\log\left(N_{pw}\right)}{2} \cdot (r_i - 1)q_i\right),$$

where N_{pw} represents the sample size of D_{pw}, $\widetilde{N}_{ijk}$ and $\widetilde{N}_{ij}$ represent the weighted statistics as defined in Equation 5.10 and 5.11, and β is used for computing the weighted $\widetilde{N}_{ijk}$ and $\widetilde{N}_{ij}$.

The following proposition shows that HC-IPW converges to a local optima when BIC is used as the score function, when the causal sufficiency assumption and Assumptions 5.3.1 and 5.3.2 hold, and when sample size $N \to \infty$.

Proposition 5.4.2. *Given the causal sufficiency assumption, Assumptions 5.3.1 and 5.3.2, assume data D is partially observed and sample size $N \to \infty$, for any DAG $\mathcal{G}$ and one of its neighbouring DAG $\mathcal{G}_{nei}$*

$$S_{BIC}\left(\mathcal{G}_{nei} \mid D_{pw}, \beta\right) > S_{BIC}\left(\mathcal{G} \mid D_{pw}, \beta\right), \textit{ iff } S_{BIC}\left(\mathcal{G}_{nei} \mid D_f\right) > S_{BIC}\left(\mathcal{G} \mid D_f\right),$$

where D_{pw} is the pairwise deleted data set which is derived from D by removing data cases with missing values amongst sufficient variables $\boldsymbol{U}$, $\beta = \prod_{R_i \in \boldsymbol{R_U}} \beta_{R_i}$, and D_f is the corresponding fully observed data set.

Proof We define the variables used in this proof as we did in the proof of Proposition 5.4.1. We also define $\boldsymbol{W}$ as a set of the necessary variables as introduced in Equation 5.2. Then, we have

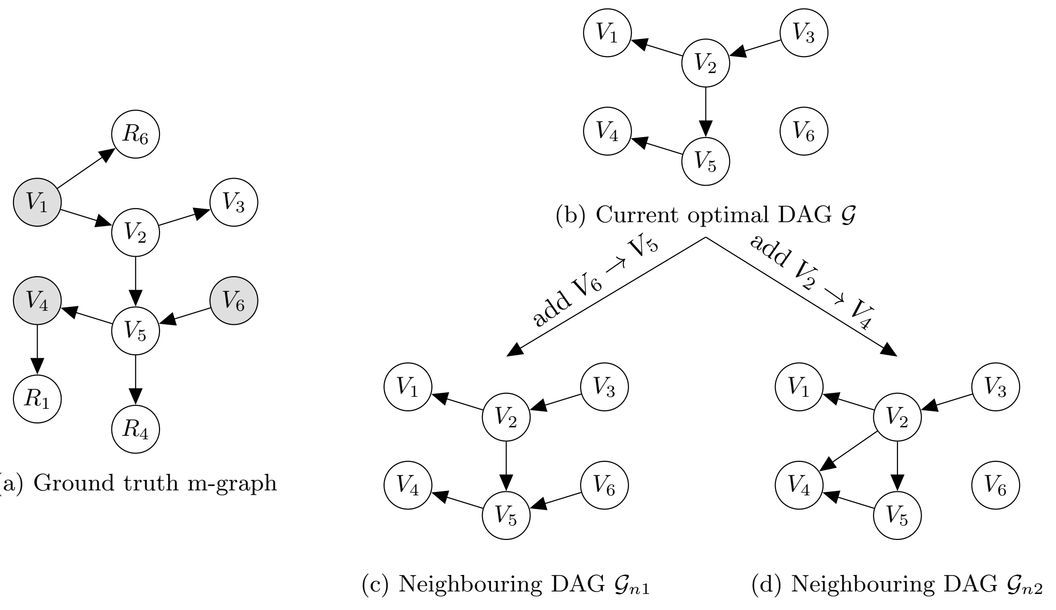
$$
\begin{aligned}
& S_{BIC}\left(\mathcal{G}_{nei} \mid D_f\right) - S_{BIC}\left(\mathcal{G} \mid D_f\right) \\
&= \sum_{i:V_i \in \boldsymbol{V}_d} \left(\sum_{D_f} \left(\log P(V_i \mid \boldsymbol{Pa}_i^{nei}) - \log P(V_i \mid \boldsymbol{Pa}_i)\right) + \frac{\log(N)}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right)\right) \\
&= \sum_{i:V_i \in \boldsymbol{V}_d} \left(\sum_{D_f} \left(\log \frac{P(V_i, \boldsymbol{Pa}_i^{nei})}{\sum_{V_i} P(V_i, \boldsymbol{Pa}_i^{nei})} - \log \frac{P(V_i, \boldsymbol{Pa}_i)}{\sum_{V_i} P(V_i, \boldsymbol{Pa}_i)}\right) + \frac{\log(N)}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right)\right) \\
&= \frac{N}{N_{pw}} \sum_{i:V_i \in \boldsymbol{V}_d} \left(\sum_{D_{pw}} \left(\log \frac{P(V_i, \boldsymbol{Pa}_i^{nei} \mid \boldsymbol{R_U} = \boldsymbol{0})\beta}{\sum_{V_i} P(V_i, \boldsymbol{Pa}_i^{nei} \mid \boldsymbol{R_U} = \boldsymbol{0})\beta} - \log \frac{P(V_i, \boldsymbol{Pa}_i \mid \boldsymbol{R_U} = \boldsymbol{0})\beta}{\sum_{V_i} P(V_i, \boldsymbol{Pa}_i \mid \boldsymbol{R_U} = \boldsymbol{0})\beta}\right) \right. \\
&\qquad \left. + \frac{\log(N)}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right)\right) \qquad (5.12) \\
&= \frac{N}{N_{pw}} \sum_{i:V_i \in \boldsymbol{V}_d} \left(\sum_{j=1}^{|\boldsymbol{Pa}_i^{nei}|} \sum_{k=1}^{|V_i|} \widetilde{N}_{ijk} \log \frac{\widetilde{N}_{ijk}}{\widetilde{N}_{ij}} - \sum_{j=1}^{|\boldsymbol{Pa}_i|} \sum_{k=1}^{|V_i|} \widetilde{N}_{ijk} \log \frac{\widetilde{N}_{ijk}}{\widetilde{N}_{ij}} + \frac{\log(N)}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right)\right) \\
&= \frac{N}{N_{pw}} \sum_{i:V_i \in \boldsymbol{V}_d} \left(\sum_{j=1}^{|\boldsymbol{Pa}_i^{nei}|} \sum_{k=1}^{|V_i|} \widetilde{N}_{ijk} \log \frac{\widetilde{N}_{ijk}}{\widetilde{N}_{ij}} - \sum_{j=1}^{|\boldsymbol{Pa}_i|} \sum_{k=1}^{|V_i|} \widetilde{N}_{ijk} \log \frac{\widetilde{N}_{ijk}}{\widetilde{N}_{ij}} + \frac{\log(N_{pw})}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right) \right. \\
&\qquad \left. + \frac{\log(N/N_{pw})}{2}\left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right)\right) \\
&= \frac{N}{N_{pw}} \left(S_{BIC}\left(\mathcal{G}_{nei} \mid D_{pw}, \beta\right) - S_{BIC}\left(\mathcal{G} \mid D_{pw}, \beta\right) + \frac{\log(N/N_{pw})}{2} \sum_{i:V_i \in \boldsymbol{V}_d} \left(|\hat{\Theta}_i^{nei}| - |\hat{\Theta}_i|\right)\right) \\
&\propto S_{BIC}\left(\mathcal{G}_{nei} \mid D_{pw}, \beta\right) - S_{BIC}\left(\mathcal{G} \mid D_{pw}, \beta\right) + O(1).
\end{aligned}
$$

In the above equations, $\beta = \prod_{R_i \in \boldsymbol{R_U}} \beta_{R_i}$, $\widetilde{N}_{ijk}$ and $\widetilde{N}_{ij}$ are defined by Equation 5.10 and 5.11. Equation 5.12 is a consequence of the identifiablity of $P(\boldsymbol{U})$ given Equation 5.8. ■

5.4.3 Hill-Climbing with adaptive Inverse Probability Weighting

Although HC-IPW diminishes potential data bias caused by systematic missing values, the learning approach achieves this by removing a greater number of data cases compared to those removed by HC-pairwise when $\boldsymbol{Pa}_{\boldsymbol{R_W}}$ contains partially observed variables, which is likely to happen when the missingness are MNAR. This can be a problem when data cases are limited. We illustrate this phenomenon with an example.

Example 5.4.3. Suppose graph (a) in Figure 5.3 represents the ground truth m-graph in which the variables in shaded backcolour V_1, V_4 and V_6 are partially observed whose missingness are caused by V_4, V_5 and V_1 respectively, as illustrated with the missing indicators R_1, R_4 and R_6 corresponding to the missingness of V_1, V_4 and V_6. Let us assume graph (b) represents the current state of the optimal DAG in the HC-pairwise/HC-IPW search process, and that graphs (c) and (d) represent two of the possible neighbouring graphs. When HC-pairwise compares $\mathcal{G}$ with $\mathcal{G}_{n1}$, it applies pairwise deletion to cases in which the necessary variables $\boldsymbol{W} = \{V_5, V_2, V_6\}$ contain missing values. Since only V_6 is partially observed out of the three necessary variables, HC-pairwise removes data cases when the value of V_6 is missing. In contrast, when HC-IPW is applied to this case,



(a) Ground truth m-graph

(b) Current optimal DAG $\mathcal{G}$

(c) Neighbouring DAG $\mathcal{G}_{n1}$

(d) Neighbouring DAG $\mathcal{G}_{n2}$

Figure 5.3: Example of a searching step in HC-pairwise/HC-IPW.

and assuming it correctly learns the parents of missingness via Algorithm 10, it computes the weights of the pairwise deleted data set through pairwise deletion based on the sufficient variables $\boldsymbol{U} = \{V_5, V_2, V_6\} \cup \{V_1, V_4, V_5\}$. Thus, HC-IPW removes data cases whenever any of the variables in U contain a missing value (in this example, V_1, V_4 and V_6 do). Therefore, HC-IPW performs learning on a smaller set of data cases compared to those in the case of HC-pairwise.

When $\boldsymbol{Pa_{R_W}}$ (refer to Equation 5.2 and Algorithm 11) does not contain any partially observed variables, the HC-IPW algorithm will perform learning on the same number of data cases as in HC-pairwise. This can happen in cases such as when comparing neighbouring DAG $\mathcal{G}_{n2}$ against $\mathcal{G}$ in Figure 5.3, where the set of necessary variables $\boldsymbol{W}$ in HC-pairwise contains $\{V_4, V_2, V_5\}$ and the set of sufficient variables $\boldsymbol{U}$ in HC-IPW is $\{V_4, V_2, V_5\} \cup \{V_5\}$. In this case, because V_5 is fully observed, applying pairwise deletion given $\boldsymbol{W}$ and $\boldsymbol{U}$ would result in the same pairwise deleted data set.

Because the effectiveness of a scoring function increases with sample size, the scoring efficiency of HC-IPW can decrease considerably when missingness are MNAR for multiple variables. This is because both the number of partially observed variables and MNAR missingness increase the number of data cases removed during the learning process. It is on this basis we investigated a third variant, called the adaptive IPW-based HC (HC-aIPW), and which can be viewed as an extension of HC-IPW. The pseudo-code of HC-aIPW is shown in Algorithm 12. The highlighted section represents the part of the code that differs from HC-IPW.

In essence, HC-aIPW aims to maximise the samples taken into consideration during the learning process. When there are partially observed variables in $\boldsymbol{Pa_{R_W}}$, HC-aIPW applies pairwise deletion given $\boldsymbol{W}$ and computes the difference in score between the current optimal DAG and the neighbouring DAG using the original pairwise deleted data set and standard scoring function. This is the only difference between HC-aIPW and HC-IPW. When there are no partially observed variables in $\boldsymbol{Pa_{R_W}}$, HC-aIPW uses the same IPW procedure as in HC-IPW to compute the difference in score between the current optimal DAG and the neighbouring DAG given the weighted pairwise deleted data set.

Algorithm 12 The HC-aIPW algorithm

```
1:  procedure HC-aIPW
        Input: data set $D$
        Output: learnt DAG $\mathcal{G}$
2:      $\mathcal{G} \leftarrow$ empty graph
3:      $\mathcal{G}_{record} \leftarrow \{\mathcal{G}\}$
4:      retrieve the parents of missing indicators via Algorithm 10
5:      repeat
6:          $\delta \leftarrow 0$
7:          repeat
8:              construct a neighbouring DAG $\mathcal{G}_{nei}$ by adding, reversing or deleting an edge from $\mathcal{G}$
9:              if $\mathcal{G}_{nei} \notin \mathcal{G}_{record}$ then
10:                 if $\boldsymbol{Pa}_{\boldsymbol{R_W}} \cap \boldsymbol{V}_m \neq \emptyset$ then
11:                     construct $D_{pw}$ by pairwise deleting $D$ given the necessary variables $\boldsymbol{W}$
12:                     if $S(\mathcal{G}_{nei} \mid D_{pw}) - S(\mathcal{G} \mid D_{pw}) > \delta$ then
13:                         $\delta \leftarrow S(\mathcal{G}_{nei} \mid D_{pw}) - S(\mathcal{G} \mid D_{pw})$
14:                         $\mathcal{G}_{update} \leftarrow \mathcal{G}_{nei}$
15:                     end if
16:                 else
17:                     construct $D_{pw}$ by pairwise deleting $D$ given the sufficient variables $\boldsymbol{U}$
18:                     compute weight $\beta$ by Equation 5.8 for $D_{pw}$
19:                     if $S(\mathcal{G}_{nei} \mid D_{pw}, \beta) - S(\mathcal{G} \mid D_{pw}, \beta) > \delta$ then
20:                         $\delta \leftarrow S(\mathcal{G}_{nei} \mid D_{pw}, \beta) - S(\mathcal{G} \mid D_{pw}, \beta)$
21:                         $\mathcal{G}_{update} \leftarrow \mathcal{G}_{nei}$
22:                     end if
23:                 end if
24:             end if
25:         until all possible edge operations have been attempted
26:         if $\delta > 0$ then
27:             $\mathcal{G} \leftarrow \mathcal{G}_{update}$
28:             $\mathcal{G}_{record} = \mathcal{G}_{record} \cup \{\mathcal{G}\}$
29:         end if
30:     until $\delta = 0$
31:     return $\mathcal{G}$
32: end procedure
```

Name	Number of variables	Average degree	Number of states
Asia	8	2.00	2
Alarm	37	2.49	$2 \sim 4$
Pathfinder	109	3.58	$2 \sim 63$
Sports	9	3.33	$3 \sim 8$
ForMed	88	3.14	$2 \sim 10$
Property	27	2.30	$2 \sim 7$

Table 5.2: The properties of the six real-world BNs.

