Improved Scalability and Accuracy of Bayesian Network Structure Learning in the Score-and-Search Paradigm

by

Charupriya Sharma

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Simon de Givry Unité de Mathématiques et Informatique Appliquées de Toulouse Centre de recherches de Toulouse, INRAE
Supervisor:	Peter van Beek David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Kate Larson David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Jesse Hoey David R. Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Bruce Hellinga Civil and Environmental Engineering University of Waterloo

Author's Declaration

Statement of Contributions

This thesis is mostly based on the following co-authored published articles. We use the taxonomy developed and refined by the Consortia Advancing Standards in Research Administration (CASRAI) and the National Information Standards Organization (NISO) to specify the individual contributions (see Table 1).

1. Zhenyu A. Liao, Charupriya Sharma, James Cussens, and Peter van Beek. Finding All Bayesian Network Structures within a Factor of Optimal. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, January, 2019.

Author	Roles
Zhenyu A. Liao	Writing – original draft; Conceptualization; Investigation; Software; Visualization;
Charupriya Sharma	Conceptualization; Formal analysis; Software; Visualization; Writing – review/editing;
James Cussens	Software; Writing – review/editing;
Peter van Beek	Supervision; Funding acquisition; Conceptualization; Methodology; Writing – review/editing

Chapter 3 is a revised and extended version of this paper.

2. Charupriya Sharma, Zhenyu A. Liao, James Cussens, and Peter van Beek. A Score-and-Search Approach to Learning Bayesian Networks with Noisy-OR Relations. *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, Aalborg, Denmark, September, 2020.

Author	Roles
Charupriya Sharma	Writing – original draft; Conceptualization; Formal analysis; Investigation; Methodology; Software; Validation; Visualization;
Zhenyu A. Liao	Software; Writing – review/editing;
James Cussens	Software; Writing – review/editing;
Peter van Beek	Supervision; Funding acquisition; Conceptualization; Methodology; Writing – review/editing

Chapter 4 is a revised and extended version of this paper.

3. Zhenyu A. Liao, Charupriya Sharma, Dongshu Luo, and Peter van Beek. An Empirical Study of Scoring Functions for Learning Bayesian Networks in Model Averaging. *Proceedings of the 35th Canadian Conference on Artificial Intelligence*, Toronto, Ontario, May, 2022.

Author	Roles
Zhenyu A. Liao	Writing – original draft; Conceptualization; Formal analysis; Investigation; Methodology; Software;
Charupriya Sharma	Conceptualization; Investigation; Software; Writing – review/editing;
Dongshu Luo	Investigation; Software;
Peter van Beek	Supervision; Funding acquisition; Validation; Visualization; Writing – review/editing

   This paper influenced the performance metrics and the presentation of the experiments shown in Chapter 3 and Chapter 5.

4. Charupriya Sharma and Peter van Beek. Scalable Bayesian Network Structure Learning with Splines. *Proceedings of the 11th International Conference on Probabilistic Graphical Models*, Almería, Spain, October, 2022.

Author	Roles
Charupriya Sharma	Writing – original draft; Conceptualization; Formal analysis; Investigation; Methodology; Software; Validation; Visualization;
Peter van Beek	Supervision; Funding acquisition; Methodology; Software; Writing – review/editing

   Chapter 6 is a revised and extended version of this paper.

Table 1: CRediT (Contributor Roles Taxonomy), a high-level taxonomy that can be used to represent the roles typically played by contributors to research outputs. The roles describe each contributor's specific contribution to the scholarly output.

Role	Description
Conceptualization	Ideas; formulation or evolution of overarching research goals and aims.
Formal analysis	Application of statistical, mathematical, computational, or other formal techniques to analyze or synthesize study data.
Funding acquisition	Acquisition of the financial support for the project leading to this publication.
Investigation	Conducting a research and investigation process, specifically performing the experiments, or data/evidence collection.
Methodology	Development or design of methodology; creation of models.
Software	Programming, software development; designing computer programs; testing of existing code components.
Supervision	Oversight and leadership responsibility for the research activity planning and execution.
Validation	Verification, whether as a part of the activity or separate, of the overall replication/reproducibility of results/experiments and other research outputs.
Visualization	Preparation, creation and/or presentation of the published work, specifically visualization/data presentation.
Writing – original draft	Preparation, creation and/or presentation of the published work, specifically writing the initial draft.
Writing – review/editing	Preparation, creation and/or presentation of the published work, specifically critical review, commentary or revision.

Abstract

A Bayesian network is a probabilistic graphical model that consists of a directed acyclic graph (DAG), where each node is a random variable and attached to each node is a conditional probability distribution (CPD). A Bayesian network (BN) can either be constructed by a domain expert or learned automatically from data using the well-known score-and-search approach, a form of unsupervised machine learning. Our interest here is in BNs as a knowledge discovery or data analysis tool, where the BN is learned automatically from data and the resulting BN is then studied for the insights that it provides on the domain such as possible cause-effect relationships, probabilistic dependencies, and conditional independence relationships. Previous work has shown that the accuracy of a data analysis can be improved by (i) incorporating structured representations of the CPDs into the score-and-search approach for learning the DAG and by (ii) learning a set of DAGs from a dataset, rather than a single DAG, and performing a technique called model averaging to obtain a representative DAG.

This thesis focuses on improving the accuracy of the score-and-search approach for learning a BN and in scaling the approach to datasets with larger numbers of random variables. We introduce a novel model averaging approach to learning a BN motivated by performance guarantees in approximation algorithms. Our approach considers all optimal and all near-optimal networks for model averaging. We provide pruning rules that retain optimality while enabling our approach to scale to BNs significantly larger than the current state of the art. We extend our model averaging approach to simultaneously learn the DAG and the local structure of the CPDs in the form of a noisy-OR representation. We provide an effective gradient descent algorithm to score a candidate noisy-OR using the widely used BIC score and we provide pruning rules that allow the search to successfully scale to medium sized networks. Our empirical results provide evidence for the success of our approach to learning Bayesian networks that incorporate noisy-OR relations. We also extend our model averaging approach to simultaneously learn the DAG and the local structure of the CPD using neural networks representations. Our approach compares favourably with approaches like decision trees, and performs well in instances with low amounts of data. Finally, we introduce a score-and-search approach to simultaneously learn a DAG and model linear and non-linear local probabilistic relationships between variables using multivariate adaptive regression splines (MARS). MARS are polynomial regression models represented as piecewise spline functions. We show on a set of discrete and continuous benchmark instances that our proposed approach can improve the accuracy of the learned graph while scaling to instances with over 1,000 variables.

Acknowledgements

I would like to thank my supervisor, Prof. Peter van Beek, for all his support and guidance, without which this thesis would not have been possible. I would also like to thank my co-authors for all of their work on our projects together, as well as Prof. Kate Larson, Prof. Jesse Hoey and Prof. Ali Ghodsi for valuable their feedback on my research.

Table of Contents

List of Figures **xii**

List of Tables **xv**

List of Acronyms and Abbreviations **xviii**

1 Introduction **1**

1.1 Related Work 2

1.1.1 Learning a Bayesian Network 3

1.1.2 Model Averaging 5

1.1.3 Incorporating Local Structure 6

1.2 Contributions 7

2 Background **9**

2.1 Bayesian Networks 9

2.2 Bayesian Network Structure Learning 10

2.2.1 Scoring Functions 12

2.3 Model Averaging 13

2.4 Representing CPDs 15

2.4.1 Noisy-OR 15

2.4.2 Decision Trees 17

2.5 Performance Evaluation Metrics . . . 18
2.5.1 DAG Performance Metrics: Skeleton . . . 18
2.5.2 DAG Performance Metrics: CPDAG . . . 19
2.5.3 Density Estimation and Inference Metric . . . 21

3 The Credible Set Approach **23**
3.1 Overview . . . 23
3.2 Credible Sets . . . 25
3.3 Pruning Rules . . . 28
3.3.1 Pruning with BIC Score . . . 29
3.3.2 Pruning with BDeu Score . . . 33
3.4 Experimental Evaluation . . . 35
3.4.1 The Credible Set Approach . . . 35
3.4.2 Credible Set vs. K-Best . . . 40
3.4.3 Credible Set vs. Bootstrapping . . . 43
3.5 Summary . . . 44

4 Local Structure: Noisy-OR **47**
4.1 Overview . . . 47
4.2 Score-and-Search with Noisy-OR . . . 48
4.2.1 BIC Score for Noisy-OR Relations . . . 50
4.2.2 Pruning Rules . . . 51
4.2.3 Overall Algorithm for Structure Learning . . . 53
4.3 Experimental Evaluation . . . 53
4.3.1 Recovery of Noisy-ORs in Synthetic Datasets . . . 54
4.3.2 Presence of Noisy-OR Relations in Standard Benchmarks . . . 55
4.3.3 Performance on Ground Truth Networks . . . 57
4.4 Summary . . . 58

5 Local Structure: Neural Networks 59
5.1 Neural Networks to Represent CPDs 60
5.1.1 Learning CPDs with Neural Networks 60
5.1.2 Learning CPDs with Decision Trees 62
5.1.3 Overall Algorithm for Structure Learning 64
5.2 Experimental Evaluation 65
5.3 Summary 69

6 Local Structure: Multivariate Adaptive Regression Splines 74
6.1 MARS: Multivariate Adaptive Regression Splines 75
6.2 Scaling with Feature Selection 77
6.3 Experimental Evaluation 79
6.3.1 Implementation Details 80
6.3.2 Performance on Discrete Variables 81
6.3.3 Performance on Continuous Variables 86
6.3.4 Model Complexity 87
6.4 Summary 87

7 Conclusions 88
7.1 Future Work 89

References 91

List of Figures

1.1 Directed acyclic graph and (partial) specification of the conditional probability distributions of the example "Cancer" Bayesian network [62]. 2

2.1 A Bayesian network with four variables. Note that $P(D = 1 \mid A, B, C) = 1 - P(D = 0 \mid A, B, C)$ for all values of A, B, and C. 10

2.2 A sample dataset with four random variables. 11

2.3 Candidate parent sets for variable D with the associated scores. 11

2.4 Model averaging example with learned networks with confidence values in blue, and the model averaged network in green. Using a threshold of $t = 0.5$, edge (C, D) is not considered significant and is removed from the final averaged network. 14

2.5 Causal structure for a Bayesian network with a noisy-OR relation, where the set of causes $\Pi_i := \{V_{i1}, \ldots, V_{i|\Pi_i|}\}$ leads to effect V_i and there is a noisy-OR relation at node V_i. 16

2.6 Two representations of the same CPD $P(D \mid A, B, C)$ for a child node D and a parent set $\{A, B, C\}$: tabular CPT (left) and decision tree representation (right). 17

2.7 An example of Markov equivalence classes for a three node network. The blue network and the green networks are in different Markov equivalence classes. 20

3.1 The deviation ϵ from the optimal BDeu score by k, where k is the number of best scoring networks retrieved. The corresponding values of the BF ($\epsilon = \log(\text{BF})$) are presented on the right. For example, if the desired BF value is 20, then all networks falling below the dash line at 20 are credible. 42

3.2 Comparison of bootstrap and credible set model averaging methods, for various scoring functions and performance measures for undirected edges (skeleton): misclassification cost (top, lower values are better); F_β score (bottom, higher values are better). All methods used a fixed threshold of 0.5. 45

3.3 Comparison of bootstrap and credible set model averaging methods, for various scoring functions and performance measures for directed edges (CPDAGs): structural Hamming distance (top, lower values are better); multi-class F_β score (bottom, higher values are better). All methods used a fixed threshold of 0.5. 46

4.1 Causal structure for a BN with a noisy-OR relation, where the set of causes $\Pi_i := \{V_{i1}, \ldots, V_{i|\Pi_i|}\}$ leads to effect V_i and there is a noisy-OR relation at variable V_i. 49

5.1 Comparison of scoring with conditional probability tables (CPT), scoring with neural networks (NN), and scoring with decision trees (DT) using structural Hamming distance over CPDAGs (top, lower values are better) and multi-class F_β score over CPDAGs (bottom, higher values are better) as the performance measure. All methods used a fixed threshold of 0.5. 70

5.2 Comparison of scoring with conditional probability tables (CPT), scoring with neural networks (NN), and scoring with decision trees (DT) using misclassification cost $\alpha \cdot \mathrm{FN} + \mathrm{FP}$ over skeletons (top, lower values are better) and F_β score over skeletons (bottom, higher values are better) as the performance measure. All methods used a fixed threshold of 0.5. 71

6.1 MARS (red) and linear regression (blue) models for the same set of points. MARS places a knot at every change of the slope. 76

6.2 Comparison of structure learning approaches on various networks from Tetrad, where n is the number of variables and the number of edges in the ground truth network: our BN-FS-MARS (BFM), tabular CPT with maximum parent set size of two (BIC2), and GES (GES). For each ground truth network, we report the following for corresponding learned networks for sample sizes $N = 1000$ and $N = 5000$: (a) F_1 score, (b) TP (true positives), (c) sum of TP and WD (wrong direction), and (d) FP (false positives). 82