5.5 Experiments

The learning accuracy of each of the three algorithms described in Section 5.4 is investigated and evaluated with reference to the Structural EM algorithm when applied to the same data. We were unable to include the other algorithms mentioned in Section 5.2 as part of the evaluation, since their implementations are not made publicly available. The Structural EM algorithm represents a state-of-the-art score-based approach for structure learning from missing data, and also explores the search space of graphs using HC. Since all the involved algorithms are based on HC, we measure their learning accuracy with reference to the results obtained when applying standard HC on complete, rather than incomplete, data. Results from complete data give us the empirical maximum performance we can achieve on these data sets using HC, before making part of the data missing. The HC and Structural EM algorithms used in this Chapter are those available in the bnlearn R package [Scutari et al., 2010]. It is worth noting that the Structural EM algorithm implemented in bnlearn R package is based on single imputation rather than belief propagation. Therefore, the results presented in this Chapter approximate the difference between the proposed methods and Friedman's Structural EM. The implementations of the three HC variants described in Section 5.4 are available online at `https://github.com/Enderlogic/HC-missing-data`.

5.5.1 Generating synthetic data and missingness

To illustrate the performance of the algorithms under different settings, we consider three types of ground truth DAGs: sparse networks, dense networks and real-world networks. We have constructed 50 random sparse and 50 random dense DAGs. Each network contains 20 to 50 nodes with two to six states per node. A sparse DAG $\mathcal{G}$ with n variables is generated from a randomly ordered variable set $V_1 < V_2 < \ldots < V_n$, where directed edges are sampled from lower ordered variables to higher ordered variables with probability $2/(n-1)$. Dense DAGs are generated with the same procedure, but the probability of drawing an edge between variables increases to $4/(n-1)$. The conditional probability distribution of variable V_i in sparse and dense DAGs is parameterised, given any configuration of its parents, by drawing a random number from the Dirichlet distribution $\text{Dir}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} = \underbrace{\{1, \ldots, 1\}}_{r_i}$, and r_i is the number of states in V_i. For real-world DAGs, we use the six real-world BNs investigated in Subsection 3.2.1. The structure and parameters of these BNs are set by either real data observations or prior knowledge as defined in the original studies. The properties of these BNs are provided in Table 5.2.

We generate complete and incomplete synthetic data using the DAGs introduced above. The complete data sets are provided as input to the standard HC algorithm, whereas the corresponding incomplete data sets are provided as input to the Structural EM and the three HC variants described in Section 5.4. We generate five complete data sets per DAG with sample sizes $N \in \{100, 500, 1000, 5000, 10000\}$. Each complete data set is then used to construct further three data sets with missing values; one per missingness assumption, MCAR, MAR or MNAR. For the MCAR case, we randomly select 50% of the variables to represent the partially observed variables, and we then remove observed data of these variables with probability p, where p represents a random value between 0.1 and 0.6. For case MAR, we had to ensure missingness are dependent on a subset of the fully observed variables, and this is done as follows:

1. Randomly select 50% of the variables as partially observed variables (same process as in MCAR);

2. Randomly assign a fully observed variable as the parent of missingness of a partially observed variable (repeat for all partially observed variables);

3. Remove observations in partially observed variables with probability $p = 0.6$ when the parent of their missingness is at its highest occurring state; otherwise, remove the observation with probability $p = 0.1$.

Generating MNAR data also involves the above 3-step procedure, but step 2 is modified as follows:

2. Randomly select 50% of the partially observed variables and randomly assign a fully observed variable as the parent of their missingness. For the remaining 50% partially observed variables, randomly assign another partially observed variable as the parent of their missingness.

5.5.2 Evaluation metrics

The structure learning performance is assessed using the F_1 score and the re-scaled SHD score. Because the experiments are based on observational data, multiple DAGs can be statistically indistinguishable due to being part of the same Markov Equivalence class. On this basis, we compare the CPDAGs between the learnt and true graphs to measure both the F_1 and SHD graphical scores.

5.5.3 Results when the true DAG is sparse

Figure 5.4 presents the average accuracy of the algorithms when the true DAGs are sparse. Each averaged score is derived from 50 CPDAGs, corresponding to each of the 50 randomly generated sparse DAGs. Tables 5.3 and 5.4 provide the mean and standard deviation of the scores. The results suggest that the two evaluation metrics are generally consistent in ranking the algorithms from best to worst performance. Both metrics suggest that all of the three proposed HC variants outperform the Structural EM algorithm when the sample size is greater than 1,000, under all three missingness scenarios MCAR, MAR and MNAR. Interestingly, the HC-aIPW algorithm almost matches the performance of HC which is applied to complete data (denoted as HC-complete in Figures 5.4), particularly for experiments with 10,000 sample size, and this observation is consistent across all three missingness assumptions. The three variants, HC-pairwise, HC-IPW and HC-aIPW, produce very similar results under MCAR, and this is because missingness under MCAR has no pattern that could be identified by the HC-IPW and HC-aIPW variants. That is, when HC-IPW and HC-aIPW do not discover any parent of missingness, they follow the search process of HC-pairwise. Under MAR, however, both HC-IPW and HC-aIPW outperform HC-pairwise as well as Structural

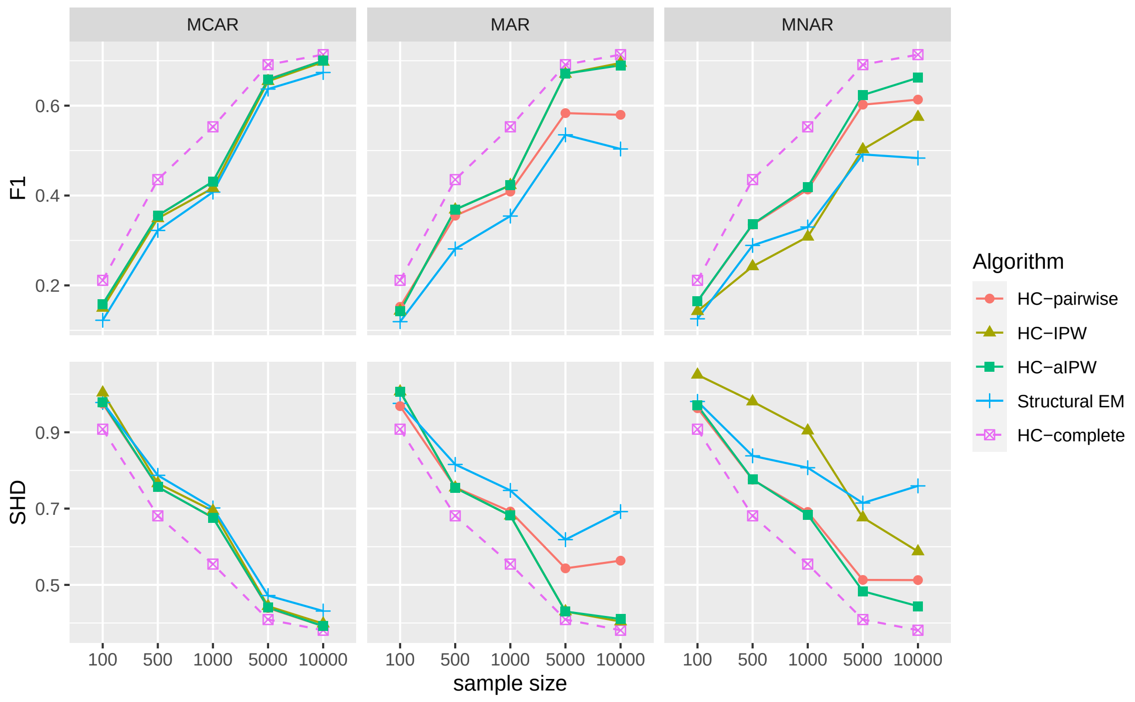


Figure 5.4: Average F_1 and re-scaled SHD scores learnt by HC-pairwise, HC-IPW, HC-aIPW and Structural EM for sparse networks, under different assumptions of missingness and sample sizes. Each score represents the average score over 50 CPDAGs. Note the scores of HC-complete are based on complete data for benchmarking purposes; i.e., the same scores are superimposed in all three missingness cases as a dashed line.

Data	Sample size	Structural EM	HC-pairwise	HC-IPW	HC-aIPW
MCAR	100	0.122 ± 0.088	0.158 ± 0.108	0.150 ± 0.104	$\mathbf{0.159 \pm 0.107}$
	500	0.325 ± 0.139	$\mathbf{0.356 \pm 0.139}$	0.349 ± 0.133	0.355 ± 0.139
	1000	0.410 ± 0.141	0.430 ± 0.149	0.417 ± 0.143	$\mathbf{0.431 \pm 0.150}$
	5000	0.642 ± 0.149	$\mathbf{0.659 \pm 0.144}$	0.654 ± 0.160	0.658 ± 0.144
	10000	0.682 ± 0.135	0.700 ± 0.140	0.697 ± 0.143	$\mathbf{0.700 \pm 0.137}$
MAR	100	0.117 ± 0.097	$\mathbf{0.152 \pm 0.102}$	0.143 ± 0.102	0.142 ± 0.101
	500	0.281 ± 0.119	0.355 ± 0.117	0.369 ± 0.140	$\mathbf{0.369 \pm 0.138}$
	1000	0.354 ± 0.136	0.409 ± 0.152	0.423 ± 0.150	$\mathbf{0.423 \pm 0.149}$
	5000	0.543 ± 0.119	0.583 ± 0.137	$\mathbf{0.671 \pm 0.147}$	0.671 ± 0.150
	10000	0.505 ± 0.141	0.580 ± 0.121	$\mathbf{0.695 \pm 0.136}$	0.690 ± 0.138
MNAR	100	0.127 ± 0.094	0.164 ± 0.098	0.143 ± 0.103	$\mathbf{0.165 \pm 0.099}$
	500	0.285 ± 0.122	0.335 ± 0.129	0.242 ± 0.091	$\mathbf{0.336 \pm 0.131}$
	1000	0.328 ± 0.123	0.413 ± 0.142	0.308 ± 0.113	$\mathbf{0.419 \pm 0.148}$
	5000	0.488 ± 0.137	0.602 ± 0.164	0.503 ± 0.124	$\mathbf{0.624 \pm 0.150}$
	10000	0.473 ± 0.148	0.613 ± 0.144	0.575 ± 0.157	$\mathbf{0.662 \pm 0.146}$

Table 5.3: Mean and standard deviation of F_1 scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for sparse networks, under the different assumptions of missingness and sample sizes.

Data	Sample size	Structural EM	HC-pairwise	HC-IPW	HC-aIPW
MCAR	100	0.978 ± 0.063	$\mathbf{0.975 \pm 0.095}$	1.004 ± 0.107	0.978 ± 0.093
	500	0.784 ± 0.121	$\mathbf{0.756 \pm 0.127}$	0.766 ± 0.120	0.757 ± 0.127
	1000	0.700 ± 0.133	0.677 ± 0.147	0.694 ± 0.140	$\mathbf{0.676 \pm 0.147}$
	5000	0.465 ± 0.181	$\mathbf{0.439 \pm 0.178}$	0.444 ± 0.191	0.441 ± 0.179
	10000	0.424 ± 0.178	0.392 ± 0.182	0.398 ± 0.183	$\mathbf{0.392 \pm 0.178}$
MAR	100	0.975 ± 0.086	$\mathbf{0.969 \pm 0.094}$	1.007 ± 0.101	1.007 ± 0.101
	500	0.814 ± 0.103	0.756 ± 0.108	0.755 ± 0.131	$\mathbf{0.754 \pm 0.129}$
	1000	0.748 ± 0.123	0.692 ± 0.160	$\mathbf{0.681 \pm 0.156}$	0.682 ± 0.154
	5000	0.609 ± 0.158	0.543 ± 0.174	$\mathbf{0.430 \pm 0.186}$	0.430 ± 0.189
	10000	0.690 ± 0.188	0.563 ± 0.161	$\mathbf{0.404 \pm 0.181}$	0.411 ± 0.184
MNAR	100	0.984 ± 0.068	$\mathbf{0.963 \pm 0.078}$	1.051 ± 0.129	0.970 ± 0.082
	500	0.836 ± 0.102	$\mathbf{0.776 \pm 0.114}$	0.980 ± 0.116	0.777 ± 0.115
	1000	0.810 ± 0.119	0.691 ± 0.140	0.904 ± 0.163	$\mathbf{0.684 \pm 0.149}$
	5000	0.721 ± 0.184	0.513 ± 0.201	0.676 ± 0.177	$\mathbf{0.483 \pm 0.185}$
	10000	0.774 ± 0.201	0.513 ± 0.185	0.588 ± 0.203	$\mathbf{0.444 \pm 0.184}$

Table 5.4: Mean and standard deviation of re-scaled SHD scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for sparse networks, under the different assumptions of missingness and sample sizes.

EM when the sample size is larger than 100 and the improvement in performance increases with sample size. From this observation, we can conclude that the IPW method successfully eliminates most of the distributional bias. Interestingly, although the construction of the Structural EM algorithm is based on the MAR assumption, its performance under MAR is considerably lower than its performance under MCAR. A possible explanation is that the single imputation process the *bnlearn* R package employs during the E step of Structural EM, instead of belief propagation, is unable to capture the uncertainty of the missing values.

Lastly, the results under MNAR suggest that HC-IPW generally performs worse than HC-pairwise across most sample sizes. This observation can be explained by the reduced sample size on which HC-IPW operates, relative to HC-pairwise, as discussed in subsection 5.4.3. Specifically, when the parents of missingness of necessary variables W contain partially observed variables (i.e., MNAR case), HC-IPW applies pairwise deletion by taking into consideration a higher number of variables compared to those considered by HC-pairwise. This means that, compared to HC-pairwise, the HC-IPW algorithm typically evaluates edge operations based on smaller samples when missingness are MNAR, which tends to yield less accurate results. From this, we can also conclude that the negative effect resulting from HC-IPW further pruning samples has not been offset by the data bias adjustments applied by the IPW method. On the other hand, the HC-aIPW algorithm which is designed to apply the IPW method only when no additional samples would be deleted compared to HC-pairwise, generally outperforms all other algorithms under MNAR, particularly under higher sample sizes.