6.3 Comparison of structure learning approaches on various networks from bnlearn: our BN-FS-MARS (BFM), tabular CPT with maximum parent set size of two (BIC2), and GES (GES). For each ground truth network, we report the following for corresponding learned networks for sample sizes $N = 1000$ and $N = 5000$: (a) -score, (b) TP (true positives), (c) sum of TP and WD (wrong direction), and (d) FP (false positives). 83

6.4 Results of BNSL on the benchmark **link** from bnlearn with N=5000 samples to show the proportion of TP=true positive, WD=wrong direction, FN=false negatives, FP=false positives in the learned DAG. 84

6.5 Results of BNSL on the benchmark **munin1** from bnlearn with N=5000 samples to show the proportion of TP=true positive, WD=wrong direction, FN=false negatives, FP=false positives in the learned DAG. 84

6.6 Comparison of structure learning approaches on various $n = 100$ networks from Dream4: our BN-FS-MARS (BFM), tabular CPT with maximum parent set size of two (BIC2), and GES (GES). For each ground truth network, we report the following for corresponding learned networks for sample sizes $N = 1000$ and $N = 5000$: (a) F_1 score, (b) TP (true positives), (c) sum of TP and WD (wrong direction), and (d) FP (false positives). 85

List of Tables

1 CRediT (Contributor Roles Taxonomy), a high-level taxonomy that can be used to represent the roles typically played by contributors to research outputs. The roles describe each contributor's specific contribution to the scholarly output. vi

1.1 Representative data analyses using Bayesian networks, where n is the number of random variables and N is the number of instances in the data set. 3

3.1 UCI datasets (*left*, *middle*) and bnlearn Bayesian networks (*right*), where n is the number of variables in the dataset or network, and N is the number of instances in the original UCI dataset. 36

3.2 The search time T and the number of collected networks $|\mathcal{G}|$ for the credible set method (BF = 3, 20, and 150) on the UCI datasets using BIC scoring, where n is the number of random variables in the dataset and N is the number of instances in the dataset. 38

3.3 The search time T and the number of collected networks $|\mathcal{G}|$ for the credible set method (BF = 3, 20, and 150) on the bnlearn networks using BIC scoring, where n is the number of random variables in the dataset, N is the number of instances in the dataset, and OM = Out of Memory. 39

3.4 The search time T and the number of collected networks k, $|\mathcal{G}_k|$ and $|\mathcal{G}_{20}|$ for KBest, KBestEC, and credible set (BF = 20) methods on the UCI datasets using BDeu scoring, where n is the number of random variables in the dataset, N is the number of instances in the dataset, OT = Out of Time, and ES = Error in Scoring. Note that $|\mathcal{G}_k|$ is the number of DAGs covered by the k-best MECs in KBestEC and $|\mathcal{M}_{20}|$ is the number of MECs for the networks collected by the credible set method. 41

4.1 (Top) Median relative error in noisy-OR parameters and (bottom) median conditional KL divergence of noisy-OR CPTs learned by our Algorithm 4.1, denoted KL, and Vomlel's [108] expectation-maximization algorithm, denoted EM, from ground truth for various parent set sizes. 55

4.2 Total number of variables where a noisy-OR relation is selected (m), and average (ave.) and maximum (max.) percentage of networks in the set of credible networks that select noisy-OR relations for these nodes, for various benchmarks, where n is the number of variables in the dataset or network and N is the number of instances in the dataset. OT indicates a dataset that could not be solved within the time limit. 56

4.3 Median relative inference error for the best and worst scoring network in the set of credible networks learned by Algorithm 4.1 and the full CPT against the ground truth network. The datasets with N instances were generated from various ground truth BNs with n nodes. 57

5.1 UCI datasets (*left*, *middle*) and bnlearn Bayesian networks (*right*), where n is the number of variables in the dataset or network, and N is the number of instances in the original UCI dataset. 66

5.2 Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using structural Hamming distance over CPDAGs as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Positive values favor CPT, negative values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks. . . 72

5.3 Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using multi-class F_1 score over CPDAGs as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Negative values favor CPT, positive values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks. 72

5.4 Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using misclassification cost FP + FN over skeletons as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Positive values favor CPT, negative values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks. . . 73

5.5 Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using F_1 score over skeletons as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Negative values favor CPT, positive values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks. 73

6.1 Bayesian networks from bnlearn, where n is the number of variables in the network and m is the number of edges in the network. 80

List of Acronyms and Abbreviations

BDeu	Bayesian Dirichlet equivalent uniform
BF	Bayes factor
BIC	Bayesian information criterion
BN	Bayesian network
BNSL	Bayesian network structure learning
CPD	Conditional probability distribution
CPDAG	Completed partially directed acyclic graph
DAG	Directed acyclic graph
MARS	Multivariate adaptive regression splines
MEC	Markov equivalence class
SHD	Structural Hamming distance

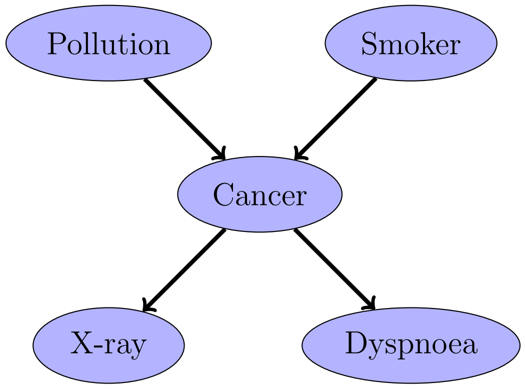
Chapter 1

Introduction

A Bayesian network is a probabilistic graphical model that consists of a labeled directed acyclic graph (DAG) in which the vertices correspond to random variables, the edges represent direct influence of one random variable on another, and each vertex is labeled with a conditional probability distribution that specifies the dependence of that variable on its set of parents in the DAG [80]. A Bayesian network can alternatively be viewed as a factorized representation of the joint probability distribution over the random variables and as an encoding of conditional independence assertions.