5.5.4 Results when the true DAG is dense

In this subsection we investigate the performance of the algorithms when applied to data sets sampled from dense networks. The performance of each algorithm is depicted in Figure 5.5, and detailed results are provided in Tables 5.6 and 5.7. An important distinction between sparse and dense networks is that learning from data sampled from dense networks makes it more likely that

Data	Sample size	Structural EM	HC-pairwise	HC-IPW	HC-aIPW
MCAR	100	161.29 ± 54.37	1.55 ± 0.68	8.02 ± 2.27	8.30 ± 1.95
	500	423.61 ± 146.51	1.55 ± 0.43	6.28 ± 1.50	6.89 ± 1.47
	1000	556.72 ± 178.34	1.54 ± 0.36	5.88 ± 1.68	6.44 ± 1.78
	5000	739.52 ± 269.94	1.73 ± 0.70	8.76 ± 2.48	9.24 ± 2.38
	10000	642.91 ± 254.91	1.63 ± 0.42	11.46 ± 3.45	11.90 ± 3.57
MAR	100	164.96 ± 46.26	1.64 ± 0.51	8.33 ± 2.15	9.14 ± 2.29
	500	421.37 ± 144.99	1.62 ± 0.42	7.50 ± 2.00	8.42 ± 2.09
	1000	554.72 ± 186.89	1.55 ± 0.34	7.73 ± 2.29	8.62 ± 2.47
	5000	740.71 ± 275.59	1.59 ± 0.32	14.81 ± 5.49	15.33 ± 5.24
	10000	657.52 ± 313.70	1.73 ± 0.53	22.49 ± 7.52	22.75 ± 7.20
MNAR	100	154.04 ± 44.86	1.54 ± 0.56	8.80 ± 3.27	8.79 ± 2.67
	500	419.91 ± 145.50	1.62 ± 0.44	8.18 ± 3.58	7.90 ± 2.47
	1000	552.16 ± 181.62	1.58 ± 0.41	7.69 ± 3.05	7.56 ± 2.06
	5000	776.32 ± 359.57	1.87 ± 0.87	13.29 ± 5.93	12.89 ± 7.74
	10000	615.91 ± 271.88	1.86 ± 0.77	20.07 ± 9.38	17.93 ± 8.38

Table 5.5: Mean and standard deviation of execution times produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for sparse networks and relative to HC when applied to complete data, under the different assumptions of missingness and sample sizes.

local parts of the graph will involve learning from partially observed variables. In other words, the effect of missing values is more severe on dense, compared to sparse, networks as shown in Subsection 5.5.3.

The results show that the HC-aIPW algorithm continues to perform best in the case of denser graphs, in terms of overall performance and over the different missingness and sample size assumptions. Specifically, HC-aIPW achieves the highest accuracy in 11 and 8 cases in terms of F_1 and SHD measures respectively, out of the 15 experiments conducted in this subsection. In contrast, the Structural EM algorithm performs best in just two experiments and only in terms of the SHD score. However, compared with the results in subsection 5.5.3, the divergence in score between Structural EM and HC-based variants is much smaller.

The performance across the three HC-based variants appears to be similar to that obtained under sparse graphs. When data are MCAR, HC-IPW and HC-aIPW produce scores that are similar to those produced by HC-pairwise, and this is expected since no observed variables should be detected as the parents of missing indicators when missingness is MCAR. When data are MAR, both HC-IPW and HC-aIPW outperform HC-pairwise since, unlike HC-pairwise, they can detect and reduce bias caused by missing values. Lastly, when data are MNAR, HC-IPW performs worst amongst all algorithms, particularly when the sample size is lowest, and this is because it tends to remove a large number of data cases when computing the local scores. On the other hand, HC-aIPW (which aims to resolve this specific drawback of HC-IPW) performs best in almost all the MNAR experiments. The consistency of the results across sparse and dense networks suggests that the performance of HC-aIPW, relative to the other algorithms considered in this study, is not sensitive to the sparsity of the network that generates the input data.

5.5.5 Results when the true DAG is a real-world network

Lastly, we apply the algorithms to data sets sampled from the six real-world networks. Figure 5.6 shows the average performance of the algorithms across all the six real-world networks and over

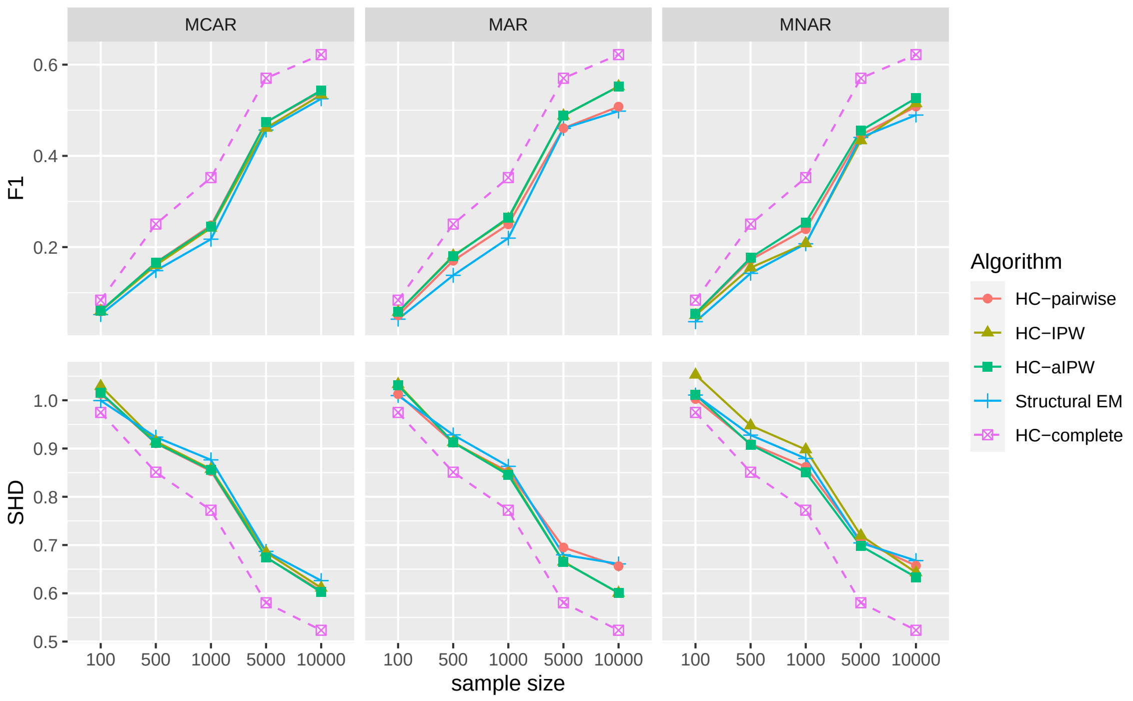


Figure 5.5: Average F_1 and re-scaled SHD scores learnt by HC-pairwise, HC-IPW, HC-aIPW and Structural EM for dense networks, under different assumptions of missingness and sample sizes. Each score represents the average score over 50 CPDAGs. Note the scores of HC-complete are based on complete data for benchmarking purposes; i.e., the same scores are superimposed in all three missingness cases as a dashed line.

Data	Sample size	Structural EM	HC-pairwise	HC-IPW	HC-aIPW
MCAR	100	0.052 ± 0.043	0.059 ± 0.046	$\mathbf{0.060 \pm 0.045}$	0.060 ± 0.046
	500	0.148 ± 0.070	0.166 ± 0.077	0.160 ± 0.071	$\mathbf{0.166 \pm 0.076}$
	1000	0.217 ± 0.094	$\mathbf{0.248 \pm 0.089}$	0.242 ± 0.088	0.245 ± 0.092
	5000	0.457 ± 0.136	$\mathbf{0.474 \pm 0.124}$	0.461 ± 0.115	$\mathbf{0.474 \pm 0.124}$
	10000	0.525 ± 0.140	0.541 ± 0.151	0.534 ± 0.147	$\mathbf{0.544 \pm 0.148}$
MAR	100	0.042 ± 0.048	0.050 ± 0.052	0.058 ± 0.053	$\mathbf{0.058 \pm 0.051}$
	500	0.138 ± 0.071	0.170 ± 0.081	$\mathbf{0.181 \pm 0.081}$	0.180 ± 0.078
	1000	0.220 ± 0.109	0.250 ± 0.095	0.262 ± 0.086	$\mathbf{0.265 \pm 0.088}$
	5000	0.460 ± 0.124	0.461 ± 0.121	0.488 ± 0.129	$\mathbf{0.488 \pm 0.128}$
	10000	0.498 ± 0.118	0.508 ± 0.126	$\mathbf{0.552 \pm 0.132}$	0.552 ± 0.133
MNAR	100	0.036 ± 0.040	0.054 ± 0.053	0.051 ± 0.054	$\mathbf{0.054 \pm 0.052}$
	500	0.143 ± 0.066	0.172 ± 0.083	0.155 ± 0.064	$\mathbf{0.177 \pm 0.084}$
	1000	0.207 ± 0.087	0.239 ± 0.093	0.208 ± 0.077	$\mathbf{0.254 \pm 0.100}$
	5000	0.440 ± 0.123	0.446 ± 0.118	0.434 ± 0.126	$\mathbf{0.455 \pm 0.124}$
	10000	0.490 ± 0.125	0.508 ± 0.115	0.515 ± 0.127	$\mathbf{0.526 \pm 0.127}$

Table 5.6: Mean and standard deviation of F_1 scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for dense networks, under the different assumptions of missingness and sample sizes.

Data	Sample size	Structural EM	HC-pairwise	HC-IPW	HC-aIPW
MCAR	100	$\mathbf{0.999 \pm 0.033}$	1.013 ± 0.036	1.029 ± 0.046	1.015 ± 0.037
	500	0.924 ± 0.047	$\mathbf{0.911 \pm 0.053}$	0.915 ± 0.050	$\mathbf{0.911 \pm 0.053}$
	1000	0.876 ± 0.068	$\mathbf{0.853 \pm 0.072}$	0.857 ± 0.071	0.855 ± 0.074
	5000	0.687 ± 0.137	$\mathbf{0.674 \pm 0.131}$	0.685 ± 0.118	0.675 ± 0.131
	10000	0.626 ± 0.156	0.605 ± 0.175	0.611 ± 0.169	$\mathbf{0.603 \pm 0.173}$
MAR	100	$\mathbf{1.010 \pm 0.032}$	1.013 ± 0.031	1.033 ± 0.050	1.031 ± 0.045
	500	0.928 ± 0.049	$\mathbf{0.912 \pm 0.056}$	0.913 ± 0.062	0.913 ± 0.062
	1000	0.863 ± 0.089	0.852 ± 0.076	0.848 ± 0.067	$\mathbf{0.845 \pm 0.070}$
	5000	0.680 ± 0.133	0.695 ± 0.133	0.665 ± 0.141	$\mathbf{0.665 \pm 0.140}$
	10000	0.661 ± 0.144	0.656 ± 0.154	$\mathbf{0.601 \pm 0.157}$	0.601 ± 0.158
MNAR	100	1.011 ± 0.035	$\mathbf{1.003 \pm 0.037}$	1.053 ± 0.064	1.011 ± 0.041
	500	0.928 ± 0.044	0.910 ± 0.052	0.948 ± 0.052	$\mathbf{0.908 \pm 0.053}$
	1000	0.879 ± 0.064	0.862 ± 0.073	0.898 ± 0.062	$\mathbf{0.851 \pm 0.079}$
	5000	0.704 ± 0.129	0.709 ± 0.119	0.720 ± 0.134	$\mathbf{0.698 \pm 0.127}$
	10000	0.668 ± 0.142	0.657 ± 0.135	0.643 ± 0.144	$\mathbf{0.633 \pm 0.151}$

Table 5.7: Mean and standard deviation of re-scaled SHD scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for dense networks, under the different assumptions of missingness and sample sizes.

all the five sample sizes. Tables 5.8 and 5.9 list the detailed results of the experiments. When the missingness is MCAR, the three HC-based variants achieve similar accuracy, as expected, and generally outperform the Structural EM algorithm when the sample size is larger than 500. When the missingness is MAR or MNAR, the performance of HC-aIPW improves over the other algorithms, especially when the sample size is larger than 500. These results are consistent with those obtained from the randomised sparse and dense networks presented in subsections 5.5.3 and 5.5.4 respectively.

5.6 Conclusion

Learning accurate BN structure from incomplete data remains a challenging task. Most BN structure learning algorithms do not support learning from incomplete data, and this is partly explained by the considerable increase in computational complexity when dealing with incomplete data. The increased computational complexity caused by missing data adds to a problem that is NP-hard even when data are complete. This challenge is even greater when missing values are systematic rather than random.

This Chapter investigated three novel HC-based variants that employ pairwise deletion and IPW strategies to deal with random and systematic missing data. The HC-pairwise and HC-IPW variants can be viewed as subversions of HC-aIPW, which is the most complete and best performing variant described in this Chapter. All of the three variants have been applied to different cases of data missingness, and their performance was compared to the state-of-the-art Structural EM algorithm that is available in the *bnlearn* R package. Moreover, all performances under missingness have been compared to HC when applied to the corresponding complete data sets. The empirical results show:

1. Pairing HC with pairwise deletion (i.e., the HC-pairwise variant) is enough to learn graphs that are more accurate, as well as less computationally expensive, compared to the graphs

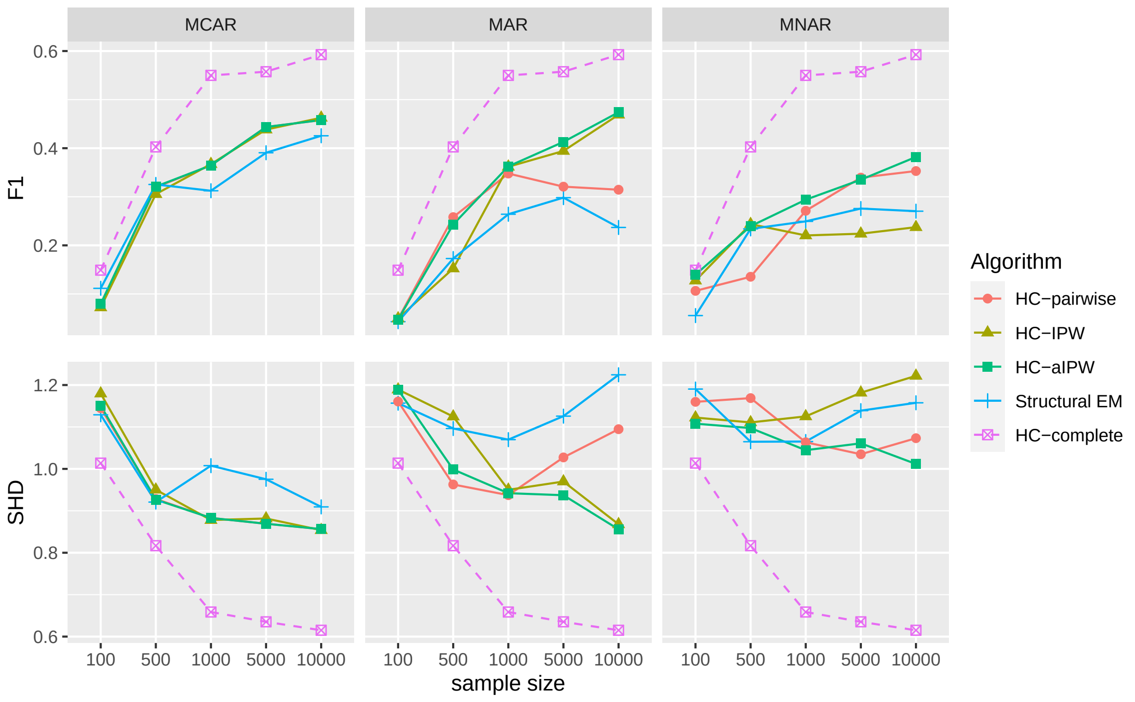


Figure 5.6: Average F_1 and re-scaled SHD scores learnt by HC-pairwise, HC-IPW, HC-aIPW and Structural EM for real-world networks, under different assumptions of missingness and sample sizes. Each score represents the average score over 50 CPDAGs. Note the scores of HC-complete are based on complete data for benchmarking purposes; i.e., the same scores are superimposed in all three missingness cases as a dashed line.