Example 1.1. *Consider the well-known "Cancer" Bayesian network over five random variables shown in Figure 1.1. Whether one is a smoker has a direct influence on whether one is diagnosed with lung cancer, and whether one has lung cancer has a direct influence on whether one experiences dyspnoea (shortness of breath). The DAG encodes the conditional independence assertion that P(X-ray | Cancer) = P(X-ray | Cancer, Pollution, Smoker)); i.e., given whether there is cancer, what may have caused the cancer is irrelevant to whether the X-ray shows cancer.*

Bayesian networks have applications in knowledge discovery, probabilistic density estimation, and prediction [27, 61, 109]. A Bayesian network (BN) can either be constructed by a domain expert or learned automatically from data, a form of unsupervised machine learning. Our interest here is in BNs as a knowledge discovery or data analysis tool, where the BN is learned automatically from data and the resulting BN is then studied for the insights that it provides on the domain such as possible cause-effect relationships, probabilistic dependencies, and conditional independence relationships. BNs have been shown to be an important data analysis tool in diverse areas including banking, environment,



P(X-ray = true | Cancer = true) = 0.90	P(Dyspnoea = true | Cancer = true) = 0.65
P(X-ray = true | Cancer = false) = 0.20	P(Dyspnoea = true | Cancer = false) = 0.30
P(X-ray = false | Cancer = true) = 0.10	P(Dyspnoea = false | Cancer = true) = 0.35
P(X-ray = false | Cancer = false) = 0.80	P(Dyspnoea = false | Cancer = false) = 0.70

Figure 1.1: Directed acyclic graph and (partial) specification of the conditional probability distributions of the example "Cancer" Bayesian network [62].

medicine, safety, software, and sports (see Table 1.1 for some recent representative examples from the literature). Accordingly, research on accurate and scalable DAG or structure learning algorithms for BNs have received significant attention over the last three decades and continue to be an active area of investigation (e.g., [26, 49, 85, 93, 97, 103, 105, 106]).

1.1 Related Work

In this section, we review relevant related work. First, we review the literature on learning a DAG from a dataset. Second, we review the literature on improving the learned DAG by learning a *set* of DAGs from a single dataset and combining the set into a single DAG, a technique called model averaging. Finally, we review the literature on improving the learned DAG by incorporating information from the local structure of the conditional probability distributions into the learning process.

Table 1.1: Representative data analyses using Bayesian networks, where n is the number of random variables and N is the number of instances in the data set.

Area	n	N	Score	Description
Banking	18	1,796	BIC	Contagion interactions between credit issuers following a sovereign default [3].
Environment	12	1,900	BIC	Factors that directly impact red tide species occurrences and concentrations [36].
Medicine	11	79	BIC†	Biophysical interactions of pneumonitis (lung inflammation) due to radiation therapy in non-small-cell lung cancer [70].
Medicine	17	637	BIC	Impact of adolescent use of cannabis on brain development [79].
Medicine	26	408	BIC	Interactions between symptoms of obsessive-compulsive disorder and depression [74].
Safety	27	3,640	BIC	Interactions between safety ratings of motor carriers, carrier characteristics and safety performance metrics [55].
Software	21	12,630	BIC	Interactions between code review measures and prevalence of post-release defects [63].
Sports	22	377	BIC†	Relationships between psychological features and team performance in football [41].

† Indicates the scoring function was unspecified and was assumed to use the default scoring function of the software package used in the analysis.

1.1.1 Learning a Bayesian Network

A Bayesian network can be learned from data using algorithms that are broadly one of three types: constraint-based, score-and-search, and hybrid algorithms which are a combination of the constraint-based and score-and-search approaches. Of these three approaches, the predominant approach in Bayesian network structure learning—and the focus of this thesis—is the score-and-search approach.

The *constraint-based* approach to learning the structure of a BN uses conditional independence tests on the data to determine the conditional independence relationships between the variables under investigation, and then constructs a graph consistent with the data. Verma and Pearl [107] were the first to provide a constraint-based algorithm. More recent constraint-based algorithms include the PC [97] and PC-stable [24] algorithms. The PC algorithm starts with a complete graph and iteratively eliminates edges from the current graph via conditional independence tests. The PC stable algorithm builds on the PC algorithm with to goal to make it more resistant to the ordering in which conditional independence tests are conducted.

The *score-and-search* approach to learning the structure of a BN uses a scoring function to evaluate the fit of a proposed BN to the data and the space of directed acyclic graphs (DAGs) is searched for the best-scoring BN. Unfortunately, learning a BN from data is NP-hard, even if the number of parents per vertex in the DAG is limited to two [20]. As well, the problem is unlikely to be efficiently approximable with a good quality guarantee [59], thus motivating the use of both global (optimal) search algorithms to find better quality solutions and local (heuristic) search algorithms for scaling to larger datasets.

Global (optimal) search algorithms for learning a BN from data have been studied extensively over the past several decades and there have been proposals based on dynamic programming [60, 95, 73], A* search [112, 33, 32], breadth-first BnB search [29, 33, 34, 32], depth-first branch-and-bound (BnB) search [101, 72, 105, 103], and integer linear programming [56, 6, 25]. Of these algorithms, the current state-of-the-art among optimal algorithms include the solver GOBNILP, a branch-and-cut solver that solves a mixed-integer programming formulation of the BN structure learning problem [6, 25], and the solver ELSA, a constraint-programming based approach [103].

Local (heuristic) search algorithms for learning a BN from data have also been studied extensively including hill-climbing [19, 18, 42], tabu search [8, 5], genetic algorithms [94], and simulated annealing [84]. The local search algorithms operate over different search spaces including the space of DAGs, where an arc is added, deleted, or parents are swapped [18, 106]; the space of equivalence classes of DAGs [1]; and topological orderings over the variables [65, 100, 54, 83, 88, 66]. Recently, neural networks have also been used to solve a continuous formulation of the DAG learning problem [111, 114] but with limited success. Of note among local search algorithms, Chickering [18] proposes an algorithm called greedy equivalent search (GES) that constructs the parent set of node in the BN in a greedy fashion using a set of operators. Ordering-based local search algorithms are also an efficient and accurate approach for structure learning [100, 88, 66].

Hybrid approaches to learning the structure of a BN combine techniques from the constraint-based and the score-and-search approaches, first using statistical tests to identify a reduced set of candidate parents for all nodes, and then using score-and-search on the reduced search space. Algorithms using hybrid learning include the Max-Min Hill

Climbing algorithm (MMHC) [104]. The MMHC algorithm first employs statistical conditional independence tests to find the skeleton of the BN, similar to the constraint-based PC algorithm; second, it then orients the skeleton using a greedy hill-climbing search. Scutari et al. [91] show in an extensive empirical evaluation that a score-and-search approach based on tabu search is both faster and more accurate overall than constraint-based and hybrid-based approaches.

1.1.2 Model Averaging

The learning algorithms discussed thus far all learn a single DAG from a dataset. When one is using BNs for knowledge discovery with limited data, learning a single model may be misleading as there may be several BNs with scores that are very close to optimal and the posterior probability of even the best-scoring BN is often close to zero.