Data	Sample size	Structural EM	HC-pairwise	HC-IPW	HC-aIPW
	100	**0.112 ± 0.103**	0.078 ± 0.048	0.072 ± 0.046	0.079 ± 0.050
	500	**0.325 ± 0.268**	0.319 ± 0.273	0.305 ± 0.279	0.321 ± 0.272
MCAR	1000	0.312 ± 0.265	0.365 ± 0.220	**0.367 ± 0.216**	0.365 ± 0.220
	5000	0.391 ± 0.248	**0.444 ± 0.213**	0.438 ± 0.219	**0.444 ± 0.213**
	10000	0.426 ± 0.246	0.458 ± 0.196	**0.463 ± 0.192**	0.458 ± 0.196
	100	0.043 ± 0.037	0.048 ± 0.039	**0.050 ± 0.041**	0.047 ± 0.039
	500	0.173 ± 0.149	**0.258 ± 0.338**	0.152 ± 0.121	0.243 ± 0.342
MAR	1000	0.264 ± 0.303	0.348 ± 0.352	0.362 ± 0.313	**0.363 ± 0.311**
	5000	0.298 ± 0.340	0.321 ± 0.256	0.394 ± 0.266	**0.413 ± 0.261**
	10000	0.237 ± 0.326	0.315 ± 0.247	0.469 ± 0.349	**0.474 ± 0.343**
	100	0.055 ± 0.047	0.106 ± 0.095	0.127 ± 0.153	**0.140 ± 0.149**
	500	0.234 ± 0.240	0.135 ± 0.079	**0.244 ± 0.234**	0.239 ± 0.236
MNAR	1000	0.249 ± 0.271	0.271 ± 0.256	0.220 ± 0.120	**0.294 ± 0.243**
	5000	0.276 ± 0.253	**0.339 ± 0.236**	0.224 ± 0.107	0.335 ± 0.241
	10000	0.270 ± 0.217	0.353 ± 0.231	0.237 ± 0.105	**0.382 ± 0.242**

Table 5.8: Mean and standard deviation of F_1 scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for real-world networks, under the different assumptions of missingness and sample sizes.

Data	Sample size	Structural EM	HC-pairwise	HC-IPW	HC-aIPW
MCAR	100	$\mathbf{1.129 \pm 0.098}$	1.145 ± 0.137	1.180 ± 0.160	1.151 ± 0.136
	500	$\mathbf{0.921 \pm 0.287}$	0.928 ± 0.284	0.950 ± 0.303	0.927 ± 0.284
	1000	1.008 ± 0.434	0.883 ± 0.303	$\mathbf{0.878 \pm 0.298}$	0.883 ± 0.303
	5000	0.975 ± 0.437	$\mathbf{0.869 \pm 0.310}$	0.882 ± 0.321	$\mathbf{0.869 \pm 0.310}$
	10000	0.909 ± 0.368	0.856 ± 0.296	$\mathbf{0.854 \pm 0.291}$	0.856 ± 0.296
MAR	100	$\mathbf{1.157 \pm 0.137}$	1.161 ± 0.150	1.189 ± 0.129	1.189 ± 0.128
	500	1.096 ± 0.217	$\mathbf{0.963 \pm 0.438}$	1.125 ± 0.189	0.999 ± 0.451
	1000	1.070 ± 0.443	$\mathbf{0.937 \pm 0.472}$	0.950 ± 0.415	0.942 ± 0.411
	5000	1.126 ± 0.597	1.027 ± 0.453	0.970 ± 0.444	$\mathbf{0.937 \pm 0.436}$
	10000	1.224 ± 0.551	1.095 ± 0.466	0.868 ± 0.609	$\mathbf{0.856 \pm 0.599}$
MNAR	100	1.190 ± 0.115	1.160 ± 0.155	1.122 ± 0.229	$\mathbf{1.108 \pm 0.214}$
	500	$\mathbf{1.065 \pm 0.313}$	1.169 ± 0.154	1.111 ± 0.288	1.097 ± 0.277
	1000	1.065 ± 0.385	1.064 ± 0.356	1.125 ± 0.179	$\mathbf{1.045 \pm 0.344}$
	5000	1.139 ± 0.424	$\mathbf{1.035 \pm 0.389}$	1.182 ± 0.206	1.061 ± 0.379
	10000	1.158 ± 0.359	1.073 ± 0.394	1.222 ± 0.218	$\mathbf{1.012 \pm 0.413}$

Table 5.9: Mean and standard deviation of re-scaled SHD scores produced by Structural EM, HC-pairwise, HC-IPW and HC-aIPW for real-world networks, under the different assumptions of missingness and sample sizes.

produced by the Structural EM algorithm.

2. Combining HC with both pairwise deletion and IPW techniques (i.e., the HC-IPW variant) further improves learning accuracy under MCAR and MAR, in general, but decreases accuracy under MNAR due to aggressive pruning employed by HC-IPW on the data cases (refer to subsection 5.4.3). Moreover, HC-IPW becomes considerably slower than HC-pairwise, although it remains an order of magnitude faster than Structural EM.

3. The HC-aIPW takes advantage of both strategies, as in HC-IPW, but relaxes the pruning strategy on the data cases and returns the overall best performance, especially under MNAR which represents the most difficult case of missingness.

4. All three HC variants described in this chapter outperform Structural EM in most cases. Importantly, the performance of HC-aIPW on missing data approaches the performance of HC on complete data when sample size is 10,000 and the ground truth graph is sparse, and this observation is consistent under all three cases of missingness.

A possible future research direction is to investigate the application of these learning strategies to search algorithms that are more complex than HC, such as Tabu, or other variants of HC such as the GES algorithm [Chickering, 2002] which explores the CPDAG, rather than DAG, space. Another possible research direction would be to combine the IPW method with the NAL score [Balov et al., 2013], which is a scoring function intended for missingness under MCAR, and further investigate the possibility of a new decomposable scoring function under systematic missingness cases of MAR and MNAR.

Chapter 6

Using Markov blanket to improve data imputation in the presence of systematic missingness

6.1 Introduction and relevant works

Generally, the problem of missingness is typically handled by imputation approaches which estimate the missing values, often using regression or generative models, to obtain a complete data set. As a result, a general method to deal with incomplete data for structure learning is to execute structure learning algorithms on the imputed data. This approach is straightforward to implement but highly dependent on the quality of the imputed data. Therefore, this chapter investigates ways to improve imputation of systematic missingness.

The imputation algorithms are often classified as either statistical or machine learning methods [Lin and Tsai, 2020]. Statistical imputation methods include Mean/Mode, which is one of the simplest methods where the imputation is derived by the mean or mode of the observed values found in the same data column. Mean and Mode yield valid statistical inferences only when missingness is MCAR. It has been shown that the missingness mechanism is *ignorable* if i) the data is MAR and ii) the parameters that govern the data generation model and missingness mechanism are distinct [Little and Rubin, 2019]. In practice, condition (ii) for ignorability is almost always true. Therefore, MAR and ignorability are usually viewed as equivalent [Allison, 2009]. Besides, if the missingness mechanism is ignorable, we could estimate the model of interest by ignoring the missingness mechanism and retrieving the likelihood of the observed part of data [Rabe-Hesketh and Skrondal, 2023]. Therefore, in the context of data imputation, the property of ignorability is critical because it implies the data generation model is *identifiable*, i.e., the joint distribution of the model can be uniquely determined by the observed distribution, so that we can use the model trained on the observed data to impute the missing values. In the discussion that follows, we will introduce methods that assume MAR.

A more advanced statistical method is the EM algorithm [Honaker et al., 2011]. EM computes the expectation of sufficient statistics given the observed data at the E-step (Expectation), and then maximises likelihood at the M-step (Maximisation). It iterates over these two steps until convergence, at which point the converged parameters are used along with the observed data to impute missing values. Another statistical algorithm is the one proposed by Hastie et al. [2015], called softImpute, which treats imputation as a matrix completion problem and solves it by finding a rank-restricted singular value decomposition. Multiple imputation is another popular statistical method for handling missing data, and considers the uncertainty of missing values. Some classic multiple-imputation algorithms include the Multivariate Normal Imputation (MVNI) [Lee and Carlin, 2010], Multiple Imputation by Chained Equations (MICE) [Van Buuren and Groothuis-Oudshoorn, 2011], and Extreme Learning Machine (ELM) [Sovilj et al., 2016].

On the other hand, one of the earliest imputation methods that come from the Machine Learning (ML) field include the k-Nearest Neighbour (k-NN) [Zhang, 2012], which imputes empty cells according to their k-nearest observed data points. A well-established ML imputation algorithm is MissForest [Stekhoven and Bühlmann, 2012], which trains a Random Forest (RF) regression model recursively given the observed data, for every variable containing missing values, and uses the trained RF model to impute missing values. Recently, deep generative networks have also been used for imputing missing data values. Yoon et al. [2018] proposed the Generative Adversarial Imputation Nets (GAIN) algorithm which trains the generator to impute missing data and the discriminator to distinguish original data and imputed data, and was shown to have higher imputation accuracy compared to previous approaches. Other ML techniques used for imputation include the optimal transport [Muzellec et al., 2020], a neural network with causal regulariser [Kyono et al., 2021], and automatic model selection [Jarrett et al., 2022].

All of the aforementioned algorithms assume that all the variables in the data correlate with each other, and use all the variables to impute the missing values. Considering all of the data variables increases the risk of over-fitting, but which can be minimised through L1 and L2 regularisation methods often employed by ML algorithms. However, regularisation leads to models that tend to lack interpretability and theoretical guarantees of correctness. Because this Chapter focuses on interpretable models, such as those produced by structure learning algorithms, we shall focus on causal feature selection which maintains interpretability, rather than regularisation. This is also partly motivated by Dzulkalnine and Sallehuddin [2019] who showed that using uncorrelated variables to impute missing values not only decreases learning efficiency, but also degrades imputation accuracy. On this basis, it has recently been suggested to include a feature selection phase that prunes off potentially unrelated variables, for each variable containing missing values, prior to imputation [Bu et al., 2016, Liu et al., 2020, Hieu Nguyen et al., 2021].

Relevant studies that focus on feature selection for imputation include the work by Doquire and Verleysen [2012] who used Mutual Information (MI) to measure the dependency between variables. They used a greedy forward search procedure to construct the feature subset, which is an iterative process that constructs feature sets that maximise MI with the dependent variable. Sefidian and Daneshpour [2019] also estimate the dependency between variables using MI, and chose to select a set of variables that increase MI above a given threshold, as the features of a given dependent variable. On the other hand, the algorithm proposed by Dzulkalnine and Sallehuddin [2019] applies a fuzzy Principle Component Analysis (PCA) approach to the complete data cases to remove irrelevant variables from the feature set, followed by a SVM classification feature selection task that returns the set of features that maximise accuracy on the dependent variable. Lastly, evolutionary optimisation algorithms have also been adopted for feature selection in imputation, and include differential evolution [Tran et al., 2018], genetic algorithms [Awawdeh et al., 2022], and particle swarm optimisation [Jin et al., 2022].

Recently, causal information has also been adopted to feature selection for missing data imputation. Kyono et al. [2021] proposed to impute missing values of a variable given its causal parents derived from the weights of the input layer in the neural network. Similarly, Yu et al. [2022] proposed the MimMB framework that learns MBs to be used for feature selection in imputation, which is an iterative process that learns MBs from the imputed data and updates the learnt MB after each iteration. Note that while MimMB is related to our work, since we also use MB construction for feature selection, an important distinction between the two is that MimMB combines MBs with imputed data whereas, as we later describe in Section 6.3, the learning phase of MBs that we propose is separated from imputation, accounts for partially observed variables and improves computational efficiency.

In this Chapter, we use the graphical expression of missingness proposed by Mohan et al. [2013], as we did in Chapter 4, known as m-graph to capture the relationship between missingness and observed variables. We first show that an original version of the GS algorithm is capable of discovering the MBs in m-graphs containing partially observed variables, when applied to test-wise deleted data. Because this approach relies on CI tests with large conditioning sets, we modify GS such that the number of conditioning sets considered for CI tests is reduced. We provide proof that the modified GS is capable of discovering the MBs of partially observed variables in m-graphs, under the same assumptions as with the original GS. We then propose a new imputation algorithm, which we call Markov Blanket MissForest (MBMF), that combines the modified GS with the state-of-the-art MissForest (MF) imputation algorithm. The Chapter is organised as follows: Section 6.2 provides necessary preliminary information that includes notation and background information, Section 6.3 describes the proposed algorithm, Section 6.4 presents the experimental results, and we provide our concluding remarks in Section 6.5.

6.2 Preliminaries

Consistent with Chapter 5, we denote the set of fully observed variables as $\boldsymbol{V}^o$ and the set of partially observed variables as $\boldsymbol{V}^m$. For every partially observed variable $V_i \in \boldsymbol{V}^m$, we define an auxiliary variable R_i to reflect the missingness in V_i, where R_i takes the value of 0 when V_i is recorded and the value of 1 when V_i is missing. We also adopt the Markov assumption, the faithfulness assumption and the causal sufficiency assumption, as well as the Assumption 5.3.1 and Assumption 5.3.2 that relate to m-graphs.

Given the faithfulness assumption, a variable is conditionally independent of all the other variables given its MB, which contains all its parents, children and parents of its children. We denote MB of a variable V_i as $MB(V_i)$. We also define the *intrinsic MB* of a variable V_i as the set of variables that are still in the MB of V_i after removing all indicator variables from $\mathcal{G}$. We denote the intrinsic MB of V_i by $iMB(V_i)$. Note that $iMB(V_i)$ is not necessarily equivalent to the set of observed variables in $MB(V_i)$. This is because the missing indicators might be a common effect of two observed variables. For example, the intrinsic MB of V_1 in Figure 6.1 is $\{V_2, V_3, V_4, V_5\}$, whereas the standard MB would have also included V_6 and R_3.

6.3 Markov Blanket based feature selection for imputation

Given the description of the m-graph and causal faithfulness assumption, the problem of feature selection under incomplete data can be converted into a MB discovery problem over m-graphs that

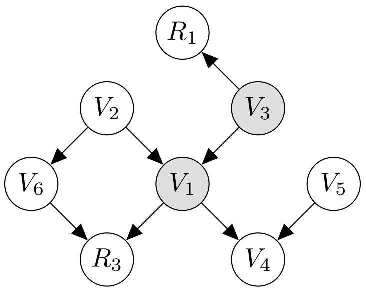


Figure 6.1: An example of m-graph. Shaded nodes represent partially observed variables.

contain partially observed variables and missing indicators[1]. Specifically, the task in this chapter is to find the MB for each partially observed variable $V_i \in \boldsymbol{V}^m$ from the m-graph that is assumed to be faithful to the underlying probability distribution $p(\boldsymbol{V}^o, \boldsymbol{V}^m, \boldsymbol{R})$, given Assumptions 5.3.1 and 5.3.2[2]. Because of the possible causal links between partially observed variables $\boldsymbol{V}^m$ and indicator variables $\boldsymbol{R}$ (i.e., in the case of MNAR), the MB of a partially observed variable is likely to contain both observed and indicator variables. For example, in Figure 6.1, the MB of V_1 is $\{V_2, V_3, V_4, V_5, V_6, R_3\}$.