An alternative to committing to a single model is to learn a set of DAGs from a single dataset and perform some form of Bayesian or frequentist model averaging [23, 53, 61]. In the context of knowledge discovery, Bayesian model averaging allows one to estimate, for example, the posterior probability that an edge is present, rather than just knowing whether the edge is present in the best-scoring network. A widely used data analysis methodology is to: (i) learn a set of networks from the data, (ii) perform model averaging to obtain a confidence measure for each edge, and (iii) select a threshold and retain all edges with confidence higher than the threshold. In this manner, a representative network can be constructed from the edges that are deemed significant. Previous work has proposed Bayesian and frequentist model averaging approaches to network structure learning including sampling [50, 71], enumerating [60] from the space of all possible DAGs, considering the space of all DAGs consistent with a given ordering of the random variables [12, 28], considering the space of tree-structured or other restricted DAGs [71, 75], and considering only the k-best scoring DAGs for some given value of k [13, 14, 15, 16, 50, 102]. Unfortunately, these existing approaches either severely constrain the structure of the Bayesian network, such as considering a single ordering or by allowing just tree-structured networks, or have only been shown to scale to small Bayesian networks with fewer than 30 random variables.

In addition to the above algorithms, any one of the local (heuristic) search algorithms for learning a BN would be fast enough that it could be turned into a model averaging approach using a technique called bootstrapping [40]. In bootstrapping, one samples the original dataset with replacement and then learns a network using the re-sampled data. This process is repeated many times to attain a set of DAGs that can then be model averaged.

1.1.3 Incorporating Local Structure

The learning algorithms discussed thus far all learn a Bayesian network from a dataset by first learning the DAG and then learning the conditional probability distributions (CPDs) associated with the DAG. The algorithms most often assume simple conditional probability tables are used to represent the CPDs for each variable. A conditional probability table (CPT) contains a row for each possible combination of values the variables in its parent set can take; i.e., the CPT is exponential in the size of the domains of the parent variables. While learning the DAG is NP-hard [20], learning the CPTs given the DAG is straightforward and can be accomplished efficiently. However, in settings with limited data or larger parent sets, using CPTs can result in over-fitting and lead to unreliable parameters. This observation has led to proposals for representing CPDs more succinctly by capturing local structure using decision trees [12], logistic regression [12], linear combinations of Boolean functions [115], non-impeding noisy-AND trees [110], and noisy-OR relations [80, 12, 77, 78, 113, 108].

In more recent work that learns a representation of the CPDs given the DAG, Bengio and Bengio [7] propose neural networks. They focused on discrete random variables, and model CPDs with neural networks with a single dense hidden layer, where the discrete inputs and output were encoded using one-hot/unary encodings, and the output layer uses a softmax. Shen et al. [92] show experimentally that a multi-layer perceptron model consistently estimates a CPD more accurately than a decision tree model.

There have also been a limited number of proposals for simultaneously learning the DAG and the local structure of the CPDs, rather than first learning the DAG and then learning the CPDs. Buntine [12] was the first to propose incorporating local structures in BN learning within the score-and-search approach but did not provide any empirical or theoretical evidence for the benefits. The absence of an edge in the DAG of a BN represents a conditional independence relation; i.e., an independence relation that holds for all possible values of the variables. Boutilier et al. [9] formalized the concept of *context-specific conditional independence* that only holds given a certain assignment of a subset of the variables. While such an independence relation cannot be captured by the underlying DAG, they show that the associated CPD can be more succinctly represented using decision trees rather than as a CPT. Friedman and Goldszmidt [38, 39] provide algorithms, theoretical foundations, and an empirical evaluation of using decision trees as structured representations in a score-and-search approach for discrete data. They show that simultaneously learning the DAG and the CPDs represented as decision trees can improve the overall Bayesian network as measured by inference error against ground truth. Chickering et al. [19] extend the score-and-search approach to simultaneously learn decision trees with equality constraints for succinctly representing common sub-trees. However, the partitioning algorithms used to split the feature space become computationally infeasible for discrete variables with large domains.

In more recent work that simultaneously learns the DAG and the local structure of the CPDs, Talvitie et al. [99] introduce a score that is based on classification and regression trees (CARTs) to find the optimal BN in a score-and-search approach that handles datasets with mixed discrete and continuous variables. Their experiments with structured CPDs show an improvement over a tabular representation in the ability of the search algorithm to find correct BNs with less data on some real-world datasets. Pensar et al. [82] introduce partial conditional independence (PCI) as a generalization of context-specific independence and introduce PCI-trees as a generalization of decision trees, where an edge in the tree may now represent several outcomes. They propose greedy local search algorithms for both structure learning and for finding a PCI-tree representation. An empirical comparison confirms the results of Friedman and Goldszmidt [38, 39] that simultaneously learning decision trees improves over CPTs as measured by inference error. The empirical results for PCI-trees versus decision trees are mixed, with no representation having a clear advantage.

1.2 Contributions

This thesis focuses on improving the accuracy of the score-and-search approach for BN structure learning and in scaling the approach to datasets with larger numbers of random variables.

- We introduce a novel model averaging approach to BN structure learning motivated by performance guarantees in approximation algorithms. Our approach considers optimal and all near-optimal networks for model averaging. We provide pruning rules generalized to be used within this model averaging framework, enabling our approach to scale to BNs significantly larger than the current state of the art (Chapter 3).

- We extend our novel score-and-search approach with model averaging to simultaneously learn the DAG and the local structure of the CPDs in the form of a noisy-OR representation. Our approach is able to choose between CPTs and noisy-OR representations automatically. We provide an effective gradient descent algorithm to score a candidate noisy-OR using the widely used BIC score and we provide pruning rules that allow the search to successfully scale to medium-sized networks. Our empirical results provide evidence for the success of our approach to learning BNs that incorporate noisy-OR relations (Chapter 4).

- We extend our score-and-search approach with model averaging to simultaneously learn the DAG and the local structure of the CPD using neural networks representations. The use of neural networks allows modeling of high-order interactions without needing an exponential number of parameters as in the CPT. Our approach compares favourably with approaches like decision trees, and performs well in instances with low amounts of data (Chapter 5).

- We introduce a novel score-and-search approach to simultaneously learn a single DAG and model linear and non-linear local probabilistic relationships between variables. We achieve this by a combination of feature selection to reduce the search space for local relationships and extending the score-and-search approach to incorporate modeling the CPDs over variables as multivariate adaptive regression splines (MARS). MARS are polynomial regression models represented as piecewise spline functions. We show on a set of discrete and continuous benchmark instances that our proposed approach can improve the accuracy of the learned graph while scaling to instances with over 1,000 variables. Our method is fast enough that it can be used in a bootstrapping model averaging approach in many instances (Chapter 6).

Chapter 2

Background

In this chapter, we review the necessary background in Bayesian network structure learning and the performance evaluation metrics used in this thesis. For more background on these topics see, for example, Darwiche [27], Koller and Friedman [61], and Witten et al. [109].

2.1 Bayesian Networks

A Bayesian network (BN) is a probabilistic graphical model that consists of a labeled directed acyclic graph (DAG), $G = (V, E)$ in which the nodes $V = \{V_1, \ldots, V_n\}$ correspond to random variables, the edges E represent direct influence of one random variable on another, and each node V_i is labeled with a conditional probability distribution (CPD) $P(V_i \mid \Pi_i)$ that specifies the dependence of the variable V_i on its set of parents Π_i in the DAG. A BN can alternatively be viewed as a factorized representation of the joint probability distribution over the random variables and as an encoding of conditional independence assertions, where the absence of an edge in a BN indicates a conditional or unconditional independence assertion.

Definition 2.1 (Conditional and unconditional independence). *Let A, B, and C be random variables. A is conditionally independent of B given C if $P(A \mid B, C) = P(A \mid C)$ holds for all values that the variables can take from their associated domains. A and B are unconditionally independent if $P(A \mid B) = P(A)$ holds.*

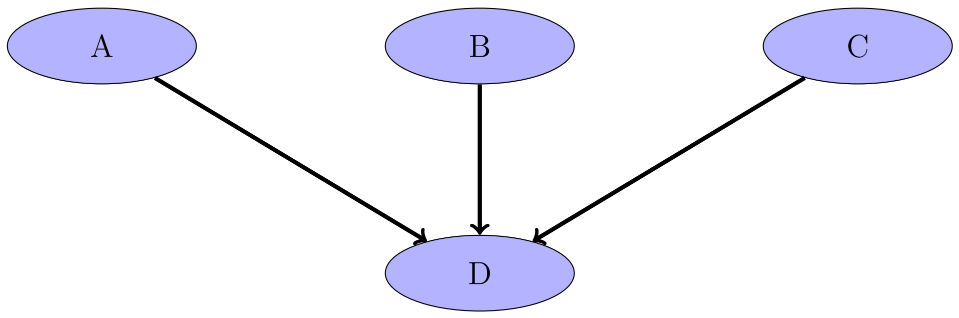
For the discrete case, each random variable V_i has domain $\Omega_i = \{v_{i1}, \ldots, v_{ir_i}\}$, where r_i is the cardinality of Ω_i and typically $r_i \geq 2$. Each set of parents Π_i has set of instantiations

$\Omega_{\Pi_i} = \{\pi_{i1}, \ldots, \pi_{ir_{\Pi_i}}\}$, where r_{Π_i} is the number of possible instantiations of the parent set Π_i of V_i. The set $\theta = \{\theta_{ijk}\}$ for all $i = \{1, \ldots, n\}, j = \{1, \ldots, r_{\Pi_i}\}$ and $k = \{1, \ldots, r_i\}$ represents parameter estimates in G obtained either from expert knowledge or from a dataset, where each θ_{ijk} estimates the conditional probability $P(V_i = v_{ik} \mid \Pi_i = \pi_{ij})$.

$$P(A=0) = \ldots \quad P(A=1) = \ldots$$

$$P(B=0) = \ldots \quad P(B=1) = \ldots \quad P(B=2) = \ldots$$

$$P(C=0) = \ldots \quad P(C=1) = \ldots$$



$$\begin{array}{ll}
P(D=0 \mid A=0,B=0,C=0) = \ldots & P(D=0 \mid A=1,B=0,C=0) = \ldots \\
P(D=0 \mid A=0,B=0,C=1) = \ldots & P(D=0 \mid A=1,B=0,C=1) = \ldots \\
P(D=0 \mid A=0,B=1,C=0) = \ldots & P(D=0 \mid A=1,B=1,C=0) = \ldots \\
P(D=0 \mid A=0,B=1,C=1) = \ldots & P(D=0 \mid A=1,B=1,C=1) = \ldots \\
P(D=0 \mid A=0,B=2,C=0) = \ldots & P(D=0 \mid A=1,B=2,C=0) = \ldots \\
P(D=0 \mid A=0,B=2,C=1) = \ldots & P(D=0 \mid A=1,B=2,C=1) = \ldots
\end{array}$$

Figure 2.1: A Bayesian network with four variables. Note that $P(D = 1 \mid A, B, C) = 1 - P(D = 0 \mid A, B, C)$ for all values of A, B, and C.

Example 2.1. *Consider the example Bayesian network over four random variables shown in Figure 2.1. The domains are $\Omega_A, \Omega_C, \Omega_D = \{0, 1\}$ with $r_A, r_c, r_D = 2$ and $\Omega_B = \{0, 1, 2\}$ with $r_B = 3$. The parent set of variable D is $\{A, B, C\}$ with $r_{\{A,B,C\}} = 12$. The absence of an edge from B to A indicates that $P(A \mid B) = P(A)$ holds.*

2.2 Bayesian Network Structure Learning

Let G be a DAG over random variables V and let $I = \{I_1, \ldots, I_N\}$ be a dataset, where each instance I_i is an n-tuple that is a complete instantiation of the variables in V. A *scoring*

A	B	C	D
0	1	1	1
1	0	1	1
0	2	0	0
1	2	0	0
0	0	1	0
0	1	0	0
1	0	0	1
...	...	...	...

Figure 2.2: A sample dataset with four random variables.

function $\sigma(G \mid I)$ assigns a real value measuring the quality of G given the dataset I. Without loss of generality, we assume that a lower score represents a better quality network structure. To simplify notation, we use $\sigma(G)$ in place of $\sigma(G \mid I)$ when the dataset is clear from context. We focus on decomposable scoring functions, where $\sigma(G) = \sum_i^n \sigma(\Pi_i)$ and Π_i is the parent set of variable V_i in the DAG G.

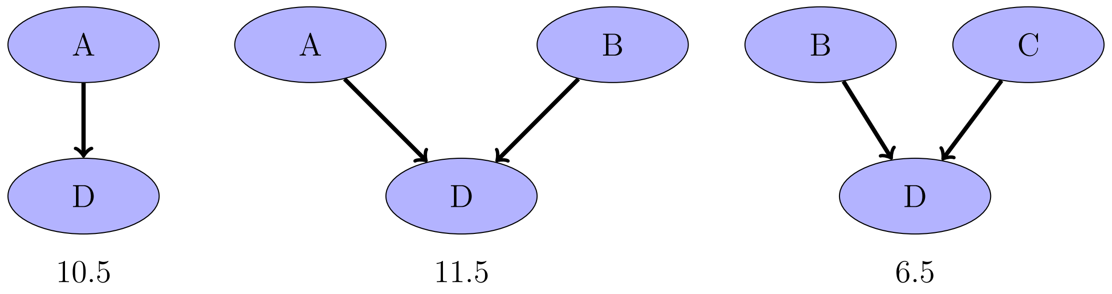


Figure 2.3: Candidate parent sets for variable D with the associated scores.