Next, we show that the GS algorithm with test-wise deletion is capable of discovering the m-graph MB of any variable from incomplete data. Here, unlike list-wise deletion which removes all data rows containing at least one missing value, we use test-wise deletion which removes data cases containing missing values in any of the variables involved in the current CI test. The pseudo-code of GS with test-wise deletion is presented in Algorithm 13. Note that we slightly modify the Grow phase, i.e., line 5, of GS to eliminate its dependency on the order of the variables in the data [Kitson and Constantinou, 2022].

Given the faithfulness assumption, Assumption 5.3.1 and Assumption 5.3.2, Proposition 6.3.1 describes the modified GS with test-wise deletion.

Proposition 6.3.1. *Given the faithfulness assumption, Assumption 5.3.1 and Assumption 5.3.2, for any observed variable V_i in a m-graph $\mathcal{G}$, the output of GS(V_i, $\boldsymbol{V}^o \cup \boldsymbol{V}^m \cup \boldsymbol{R} \setminus \{V_i, R_i\}, D$) is the $MB(V_i)$ in $\mathcal{G}$.*

Proof Given Assumption 5.3.1 and Assumption 5.3.2, R_i is not a child of V_i nor a parent of any other variables. Therefore, we do not need to consider R_i when learning $MB(V_i)$.

We firstly show that the output Candidate Markov Blanket (*CMB*; i.e., the output of Algorithm 1) set learnt at the *Grow* phase contains all the variables of $MB(V_i)$. Assume Y is a parent or a child of V_i, given the faithfulness condition, $V_i \not\perp\!\!\!\perp Y \mid CMB, \boldsymbol{R}_{\{V_i,Y\}\cup CMB} = \boldsymbol{0}$ for any $CMB \subseteq \boldsymbol{V}^o \cup \boldsymbol{V}^m \cup \boldsymbol{R} \setminus \{V_i, R_i\}$. Therefore, Y will always be added to *CMB* during the *Grow* phase. Assume Z is a parent of Y and Y is a child of V_i. Once Y is added to *CMB*, V_i and Z will not be d-separated by $CMB \cup \boldsymbol{R}_{\{V_i,Y\}\cup CMB}$ and thus, Z will also be added to *CMB*.

Lastly, we show that the *Shrink* phase preserves the variables in $MB(V_i)$ only. Let us assume that T is the first variable in $MB(V_i)$ when the algorithm enters the *Shrink* phase and attempts to remove variables in *CMB*, which have already been added during the *Grow* phase. Because

[1]To impute the missing values of an incomplete variable, we consider its MB, rather than only its parent variables, for two reasons. Firstly, the parents of an incomplete (or even a complete) variable are not guaranteed to be identifiable from observational data. Secondly, the MB contains the set of nodes that can make the given variable independent over all other variables present in the input data.

[2]While the Assumptions 5.3.1 and 5.3.2 restrict the possible graphs that explain the missingness mechanism $p(\boldsymbol{R} \mid \boldsymbol{V}^o \cup \boldsymbol{V}^m)$, the true graph remains unknown and needs to be inferred from data by the algorithm.

Algorithm 13 The Grow and Shrink (GS) algorithm with test-wise deletion

```
1: procedure GS($X, \boldsymbol{S}, D$)
       Input: target variable $X$, candidate variables set $\boldsymbol{S}$, data $D$
       Output: Candidate Markov Blanket $CMB$ of $X$
2:     $CMB \leftarrow \varnothing$
       ▷ Grow Phase
3:     repeat
4:         if $\exists S_i \in \boldsymbol{S}, s.t. X \not\perp\!\!\!\perp S_i \mid CMB, \boldsymbol{R}_{\{X,S_i\}\cup CMB} = \boldsymbol{0}$ then
5:             add $S_i$ with the lowest p-value to $CMB$
6:             remove $S_i$ from $\boldsymbol{S}$
7:         end if
8:     until $CMB$ stays unchanged
       ▷ Shrink Phase
9:     for each $Y \in CMB$ do
10:        if $X \perp\!\!\!\perp Y \mid CMB \setminus \{Y\}, \boldsymbol{R}_{\{X\}\cup CMB} = \boldsymbol{0}$ then
11:            remove $Y$ from $CMB$
12:        end if
13:    end for
14:    return $CMB$
15: end procedure
```

$MB(V_i) \setminus \{T\} \subseteq CMB$, irrespective of T being a neighbour or a parent of a child of V_i, we always have $V_i \not\perp\!\!\!\perp T \mid CMB \setminus \{T\}, \boldsymbol{R}_{\{V_i\}\cup CMB} = \boldsymbol{0}$ given the faithfulness condition. Therefore, no variable in $MB(V_i)$ will be removed during the *Shrink* phase. On the other hand, if we assume $T \notin MB(V_i)$, since $MB(V_i) \subseteq CMB \cup \boldsymbol{R}_{\{V_i\}\cup CMB} \setminus \{T\}$, we have $V_i \perp\!\!\!\perp T \mid CMB \setminus \{T\}, \boldsymbol{R}_{\{V_i\}\cup CMB} = \boldsymbol{0}$ given the faithfulness condition and thus, T will be removed from *CMB*. ■

Therefore, an intuitive way to determine the relevant features for a given variable is to apply the function GS(V_i, $\boldsymbol{V}^o \cup \boldsymbol{V}^m \cup \boldsymbol{R} \setminus \{V_i, R_i\}, D$) on every $V_i \in \boldsymbol{V}^m$. However, this is impractical since the maximum size of the conditioning sets used for CI testing is $|\boldsymbol{V}^o| + 2|\boldsymbol{V}^m| - 3$. In practice, the accuracy of CI tests drops dramatically as we increase the size of the conditional set [Tsamardinos et al., 2003b]. To address this, we propose the Markov Blanket Feature Selection (MBFS, Algorithm 14) that aims to restrict the maximum size of the conditional set used by CI tests to $|\boldsymbol{V}^o| + |\boldsymbol{V}^m| - 1$.

MBFS involves two phases, where the first phase involves learning the *intrinsic MB* of each partially observed variable. Given a m-graph $\mathcal{G}$, we define the *intrinsic MB* of a variable V_i as the set of variables that are still in the MB of V_i after removing all indicator variables from $\mathcal{G}$. We denote the intrinsic MB of V_i by $iMB(V_i)$. Note that $iMB(V_i)$ is not necessarily equivalent to the set of observed variables in $MB(V_i)$. This is because the missing indicators might be a common effect of two observed variables. For example, the intrinsic MB of V_1 in Figure 6.1 is $\{V_2, V_3, V_4, V_5\}$, whereas the standard MB would have also included V_6 and R_3. It is worth noting that, during phase 1, some nodes that do not belong in $iMB(V_i)$ may still be included in the output *CMB*. However, as we show in the proof of Proposition 6.3.2, these nodes are still in the $MB(V_i)$ in m-graph. The phase 2 aims to learn all the parents of the missing indicators, in order to complete the feature set of $MB(V_i)$. Proposition 6.3.2 states that MBFS is capable of learning $MB(V_i)$ from missing data for any $V_i \in \boldsymbol{V}^m$ in a m-graph and thus, it could serve as an effective feature selection approach for imputation algorithms.

Proposition 6.3.2. *Given the faithfulness condition, Assumption 5.3.1 and Assumption 5.3.2,*

Algorithm 14 The Markov Blanket-based Feature Selection (MBFS) algorithm

```
1: procedure MBFS($V_i, D$)
        Input: partially observed variable $V_i$, data $D$
        Output: candidate Markov Blanket $CMB$ of $V_i$
      ▷ Phase 1 (discover intrinsic MB)
2:    $CMB \leftarrow \mathrm{GS}\left(V_i, \boldsymbol{V}^o \cup \boldsymbol{V}^m \backslash \{V_i\}, D\right)$
      ▷ Phase 2 (discover other variables in MB caused by indicators)
3:    for each $R_j \in \boldsymbol{R} \backslash \{R_i\}$ do
4:        $CPS \leftarrow \boldsymbol{V}^o \cup \boldsymbol{V}^m \backslash \{V_j\}$
5:        for each $V_k \in CPS$ do
6:            if $R_j \perp\!\!\!\perp V_k \mid \boldsymbol{S}, \boldsymbol{R}_{\{V_k\} \cup \boldsymbol{S}} = \mathbf{0}$ for any $\boldsymbol{S} \subseteq CPS$ then
7:                remove $V_k$ from CPS
8:            end if
9:        end for
10:       if $V_i \in CPS$ then
11:           $CMB \leftarrow CMB \cup \{R_j\} \cup CPS$
12:       end if
13:   end for
14:   return $CMB$
15: end procedure
```

for any observed variable V_i in a m-graph $\mathcal{G}$, MBFS(V_i, D) returns MB(V_i) in $\mathcal{G}$.

Proof For convenience, we denote path p as the path being *blocked* by a variable set $\boldsymbol{S}$ if there is at least one node on p that satisfies either 1) it is a non-collider and in $\boldsymbol{S}$ or 2) it is a collider and neither it nor any of its descendants are in $\boldsymbol{S}$. Thus, if all paths between V_i and V_j are blocked by $\boldsymbol{S}$, V_i and V_j are d-separated by $\boldsymbol{S}$.

We first show that the CMB returned by $\mathrm{GS}\left(V_i, \boldsymbol{V}^o \cup \boldsymbol{V}^m \backslash \{V_i\}, D\right)$ (phase 1 of MBFS) contains all but only variables in $iMB(V_i)$, in addition to some other observed variables belonging to $MB(V_i)$. For the Grow phase in $\mathrm{GS}\left(V_i, \boldsymbol{V}^o \cup \boldsymbol{V}^m \backslash \{V_i\}, D\right)$, assume that Y is a parent or a child of V_i, and that given the faithfulness condition, $V_i \not\perp\!\!\!\perp Y \mid CMB, \boldsymbol{R}_{\{V_i,Y\} \cup CMB} = \mathbf{0}$ for any $CMB \subseteq \boldsymbol{V}^o \cup \boldsymbol{V}^m \backslash \{V_i\}$. Therefore, Y will be added in CMB. Assume that Z is a parent of an observed variable Y, which is a child of V_i. After Y is added to CMB, V_i and Z can no longer be d-separated by $CMB \cup \boldsymbol{R}_{\{V_i,Y\} \cup CMB}$. Thus, Z will also be added to CMB eventually. As a result, all variables in $iMB(V_i)$ will be included in CMB at the end of the Grow phase.

For the Shrink phase, we prove two things: 1) that all the variables in $iMB(V_i)$ will remain in CMB, and 2) that all the variables not in the $MB(V_i)$ will be removed from CMB.

1. Suppose that T is the first variable in $iMB(V_i)$ that the algorithm attempts to remove from CMB, and that V_i and T are connected by either a direct edge or a path through a collider in $iMB(V_i)$. Therefore, T and V_i cannot be d-separated by $CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \backslash \{T\}$, since $iMB(V_i) \subseteq CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \backslash \{T\}$. Then, we have $V_i \not\perp\!\!\!\perp T \mid CMB \backslash \{T\}, \boldsymbol{R}_{\{V_i\} \cup CMB} = \mathbf{0}$ given the faithfulness condition.

2. Suppose that T is an observed variable in CMB such that $T \notin MB(V_i)$. Given the faithfulness condition, all we need to prove is that all paths between V_i and T are blocked by $CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \backslash \{T\}$.

    - For a path p between V_i and T composed by observed variables only, there is at least one non-collider node on p that belongs to $iMB(V_i)$, such that p is blocked by $CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \backslash \{T\}$, since $iMB(V_i) \subseteq CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \backslash \{T\}$.

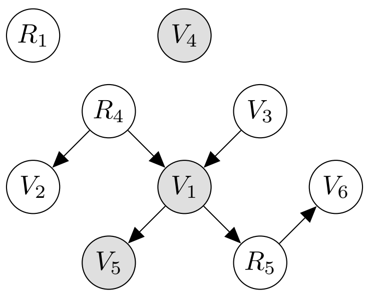


Figure 6.2: A hypothetical m-graph used to described the implications of violating Assumption 5.3.1 of MBFS. Shaded nodes represent partially observed variables.

- For a path p between V_i and T contains indicator variables, when an observed node V_j is adjacent to V_i on p, at least one non-collider (either V_j or the parent of V_j) on p will also be in $iMB(V_i)$. This is because no indicator variable can be a parent of V_j according to Assumption 5.3.1, which implies that p will be blocked by $iMB(V_i)$ as well as $CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \setminus \{T\}$. If the adjacent node of V_i on p is an indicator variable R_j, then according to Assumption 5.3.1 and Assumption 5.3.2, $R_j \neq R_i$ and it must have another parent $V_k \notin \{T, V_j\}$ on p since $T \notin MB(V_i)$. If $R_j \notin \boldsymbol{R}_{CMB}$, path p is blocked by $CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \setminus \{T\}$. If $R_j \in \boldsymbol{R}_{CMB}$, V_k must be included in CMB during the Grow phase, and cannot be removed during the Shrink phase since V_i and V_k cannot be d-separated by $CMB - \{V_k\} \cup \boldsymbol{R}_{\{V_i\} \cup CMB}$. Therefore, p is still blocked by $CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \setminus \{T\}$ since V_k is a non-collider on p in $CMB \cup \boldsymbol{R}_{\{V_i\} \cup CMB} \setminus \{T\}$.

In phase 2 of MBFS, all the parents of each R_j should remain in CPS as they cannot be d-separated from R_j given any set composed of observed variables. Besides, all of the other variables should be removed from CPS as they can be d-separated from R_j given any set that contains all the parents of R_j. As a result, CMB should return $MB(V_i)$ at the end of phase 2. ■

If Assumption 5.3.1 is violated (e.g., when an indicator variable is also a parent of some other variable), then both phases of MBFS may produce redundant variables in the CMB set. Suppose that the true m-graph is the one shown in Figure 6.2. If we use MBFS to learn the MB of V_1, then V_2 will be included in the CMB set of Phase 1, and this is because V_1 and V_2 are independent only when conditioning on R_4, whereas V_4 and V_1 are always independent. This implies that MBFS will never test V_1 and V_2 for CI conditional on R_4 during Phase 1. In Phase 2, V_6 will be included in the CPS set of R_5 and thus, it will also be added in CMB of V_1.

If Assumption 5.3.2 is violated (e.g., when a partially observed variable V_i is also the parent of its own missingness indicator R_i), this will cause MBFS not to include R_i in the CMB of V_i, since MBFS would not consider V_i as a candidate variable in the CPS of R_i during Phase 2.

We then propose a modified version of MissForest that incorporates MBFS as a feature selection process. The modified version of MissForest, which we call Markov Blanket MissForest (MBMF), takes the feature set $MBFS(V_i, D)$ for each partially observed variable V_i, as opposed to considering all of the other observed variables as the explanatory features of V_i in the Random Forest regression model used in MissForest. In other words, MBMF accounts for the possible causal relationships between partially observed variables and the missing indicators, to minimise the risk of considering irrelevant observed variables for imputation by MissForest.