Example 2.2. *Figure 2.2 shows an example dataset over four random variables. Consider the variable D. There are eight candidate parent sets for D: the empty parent set, the singleton parent sets A, B, and C, and so on. Three of the possible candidate parent sets and their associated scores for variable D are shown in Figure 2.3.*

The Bayesian network structure learning problem is defined as follows.

Definition 2.2 (Bayesian network structure learning)**.** *Given a dataset $I = \{I_1, \ldots, I_N\}$ over random variables $V = \{V_1, \ldots, V_n\}$ and a scoring function σ, the Bayesian network structure learning (BNSL) problem is to find the directed acyclic graph $G^* := \text{argmin}_G\, \sigma(G)$.*

The predominant method for Bayesian network structure learning (BNSL) from data is the *score-and-search* method. In score-and-search, a scoring function is used to indicate how well a network fits the data and a search algorithm is used to find a DAG which achieves the best possible score by choosing a parent set for each variable. Finding such a network is known to be NP-hard [21]. Here we give the overall score-and-search algorithm for BNSL given a dataset I and a scoring function σ.

- **Step 1 (Score).** For each variable V_i, $i \in 1, \ldots, n$, determine the σ scores for all candidate parent sets that could not be pruned with pruning rules for σ.

  Step 2 (Search). The scores obtained in Step 1 are used to learn the optimal scoring network; i.e., the network with the smallest sum of scores of the parent sets $\text{argmin}_G \sum_i^n \sigma(\Pi_i)$.

Pruning techniques that preserve optimality can be used to reduce the number of candidate parent sets that need to be considered, but in the worst-case the number of candidate parent sets for each variable V_i is exponential in n, where n is the number of vertices in the DAG. Another approach to reduce the number of candidate parent sets is to limit the maximum in-degree of each node to a small constant.

2.2.1 Scoring Functions

Scoring functions usually balance goodness of fit to the data with a penalty term for model complexity to avoid overfitting. Common scoring functions include AIC [2], BIC/MDL [89, 64, 87], BDeu [12, 51] qBDJ [98], and qNML [96] for the discrete case, and BGe [43] for the continuous case. An important property of these (and most) scoring functions is decomposability, where the score of the entire network $\sigma(G)$ can be rewritten as the sum of local scores associated to each vertex $\sum_{i=1}^n \sigma(V_i, \Pi_i)$ that only depends on V_i and its parent set Π_i in G. The local score is abbreviated below as $\sigma(\Pi_i)$ when the local node V_i is clear from context.

In this work, we focus on the Bayesian Information Criterion (BIC) score and the Bayesian Dirichlet score, specifically BDeu. In what follows, the parameter θ_{ijk} is an estimate of the conditional probability $P(V_i = v_{ik} \mid \Pi_i = \pi_{ij})$, n_{ijk} is the number of instances in a dataset I where v_{ik} and π_{ij} co-occur, and n_{ij} is the number of instances in I where π_{ij} occurs.

The BIC scoring function is defined as,

$$\sigma_{BIC}(G) = -\max_{\theta} L_{G,I}(\theta) + t(G) \cdot w = -\sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} n_{ijk} \log \frac{n_{ijk}}{n_{ij}} + \sum_{i=1}^{n} r_{\Pi_i}(r_i - 1)\frac{\log N}{2}.$$

Here, $w = \frac{\log N}{2}$, $t(G)$ is a penalty term, and $L_{G,I}(\theta)$ is the log likelihood given by,

$$L_{G,I}(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} \log \theta_{ijk}^{n_{ijk}}.$$

As the BIC function is decomposable, we can associate a score to Π_i, a candidate parent set of V_i as follows,

$$\sigma_{BIC}(\Pi_i) = -\max_{\theta_i} L(\theta_i) + t(\Pi_i) \cdot w = -\sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} n_{ijk} \log \frac{n_{ijk}}{n_{ij}} + r_{\Pi_i}(r_i - 1)\frac{\log N}{2}. \quad (2.1)$$

Here, $L(\theta_i) = \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} \log \theta_{ijk}^{n_{ijk}}$ and $t(\Pi_i) = r_{\Pi_i}(r_i - 1)$.

The BDeu scoring function is defined as,

$$\sigma_{BDeu}(G) = -\sum_{i=1}^{n} \left(\sum_{j=1}^{r_{\Pi_i}} \left(\log \frac{\Gamma\left(\frac{\alpha}{r_{\Pi_i}}\right)}{\Gamma\left(\frac{\alpha}{r_{\Pi_i}} + n_{ij}\right)} + \sum_{k=1}^{r_i} \log \frac{\Gamma\left(\frac{\alpha}{r_i r_{\Pi_i}} + n_{ijk}\right)}{\Gamma\left(\frac{\alpha}{r_i r_{\Pi_i}}\right)} \right) \right),$$

where Γ is the Gamma fucntion, α is called the equivalent sample size, and $n_{ij} = \sum_k n_{ijk}$. As the BDeu function is decomposable, we can associate a score to Π_i, a candidate parent set of V_i as follows,

$$\sigma_{BDeu}(\Pi_i) = -\sum_{j=1}^{r_{\Pi_i}} \left(\log \frac{\Gamma\left(\frac{\alpha}{r_{\Pi_i}}\right)}{\Gamma\left(\frac{\alpha}{r_{\Pi_i}} + n_{ij}\right)} + \sum_{k=1}^{r_i} \log \frac{\Gamma\left(\frac{\alpha}{r_i r_{\Pi_i}} + n_{ijk}\right)}{\Gamma\left(\frac{\alpha}{r_i r_{\Pi_i}}\right)} \right). \quad (2.2)$$

2.3 Model Averaging

The objective of BNSL is finding a single best scoring BN. This has been often found to be problematic. Insufficient data can lead to many networks with scores close to optimal but with small posterior probabilities, and selecting a single model with the best score can thus