	Number of variables	Number of edges	Data type
ECOLI70	46	70	Continuous
MAGIC-IRRI	64	102	Continuous
ARTH150	107	150	Continuous

Table 6.1: Summary of the real-world BNs used in Subsection 6.4.1.

6.4 Experiments

We evaluate the proposed MBMF algorithm with reference to the standard version of MissForest (MF), the commonly used imputation algorithms Mean and Mode, the K-Nearest Neighbour (KNN), and two state-of-the-art algortihms; the softImpute and GAIN algorithms. While the evaluation includes experiments on both continuous and categorical data, some of the other algorithms can only process one of the two types of input data and hence, their application is restricted to continuous data (Mean and GAIN) or categorical data (Mode). We use the scikit-learn python package [Pedregosa et al., 2011] to test the Mean, Mode and KNN algorithms, the MissForest R package [Stekhoven and Stekhoven, 2013] to test MF, the softImpute R package [Hastie et al., 2015] to test SoftImpute, and the publicly available source code of GAIN. The implementation of MBMF, described in this paper, is available at: https://github.com/Enderlogic/Markov-Blanket-based-Feature-Selection.

MBMF is applied to continuous data using the Pearson's correlation test for CI tests, and to categorical data using the G-test statistic, both of which are the default choices for GS. We also consider the default threshold for independence, which is 0.1 for CI p-value tests. The other algorithms are also tested with their default hyper-parameters as implemented in their corresponding packages discussed above.

6.4.1 Synthetic case studies based on real-world BNs

We first apply the algorithms to synthetic data sampled from three continuous BNs, ECOLI70, MAGIC-IRRI and ARTH150, taken from the bnlearn repository [Scutari et al., 2010]. Details about these graphical networks can be found in Table 6.1.

We generate complete data sets for each network with sample sizes 500, 1000, 2000 and 3000. Then, for each complete data set, we create nine incomplete data sets composed of different combinations of missingness rates (i.e., 10%, 30% and 50%) and missingness assumptions (i.e., MCAR, MAR and MNAR). We randomly choose 50% of the variables to be made partially observed. We use α to represent the rate of missingness for each of those variables. This process differs depending on the underlying assumption of missingness. Specifically,

1. For the MCAR condition, a value in V_i is removed with probability $P(R_i = 1) = \alpha$.
2. For the MAR condition, a fully observed variable $V_j \in \boldsymbol{V}^o$ is randomly assigned as the parent of R_i. For continuous data, we denote the highest 10th-quantile of V_j as q_j, and remove values in V_i with the following conditional probabilities: $P(R_i = 1 \mid V_j > q_j) = 2\alpha$, $P(R_i = 1 \mid V_j <= q_j) = \frac{8}{9}\alpha$. For categorical data, we select a set of states $\boldsymbol{v}_j$ of V_j such that $\alpha < P(V_j = \boldsymbol{v}_j) < 1$. Then we remove values in V_i with the following conditional probabilities: $P(R_i = 1 \mid V_j \neq \boldsymbol{v}_j) = 2\alpha$, $P(R_i = 1 \mid V_j = \boldsymbol{v}_j) = 2\alpha - \frac{\alpha}{P(V_j = \boldsymbol{v}_j)}$.

3. For the MNAR condition, a partially observed variable $V_j \in \boldsymbol{V}^m \setminus \{V_i\}$ is randomly assigned as the parent of R_i. Variable values are then removed for each V_i using the same strategy as in the case of MAR.

6.4.2 Evaluation process

The imputation accuracy is evaluated using two different approaches. The first approach involves retrieving the Root Mean Squared Error (RMSE) between imputed data and complete data. Because RMSE is sensitive to the discrepancy between absolute data values, we normalise the complete data column-wise and re-scale the imputed data with the same normalisation parameters to eliminate bias.

The second approach involves assessing the impact of imputation on structural learning accuracy. We do this by comparing the CPDAGs learnt by the state-of-the-art GES causal structure learning algorithm [Chickering, 2002] from imputed data sets produced by the different imputation algorithms. The second approach is helpful because, while it is reasonable to assume that higher imputation accuracy helps causal machine learning, it is possible that some imputed values are more important than others. Causal structure learning represents a good approach to test this, and we use the F_1 score to measure the accuracy of graphical structures learnt by GES.

6.4.3 Results

Figure 6.3 depicts the average RMSE of imputed data produced by the different algorithms under different sample sizes. Note that the results we report on GAIN are, to some degree, inconsistent with the results presented in the original paper [Yoon et al., 2018], but are consistent with the results presented in follow-up studies [You et al., 2020, Nazabal et al., 2020, Kyono et al., 2021, Jarrett et al., 2022]. In general, the proposed MBMF algorithm is found to outperform the baseline MF under all scenarios of missingness. Specifically, for MCAR and MAR scenarios with missing rate 10%, MBMF provides a considerable improvement over MF, but this improvement diminishes with the higher missing rates of 30% and 50%. This is because a higher missing rate tends to decrease the sample size of the test-wise deleted data, which in turn reduces the accuracy of CI tests and the discovered *MB* set by MBFS. Importantly, MBMF outperforms MF considerably under all MNAR settings, which better reflect real-world missingness that is generally systematic. None of the other imputation algorithms provide satisfactory performance in terms of RMSE; at least relative to the MBMF and MF algorithms.

Figure 6.4 presents the average F_1 scores corresponding to the graphs learnt by GES, given the imputed data sets produced by the different imputation algorithms, and across the different sample sizes. While this evaluation approach decreases the discrepancy in performance scores between the top performing imputation algorithms, the results are consistent with those presented in Figure 6.3 since MBMF and MF are found to perform better than the other algorithms in almost all cases, and MBNF performs better than MF in most experiments. Specifically, MBMF and MF produce similar performance when the rate of missingness is lowest at 10%, with their performance being close to that produced with complete data (dashed line). These results serve as empirical evidence that both MFBF and MF imputation algorithms perform exceptionally well with relatively low rates of missingness. When the rate of missingness increases to 30%, MBMF performs better than MF in most cases. However, when the rate of missingness is highest, at 50%, there is no clear winner between MBMF and MF.

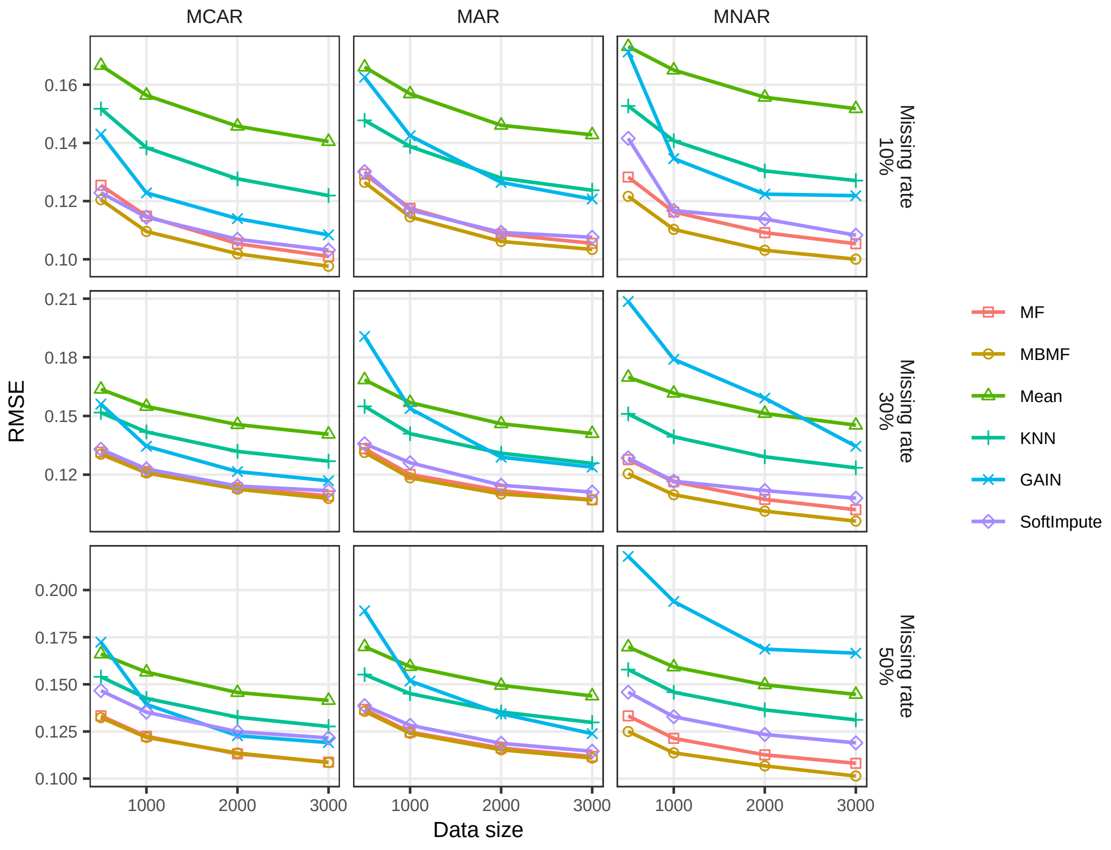


Figure 6.3: Average RMSE between complete and imputed data produced by the different algorithms. A Lower score represents better performance.

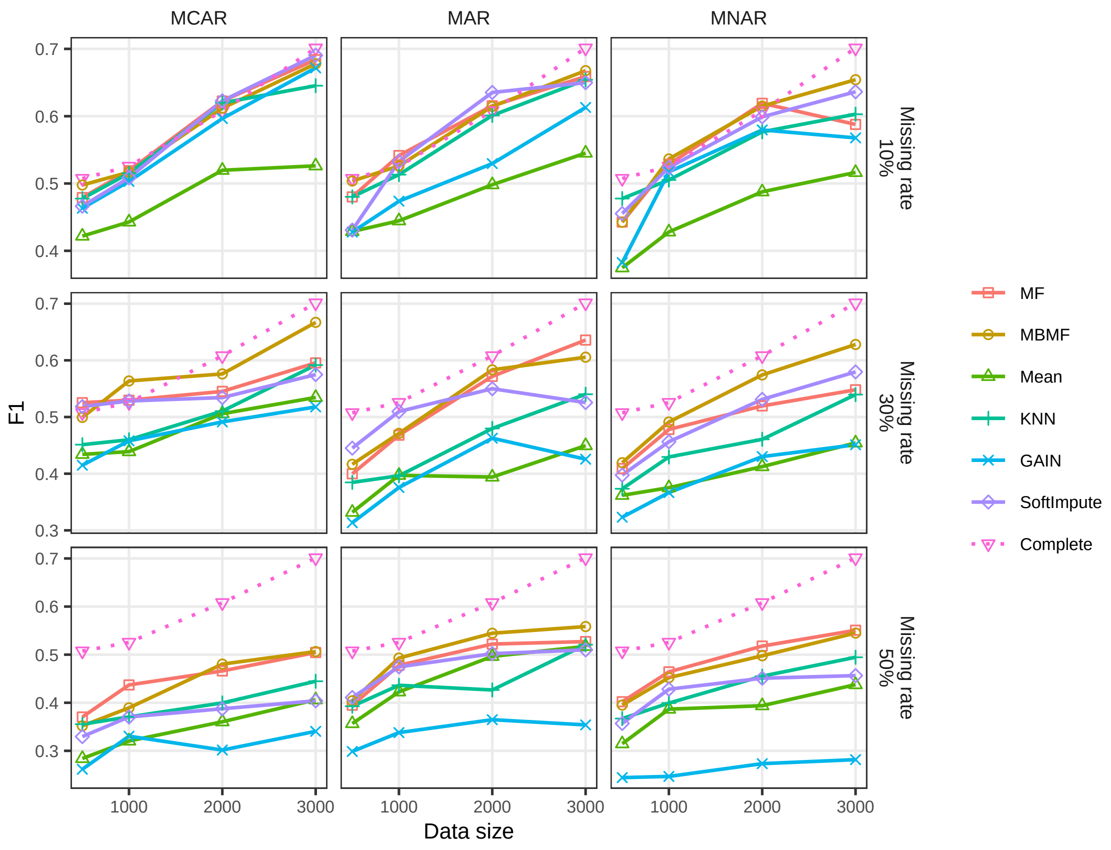


Figure 6.4: Average F_1 scores of the graphs learnt by GES from data imputed by the different algorithms. A higher F_1 score represents better performance. The dashed line represents the performance of GES when applied to complete data.

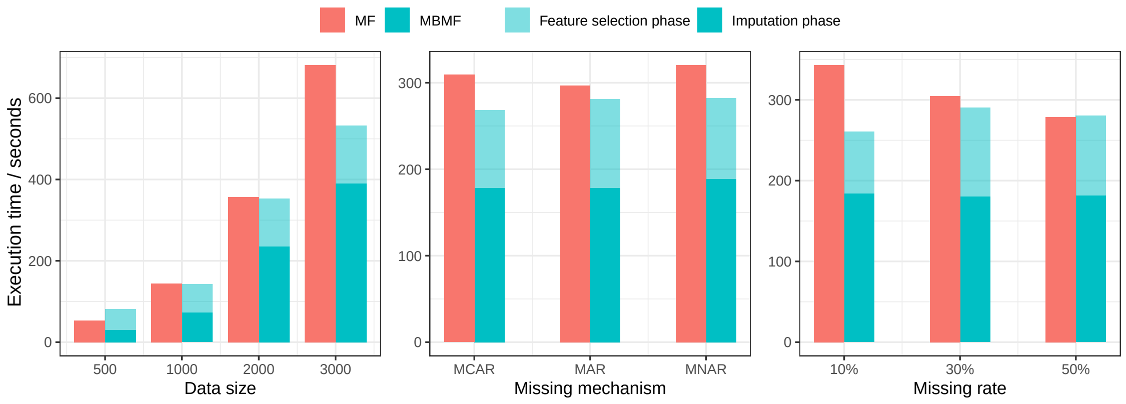


Figure 6.5: Average execution time of MF and MBMF under different sample sizes, mechanisms of missingness, and rates of missingness.

	Number of variables	Number of instances	Data type
Iris	4	150	Continuous
Breast	30	569	Continuous
Wine	11	1599	Continuous
Game	10	958	Categorical
Car	7	1728	Categorical
Mushroom	22	8124	Categorical

Table 6.2: Summary of the real-world data sets used in Subsection 6.4.4.

Lastly, we evaluate the computational efficiency of MBMF relative to the original MF. As shown in Figure 6.5, MBMF is generally more efficient than MF. Note that while MBMF involves an additional phase needed to perform feature selection, the additional time spent by MBMF in that phase is countered by the reduced time MBMF spends to actually impute values. This is because MBMF will almost always consider less features than MF during the imputation phase. Specifically, MBMF is slower than MF at the lowest sample size, but becomes increasingly faster than MF with increasing sample size. Averaging the results across the different mechanisms of missingness and missing rates also shows that MBMF is, in general, more efficient than MF. Note that MF is slightly more efficient when the rate of missingness is at its highest, 50%, since MF trains its RF regression model on observed data rows only; implying that a higher rate of missingness decreases the training data passed to the RF. The impact of the rate of missingness is higher on MF than MBMF since the number of independent features considered by MF is, in general, considerably higher than those considered by MBMF.

6.4.4 Real-world case study

We repeat the evaluation by applying the imputation algorithms to six real-world data sets retrieved from the UCI data repository [Dua and Graff, 2017]. A summary of these data sets is given in Table 6.2. We simulate missingness using the same strategy as described in Subsection 6.4.1. Specifically, for each complete real data set, we generate nine incomplete data sets composed of different rates and assumptions of missingness.

We use RMSE to evaluate the imputation accuracy when the algorithms are applied to continuous data, and the Proportion of Falsely Classified entries (PFC) when applied to categorical data.

	Mean/Mode	KNN	SoftImpute	GAIN	MF	MBMF
Iris	$.284 \pm .085$	$.103 \pm .029$	$.198 \pm .055$	$.164 \pm .060$	$.092 \pm .026$	$\mathbf{.092 \pm .025}$
Breast	$.149 \pm .012$	$.113 \pm .009$	$.140 \pm .040$	$.098 \pm .029$	$.057 \pm .009$	$\mathbf{.054 \pm .009}$
Wine	$.123 \pm .009$	$.109 \pm .013$	$.101 \pm .010$	$.106 \pm .009$	$\mathbf{.039 \pm .013}$	$.040 \pm .013$
Game	$.563 \pm .020$	$.501 \pm .072$	-	-	$.505 \pm .064$	$\mathbf{.496 \pm .070}$
Car	$.675 \pm .074$	$.613 \pm .073$	-	-	$.655 \pm .107$	$\mathbf{.593 \pm .061}$
Mushroom	$.443 \pm .062$	$.294 \pm .083$	-	-	$.207 \pm .060$	$\mathbf{.203 \pm .058}$

Table 6.3: Average RMSE and PFC scores, and their standard deviations, for different imputation algorithms and real-world data combinations. Lower RMSE and PFC scores represent better performance.

Note that the second evaluation approach, which involves investigating the impact of imputation accuracy on causal structure learning as described in subsection 6.4.2, is unsuitable here since we do not have a ground truth causal graph.

Table 6.3 presents the average scores and standard deviation. The results show that MBMF and MF continue to outperform the other algorithms, and that MBMF continues to outperform MF in most of the experiments. On this basis, we conclude that the results obtained from the real-world data sets are consistent with the results obtained from the synthetic data sets.

6.5 Conclusion

This Chapter described a novel feature selection algorithm, called MBFS, that recovers the Markov Blanket of partially observed variables based on the graphical expression of missingness known as the m-graph, which captures observed variables together with missingness indicators and the possible causal links between them. We incorporated MBFS into the imputation process of MissForest, to formulate a new algorithm suitable for imputation under both random and systematic missingness, which we call MBMF.

Empirical experiments based on both synthetic and real-world data sets show that MBMF outperforms the baseline MissForest in most of the experimental settings, and outperforms considerably other well-known or state-of-the-art imputation algorithms under both random and systematic missingness. Moreover, while MBMF incorporates an additional learning phase needed to perform feature selection for each partially observed variable found in the input data, the results show that MBMF is generally more efficient than the baseline MissForest, especially at larger sample sizes where efficiency matters the most. This is because the time saved during imputation due to prior feature selection is higher than the additional time spent determining the best features for each partially observed variable.

Because the feature selection phase can be independent of the imputation phase (i.e., by using the MBFS algorithm alone), future research works could extend this work to different relevant directions where feature selection is deemed to be important. For example, MBFS could be combined with other imputation algorithms, including those based on deep learning [Mattei and Frellsen, 2019, Fortuin et al., 2020, Lin et al., 2022] which are generally powerful but which tend to be time consuming and to overfit the data, since they typically process large numbers of uncorrelated features. Moreover, since each MB discovered by MBFS could be used to construct the complete m-graph of the input data set, a rather different possible direction for future research would be to investigate the capability of MBFS in recovering the entire m-graph of the input data. This task would require structural rules to deal with collisions between MBs as well as cycles, and would essentially convert MBFS into a structure learning algorithm.

Chapter 7

Conclusions and directions for future work

7.1 Conclusion

This thesis focuses on data noise and the negative repercussions it has on structure learning. We investigate new approaches and algorithms aimed at mitigating the impact of data noise on structure learning. The thesis focuses on two types of data noise that are commonly found in real data; namely measurement error and systematic missing data, but the investigations extent to other types of data noise.

Chapter 3 begins with an empirical evaluation that examines the impact of data noise on 15 structure learning algorithms that come from different classes of learning. This evaluation considers four types of data noise: missing values, measurement error (or incorrect values), merging states, and latent variables. The findings highlight the considerably impact of data noise on the accuracy of structure learning algorithms. Specifically, missing values and measurement error are found to strongly influence on the accuracy of structure learning, surpassing the impact of latent variables and merging states. Chapter 3 also introduces a novel structure learning algorithm called MAHC, which combines model averaging and pruning strategies with hill-climbing search, aimed at learning in the presence of data noise. The results demonstrate that MAHC performs competitively when the input data are clean, and better than most algorithms in the presence of multiple types of data noise. There results highlight the importance of model averaging over maximising the score of individual graphical structures, particularly when the input data are imperfect.

Chapter 4 introduces the SED algorithm which is a heuristic post-processing algorithm that can be directly applied to the learnt graph produced by other structure learning algorithms. The aim of SED is to discover and eliminate spurious (i.e., false positive) edges that are often produced in the presence of measurement error. The evaluation demonstrates that, in the absence of measurement error, SED generally preserves or slightly enhances the learnt graphs. When measurement error is present in the input data, however, SED considerably improves the learnt graphs, leading to enhanced accuracy and reliability.

In Chapter 5, a modified version of the HC algorithm, referred to as HC-aIPW, is presented to handle missing values in discrete data sets. HC-aIPW considers all three assumptions of missingness; namely MCAR, MAR and MNAR, and employs pairwise deletion to efficiently leverage the observed part of the input data, and an IPW approach to eliminate the potential bias caused by missing values when data are not MCAR. The empirical investigations show that HC-aIPW outperforms the commonly used and state-of-the-art Structural EM algorithm, both in terms of learning accuracy and efficiency, as well as both when data are missing at random and not at random.

Chapter 6 focuses on data imputation. The state-of-the-art assumes that all the variables in the data are relevant in imputing all missing values. In contrast, this Chapter describes a novel Markov blanket-based feature selection algorithm, called MBFS, that identifies the set of the variables that are relevant in imputing the missing values for each partially observed variable. Each set of related variables identified is then used as the set of independent variables in regression or generative models for each corresponding partially observed variable during the imputation process. MBFS is then combined with the MissForest imputation algorithm, to form the MBMF algorithm that performs imputation by restricting the set of variables considered for imputation to those identified as relevant by Markov blanket discovery. The results demonstrate that MBMF outperforms the state-of-the-art imputation algorithms both in terms of accuracy and efficiency, especially in the case where missing data are not missing at random.

7.2 Future research

Drawing upon the research conducted in this thesis, several future research directions emerge. Firstly, the approaches proposed in Chapter 4 and 5 focus on addressing a single type of data noise. However, in practice, real data are likely to contain multiple types of data noise. Approaches designed to account for multiple types of data noise are overlooked in the literature and thus, future work in this area would help in constructing structure learning algorithms that are more effective in practice.

Secondly, the sample size of the input data is also important in determining structure learning performance. Generative models have demonstrated their effectiveness in producing images and text. However, the potential of generative models remain underinvestigated in creating tabular data that may complement existing data, to address issues related with limited sample size. Xu et al. [2019] introduced a method that generates synthetic data to complement real data with high fidelity. However, the variables in the data set generated by their method tend to be correlated, rendering it unsuitable for structure learning. Developing generative models capable of capturing the independence relationships between variables could enable more effective structure learning in the presence of limited sample size.

Lastly, while synthetic experiments are important, they do not accurately reflect the effectiveness of these algorithms in practice. Therefore, it is imperative to adjust expectations of structure learning performance for real-world data sets. In [Constantinou et al., 2023] we investigate how 29 different structure learning algorithms behave when applied to COVID-19 data. We assessed the graphs produced by each structure learning algorithm independently, as well as in groups using model averaging. Specifically, we grouped algorithms in terms of their learning class or input data format, which includes discrete, continuous, and mixed data of the same information. The findings highlight the following open problems:

- The learnt graphs are highly sensitive to the selection of the algorithm and input data format combination. For example, graphs learnt with continuous data are found to be much denser than graphs learnt with discrete data, even for the same structure learning algorithms. The high inconsistency between the results suggests that continuous data may violate additional assumptions made by some algorithms, such as that continuous data follow linear Gaussian distributions. Considering score functions with weaker assumptions[Huang et al., 2018], or conditional independence tests [Zhang et al., 2011], might be more appropriate for real-world data. The high variability between algorithms also makes a case, in practice, for model averaging that would consider the results obtained over a set of algorithms, rather than focusing on the results obtained by a single algorithm.

- The different learnt graphical structures are found to have minor impact on predictive accuracy, and yet considerable differences arise when evaluating these graphs in terms of interventional or sensitivity analysis. These empirical findings further highlight the inability of predictive validation in providing meaningful answers to questions about causal reasoning.

- While model averaging is found to indeed reduce variability, we also find that the average graphs for each group of learnt structures (e.g., score-based vs constraint-based, or discrete vs continuous) are all very different from one another. Future studies could focus on developing more sophisticated approaches to model averaging. For instance, our results assume equal contribution from each structure learning algorithm in the average graph. However, it may be beneficial to assume a weighted average that prioritises edges learnt by algorithms known to be more accurate than others.

- The COVID-19 data set contains temporal dependencies that shift over time. For example, the effect of lockdown cannot be effectively measured between variables unless we account for its lag effect. We note that the traditional causal structure learning algorithms investigated in this thesis are oblivious to distribution shifts that may occur over time. Few studies have investigated how this might influence structure learning[Gong et al., 2017, Huang et al., 2020, Löwe et al., 2022] and hence, the problem of structure learning from time-varying data remains underinvestigated.

References

Paul D Allison. Missing data. *The SAGE handbook of quantitative methods in psychology*, pages 72–89, 2009.

Russell G Almond, Robert J Mislevy, Linda S Steinberg, Duanli Yan, and David M Williamson. *Bayesian networks in educational assessment.* Springer, 2015.

Charles K Assaad, Emilie Devijver, and Eric Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022.

Shatha Awawdeh, Hossam Faris, and Hazem Hiary. Evoimputer: An evolutionary approach for missing data imputation and feature selection in the context of supervised learning. *Knowledge-Based Systems*, 236:107734, 2022.

Melissa J Azur, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. Multiple imputation by chained equations: what is it and how does it work? *International Journal of Methods in Psychiatric Research*, 20(1):40–49, 2011.

Nikolay Balov et al. Consistent model selection of discrete Bayesian networks from incomplete data. *Electronic Journal of Statistics*, 7:1047–1077, 2013.

Ingo A Beinlich, Henri Jacques Suermondt, R Martin Chavez, and Gregory F Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pages 247–256, 1989.

Tineke Blom, Anna Klimovskaia, Sara Magliacane, and Joris M Mooij. An upper bound for random measurement error in causal discovery. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2018.

Tjebbe Bodewes and Marco Scutari. Learning Bayesian networks from incomplete data with the node-average likelihood. *International Journal of Approximate Reasoning*, 138:145–160, 2021.

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

Remco R Bouckaert. Properties of Bayesian belief network learning algorithms. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109, 1994.

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

Fanyu Bu, Zhikui Chen, Qingchen Zhang, and Laurence T Yang. Incomplete high-dimensional data imputation algorithm using feature selection and clustering analysis on cloud. *The Journal of Supercomputing*, 72(8):2977–2990, 2016.

David Maxwell Chickering. Learning bayesian networks is np-complete. *Learning from data: Artificial intelligence and statistics V*, pages 121–130, 1996.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3(Nov):507–554, 2002.

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

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

Diego Colombo, Marloes H Maathuis, Markus Kalisch, and Thomas S Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, pages 294–321, 2012.

Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

Anthony C Constantinou. The Bayesys user manual. *Queen Mary University of London, London, England, UK, http://bayesian-ai. eecs. qmul. ac. uk/bayesys*, 2019.

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

Anthony C Constantinou. Investigating the efficiency of the asian handicap football betting market with ratings and Bayesian networks. *Journal of Sports Analytics*, 8(3):171–193, 2022.

Anthony C Constantinou and Norman Fenton. The future of the london buy-to-let property market: Simulation with temporal Bayesian networks. *PLoS ONE*, 12(6):e0179297, 2017.

Anthony C Constantinou, Mark Freestone, William Marsh, Norman Fenton, and Jeremy Coid. Risk assessment and risk management of violent reoffending among prisoners. *Expert Systems with Applications*, 42(21):7511–7529, 2015.

Anthony C Constantinou, Norman Fenton, William Marsh, and Lukasz Radlinski. From complex questionnaire and interviewing data to intelligent Bayesian network models for medical decision support. *Artificial Intelligence in Medicine*, 67:75–93, 2016.

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

Anthony C Constantinou, Yang Liu, Neville K Kitson, Kiattikun Chobtham, and Zhigao Guo. Effective and efficient structure learning with pruning and model averaging strategies. *International Journal of Approximate Reasoning*, 151:292–321, 2022.

Anthony C Constantinou, Neville K Kitson, Yang Liu, Kiattikun Chobtham, Arian Hashemzadeh Amirkhizi, Praharsh A Nanavati, Rendani Mbuvha, and Bruno Petrungaro. Open problems in causal structure learning: A case study of covid-19 in the uk. *Expert Systems with Applications*, 234:121069, 2023.

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

Haoyue Dai, Peter Spirtes, and Kun Zhang. Independence testing-based approach to causal discovery under measurement error and linear non-gaussian models. In *Proceedings of the Thirty-Fifth Conference on Neural Information Processing Systems*, 2022.

Rhian M Daniel, Michael G Kenward, Simon N Cousens, and Bianca L De Stavola. Using causal diagrams to guide analysis in missing data problems. *Statistical methods in medical research*, 21 (3):243–256, 2012.

Martijn de Jongh and Marek J Druzdzel. A comparison of structural distance measures for causal Bayesian network models. *Recent advances in intelligent information systems, challenging problems of science, computer science series*, pages 443–456, 2009.

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

Vera Djordjilović, Monica Chiogna, and Jiří Vomlel. An empirical comparison of popular structure learning algorithms with a view to gene network inference. *International Journal of Approximate Reasoning*, 88:602–613, 2017.

Gauthier Doquire and Michel Verleysen. Feature selection with missing data using mutual information estimators. *Neurocomputing*, 90:3–11, 2012.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Mohamad Faiz Dzulkalnine and Roselina Sallehuddin. Missing data imputation with fuzzy feature selection for diabetes dataset. *SN Applied Sciences*, 1(4):1–12, 2019.

Michel Fortin and Roland Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems.* Elsevier, 2000.

Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. Gp-vae: Deep probabilistic time series imputation. In *Proceedings of the Twenty-Third International Conference on Artificial Intelligence and Statistics*, pages 1651–1661, 2020.

Nir Friedman et al. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 125–133, 1997.

Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schölkopf. Kernel measures of conditional dependence. In *Proceedings of the Twentieth Conference on Neural Information Processing Systems*, 2007.

Alexander Gain and Ilya Shpitser. Structure learning under missing data. In *Proceedings of the Ninth Conference on Probabilistic Graphical Models*, pages 121–132, 2018.

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

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Proceedings of the Thirty-Fourth Conference on Neural Information Processing Systems*, pages 9574–9586, 2021.

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

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

Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080. Cambridge, MA, 1988.

Mingming Gong, Kun Zhang, Bernhard Schölkopf, Clark Glymour, and Dacheng Tao. Causal discovery from temporally aggregated time series. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, volume 2017, 2017.

John W Graham. Missing data analysis: Making it work in the real world. *Annual Review of Psychology*, 60:549–576, 2009.

Armen Hakhverdian. The causal flow between public opinion and policy: government responsiveness, leadership, or counter movement? *West European Politics*, 35(6):1386–1406, 2012.

Trevor Hastie, Rahul Mazumder, Jason D Lee, and Reza Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1): 3367–3402, 2015.

Dominique MA Haughton. On the choice of a model to fit data from an exponential family. *The annals of statistics*, pages 342–355, 1988.

David Heckerman, Dan Geiger, and David M Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

David Earl Heckerman, Eric J Horvitz, and Bharat N Nathwani. Toward normative expert systems: Part i the pathfinder project. *Methods of Information in Medicine*, 31(02):90–105, 1992.

Daniel F Heitjan and Donald B Rubin. Ignorability and coarse data. *The annals of statistics*, pages 2244–2253, 1991.

Minh Hieu Nguyen, Jimmy Armoogum, and Emeli Adell. Feature selection for enhancing purpose imputation using global positioning system data without geographic information system data. *Transportation Research Record*, 2675(5):75–87, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, pages 6840–6851, 2020.

James Honaker, Gary King, and Matthew Blackwell. Amelia ii: A program for missing data. *Journal of Statistical Software*, 45:1–47, 2011.

Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

Patrik Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In *Proceedings of the Twenty-First Conference on Neural Information Processing Systems*, 2008.

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

Biwei Huang, Kun Zhang, Yizhu Lin, Bernhard Schölkopf, and Clark Glymour. Generalized score functions for causal discovery. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1551–1560, 2018.

Biwei Huang, Kun Zhang, Jiji Zhang, Joseph Ramsey, Ruben Sanchez-Romero, Clark Glymour, and Bernhard Schölkopf. Causal discovery from heterogeneous/nonstationary data. *The Journal of Machine Learning Research*, 21(1):3482–3534, 2020.

Antti Hyttinen, Frederick Eberhardt, and Matti Järvisalo. Constraint-based causal discovery: Conflict resolution with answer set programming. In *Proceedings of the Thirtieth conference on Uncertainty in artificial intelligence*, pages 340–349, 2014.

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

Fattaneh Jabbari, Joseph Ramsey, Peter Spirtes, and Gregory Cooper. Discovery of causal models that contain latent variables through Bayesian scoring of independence constraints. In *Machine Learning and Knowledge Discovery in Databases: European Conference*, pages 142–157, 2017.

Daniel Jarrett, Bogdan C Cebere, Tennison Liu, Alicia Curth, and Mihaela van der Schaar. Hyperimpute: Generalized iterative imputation with automatic model selection. In *Proceedings of the Thirty-Ninth International Conference on Machine Learning*, pages 9916–9937, 2022.

Heejin Jin, Surin Jung, and Sungho Won. missForest with feature selection using binary particle swarm optimization improves the imputation accuracy of continuous data. *Genes & Genomics*, 44(6):651–658, 2022.

Chisimkwuo John, Emmanuel J Ekpenyong, and Charles C Nworu. Imputation of missing values in economic and financial time series data using five principal component analysis approaches. *CBN Journal of Applied Statistics*, 10(1):51–73, 2019.

Neville K Kitson and Anthony C Constantinou. The impact of variable ordering on bayesian network structure learning. *arXiv preprint arXiv:2206.08952*, 2022.

Neville K Kitson, Anthony C Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of Bayesian network structure learning. *Artificial Intelligence Review*, pages 1–94, 2023.

Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *The Journal of Machine Learning Research*, 5:549–573, 2004.

ME Kragt, LTH Newham, AJ Jakeman, et al. A Bayesian network approach to integrating economic and biophysical modelling. In *Proceedings of the Eighteenth World IMACS/MODSIM Congress on Modelling and Simulation*, pages 2377–2383. Citeseer, 2009.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Trent Kyono, Yao Zhang, Alexis Bellot, and Mihaela van der Schaar. Miracle: Causally-aware imputation via learning missing data mechanisms. In *Proceedings of the Thirty-Fourth Conference on Neural Information Processing Systems*, pages 23806–23817, 2021.

Steffen L Lauritzen and David 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.

Katherine J Lee and John B Carlin. Multiple imputation for missing data: fully conditional specification versus multivariate normal imputation. *American Journal of Epidemiology*, 171(5): 624–632, 2010.

Honghao Li, Vincent Cabeli, Nadir Sella, and Hervé Isambert. Constraint-based causal structure learning with consistent separating sets. In *Proceedings of the Thirty-Second Conference on Neural Information Processing Systems*, 2019.

Wei-Chao Lin and Chih-Fong Tsai. Missing value imputation: a review and analysis of the literature (2006–2017). *Artificial Intelligence Review*, 53(2):1487–1509, 2020.

Wei-Chao Lin, Chih-Fong Tsai, and Jia Rong Zhong. Deep learning for missing value imputation of continuous data and the effect of data discretization. *Knowledge-Based Systems*, 239:108079, 2022.

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

Chia-Hui Liu, Chih-Fong Tsai, Kuen-Liang Sue, and Min-Wei Huang. The feature selection effect on missing value imputation of medical datasets. *Applied Sciences*, 10(7):2344, 2020.

Yang Liu and Anthony Constantinou. Improving the imputation of missing data with markov blanket discovery. In *The Eleventh International Conference on Learning Representations*, 2023.

Yang Liu and Anthony C Constantinou. Greedy structure learning from data that contain systematic missing values. *Machine Learning*, 111(10):3867–3896, 2022.

Yang Liu, Anthony C Constantinou, and Zhigao Guo. Improving Bayesian network structure learning in the presence of measurement error. *Journal of Machine Learning Research*, 23(324): 1–28, 2022.

Sindy Löwe, David Madras, Richard Zemel, and Max Welling. Amortized causal discovery: Learning to infer causal graphs from time-series data. In *Proceedings of the First Conference on Causal Learning and Reasoning*, pages 509–525, 2022.

Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. In *Proceedings of the Twelfth Conference on Neural Information Processing Systems*, 1999.

Fernando Martel García. Definition and diagnosis of problematic attrition in randomized controlled experiments. *Available at SSRN 2302735*, 2013.

Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *Proceedings of the Thirty-Sixth International Conference on Machine Learning*, pages 4413–4423, 2019.

Scott McLachlan, Kudakwashe Dube, Graham A Hitman, Norman E Fenton, and Evangelia Kyrimi. Bayesian networks in healthcare: Distribution by medical condition. *Artificial Intelligence in Medicine*, 107:101912, 2020.

Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 403–410, 1995.

Karthika Mohan, Judea Pearl, and Jin Tian. Graphical models for inference with missing data. In *Proceedings of the Twenty-Sixth Conference on Neural Information Processing Systems*, pages 1277–1285, 2013.

Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. Missing data imputation using optimal transport. In *Proceedings of the Thirty-Seventh International Conference on Machine Learning*, pages 7130–7140, 2020.

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, 107:107501, 2020.

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

Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.

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

Sascha Ott, Seiya Imoto, and Satoru Miyano. Finding optimal models for small gene networks. In *Biocomputing*, pages 557–567, 2003.

Alma B Pedersen, Ellen M Mikkelsen, Deirdre Cronin-Fenton, Nickolaj R Kristensen, Tra My Pham, Lars Pedersen, and Irene Petersen. Missing data and multiple imputation in clinical epidemiological research. *Clinical Epidemiology*, 9:157, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jonas Peters, Joris M Mooij, Dominik Janzing, and Bernhard Schölkopf. Causal discovery with continuous additive noise models. *The Journal of Machine Learning Research*, 15(1):2009–2053, 2014.

Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.

Sophia Rabe-Hesketh and Anders Skrondal. Ignoring non-ignorable missingness. *psychometrika*, 88(1):31–50, 2023.

Joseph Ramsey. Improving accuracy and scalability of the PC algorithm by maximizing p-value. *arXiv preprint arXiv:1610.00378*, 2016.

Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A million variables and more: the fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics*, 3(2):121–129, 2017.

Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

Andrea Ruggieri, Francesco Stranieri, Fabio Stella, and Marco Scutari. Hard and Soft EM in Bayesian Network Learning from Incomplete Data. *Algorithms*, 13(12):329, 2020.

Mauro Scanagatta. Bayesian network learning improved project. `https://github.com/mauro-idsia/blip`, 2019.

Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon. Learning Bayesian networks with thousands of variables. In *Proceedings of the Twenty-Eighth Conference on Neural Information Processing Systems*, 2015.

Mauro Scanagatta, Giorgio Corani, and Marco Zaffalon. Improved local search in Bayesian networks structure learning. In *Advanced Methodologies for Bayesian Networks*, pages 45–56, 2017.

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

Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

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

Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019.

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

Amir Masoud Sefidian and Negin Daneshpour. Missing value imputation using a novel grey based fuzzy c-means, mutual information based feature selection, and regression model. *Expert Systems with Applications*, 115:68–94, 2019.

Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(10), 2006.

Tomi Silander, Petri Kontkanen, and Petri Myllymäki. On sensitivity of the map Bayesian network structure to the equivalent sample size parameter. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 360–367, 2007.

Tomi Silander, Janne Leppä-Aho, Elias Jääsaari, and Teemu Roos. Quotient normalized maximum likelihood criterion for learning Bayesian network structures. In *Proceedings of the Twenth-First International Conference on Artificial Intelligence and Statistics*, pages 948–957, 2018.

Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.

Ajit P Singh and Andrew W Moore. *Finding optimal Bayesian networks by dynamic programming.* Carnegie Mellon University. Center for Automated Learning and Discovery, 2005.

Dušan Sovilj, Emil Eirola, Yoan Miche, Kaj-Mikael Björk, Rui Nian, Anton Akusok, and Amaury Lendasse. Extreme learning machine for missing data using multiple imputations. *Neurocomputing*, 174:220–231, 2016.

Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

Peter Spirtes, Clark Glymour, and Richard Scheines. Causality from probability. In *Proceedings of the Conference on Advanced Computing for the Social Sciences*, 1990.

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

Helen M Stallman, Denise Beaudequin, Daniel F Hermens, and Daniel Eisenberg. Modelling the relationship between healthy and unhealthy coping strategies to understand overwhelming distress: A Bayesian network approach. *Journal of Affective Disorders Reports*, 3:100054, 2021.

Harald Steck and Tommi Jaakkola. On the dirichlet prior and Bayesian regularization. In *Proceedings of the Fifteenth Conference on Neural Information Processing Systems*, 2002.

Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Daniel J Stekhoven and Maintainer Daniel J Stekhoven. Package 'missforest'. *R package version*, 1, 2013.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, pages 3008–3021, 2020.

Eric V Strobl, Shyam Visweswaran, and Peter L Spirtes. Fast causal inference with non-random missingness by test-wise deletion. *International Journal of Data Science and Analytics*, 6(1): 47–62, 2018.

Marc Teyssier and Daphne Koller. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 584–590, 2005.

Felix Thoemmes and Norman Rose. A cautious note on auxiliary variables that can increase bias in missing data problems. *Multivariate Behavioral Research*, 49(5):443–459, 2014.

Yan Tian, Kaili Zhang, Jianyuan Li, Xianxuan Lin, and Bailin Yang. Lstm-based traffic flow prediction with missing data. *Neurocomputing*, 318:297–305, 2018.

Cao Truong Tran, Mengjie Zhang, Peter Andreae, Bing Xue, and Lam Thu Bui. Improving performance of classification on incomplete data using feature selection and clustering. *Applied Soft Computing*, 73:848–861, 2018.

Michail Tsagris. A new scalable Bayesian network learning algorithm with applications to economics. *Computational Economics*, 57(1):341–367, 2021.

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

Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale markov blanket discovery. In *FLAIRS Conference*, volume 2, pages 376–380, 2003b.

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

Ruibo Tu, Cheng Zhang, Paul Ackermann, Karthika Mohan, Hedvig Kjellström, and Kun Zhang. Causal discovery in the presence of missing data. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 1762–1770, 2019.

Caroline Uhler, Garvesh Raskutti, Peter Bühlmann, and Bin Yu. Geometry of the faithfulness assumption in causal inference. *The Annals of Statistics*, pages 436–463, 2013.

Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45:1–67, 2011.

Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 255–270, 1990.

Feng Wang, Xing Ge, and Danwen Huang. Government intervention, human mobility, and covid-19: a causal pathway analysis from 121 countries. *Sustainability*, 14(6):3694, 2022.

Naftali Weinberger. Faithfulness, coordination and causal coincidences. *Erkenntnis*, 83(2):113–133, 2018.

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

Michalis Xenos. Prediction and assessment of student behaviour in open and distance education in computers using Bayesian networks. *Computers & Education*, 43(4):345–359, 2004.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Proceedings of the Thirty-Second Conference on Neural Information Processing Systems*, 2019.

Yuqin Yang, AmirEmad Ghassami, Mohamed Nafea, Negar Kiyavash, Kun Zhang, and Ilya Shpitser. Causal discovery in linear latent variable models subject to measurement error. In *Proceedings of the Thirty-Fifth Conference on Neural Information Processing Systems*, 2022.

Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, pages 5689–5698, 2018.

Jiaxuan You, Xiaobai Ma, Yi Ding, Mykel J Kochenderfer, and Jure Leskovec. Handling missing data with graph representation learning. In *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, pages 19075–19087, 2020.

Kui Yu, Yajing Yang, and Wei Ding. Causal feature selection with missing data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4):1–24, 2022.

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

Tadesse Zemicheal and Thomas G Dietterich. Anomaly detection in the presence of missing values for weather data quality control. In *Proceedings of the Second ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 65–73, 2019.

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

K Zhang and A Hyvärinen. On the identifiability of the post-nonlinear causal model. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 647–655, 2009.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 804–813, 2011.

Kun Zhang, Mingming Gong, Joseph Ramsey, Kayhan Batmanghelich, Peter Spirtes, and Clark Glymour. Causal discovery with linear non-Gaussian models under measurement error: Structural identifiability results. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 1063–1072, 2018.

Shichao Zhang. Nearest neighbor selection for iteratively knn imputation. *Journal of Systems and Software*, 85(11):2541–2552, 2012.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In *Proceedings of the Thirty-First Conference on Neural Information Processing Systems*, 2018.