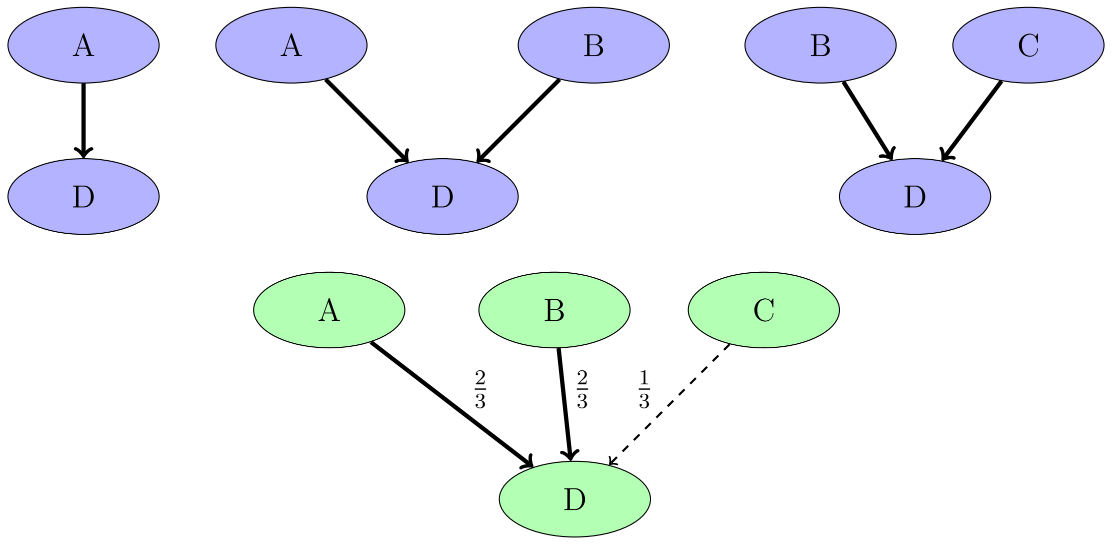


Figure 2.4: Model averaging example with learned networks with confidence values in blue, and the model averaged network in green. Using a threshold of $t = 0.5$, edge (C, D) is not considered significant and is removed from the final averaged network.

be misleading and inaccurate for prediction and knowledge discovery. Instead of learning one best-scoring network, an alternative method is some form of Bayesian or frequentist model averaging [23, 53, 61], where we enumerate or sample from the space of possible BNs. This has advantages in knowledge discovery where Bayesian model averaging can allow estimation of measures like the posterior probability that an edge is present.

We can use model averaging in structure learning for Bayesian networks in the following way: (i) learn a set of networks with good scores that fit the given data, (ii) perform model averaging over the set of networks and compute a real-valued confidence value $P(e \mid I)$ for each edge e in the learned set of networks , and (iii) set a minimum threshold value t and report all significant edges, which are edges e such that their confidence values exceed the threshold $P(e \mid I) > t$. In this manner, a model-averaged network can be constructed from the significant edges, and used to analyze causal and probabilistic dependencies.

Example 2.3. *Figure 2.4 shows the learned networks with confidence values in blue, and the model averaged network in green. As the threshold is set to $t = 0.5$, edge (C, D) is not significant and is removed from the final averaged network.*

One popular method to compute the confidence values uses bootstrapping, which is the process of random sampling with replacement. Bootstrapping assigns measures of accuracy (or any performance measure or statistic) to sample estimates. In the context of structure learning in Bayesian networks Friedman et al. [40] proposed bootstrapping with thresholds to determine the existence of edges and other features. In particular, the non-parametric approach samples the original dataset with replacement and then learns a network using the re-sampled data. After performing the resampling procedure many times, we can get the empirical probabilities of all edges by averaging on the learned networks. Finally, a threshold is applied to get the averaged network.

Another method to compute the confidence values is the k-best approach, which considers only the k best scoring DAGs for some given value of k [13, 14, 15, 16, 50, 102], though these methods suffer from a rigorous method to set k and do not scale to larger numbers of variables. In Chapter 3 we propose a novel method that compares favourably to bootstrapping in terms of accuracy and to k-best in terms of scaling.

2.4 Representing CPDs

Implementations of score-and-search usually employ simple conditional probability tables (CPTs) to represent the CPD; i.e., the function $P(V_i \mid \Pi_i)$ over discrete variables (e.g., [6, 18, 105]). CPTs are tables that contain an entry for each possible combination of values for V_i and Π_i. In settings with a large number of variables, or with variables having large domains, such a representation might not be efficient as CPTs grow exponentially with the size of the parent set. Additionally, CPTs can ignore underlying structure within CPDs and result in overfitting. These two disadvantages have led to research of structured representations for CPDs that can model complex relationships between a child and its parents without exponential size and overfitting risks. We discuss two alternative CPD representations used in Bayesian networks: noisy-OR and decision trees.

2.4.1 Noisy-OR

An alternative to tabular CPTs is using a deterministic function to represent the relationship between a variable and its parents. One such model is the noisy-OR [45, 80]. This models the CPD over causes (parents) and effects (children). The noisy-OR assumes a form of causal independence (CI) and allows one to specify a CPT with just n parameters instead of 2^{n+1}.

With the noisy-OR relation one assumes that there are a set of causes $\Pi_i := \{V_{i1}, \ldots, V_{i|\Pi_i|}\}$ leading to an effect V_i, where $V_i, V_{ij} \in V$ for all $j \in \{1, ...|\Pi_i|\}$ and $V_i \notin \Pi_i$ (see

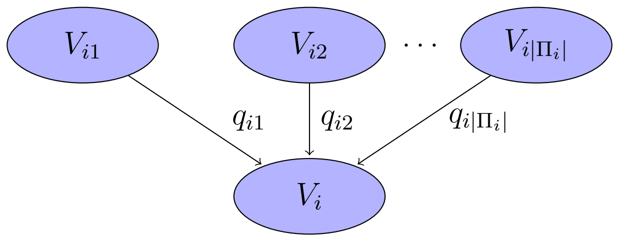


Figure 2.5: Causal structure for a Bayesian network with a noisy-OR relation, where the set of causes $\Pi_i := \{V_{i1}, \ldots, V_{i|\Pi_i|}\}$ leads to effect V_i and there is a noisy-OR relation at node V_i.

Figure 2.5). Each cause $V_{ij} \in \Pi_i$ is either present or absent, and each V_{ij} in isolation is likely to cause V_i and the likelihood is not diminished if more than one cause is present. Further, one assumes that all possible causes are given and when all causes are absent, the effect is absent. Finally, one assumes that the mechanism or reason that inhibits a V_{ij} from causing V_i is independent of the mechanism or reason that inhibits a $V_{ij'}$, $j' \neq j$, from causing V_i.

For a node V_i and parent set Π_i, a noisy-OR relation specifies a CPT using $|\Pi_i|$ parameters, $\mathbf{q}_i = q_{i1}, \ldots, q_{i|\Pi_i|}$, one for each parent, where q_{ij} is the probability that V_i is false given that V_{ij} is true and all of the other parents are false,

$$P(V_i = 0 \mid V_{ij} = 1, V_{ij'} = 0_{[\forall j', j' \neq j]}) = q_{ij}.$$

From these parameters, the full CPT representation of size 2^{n+1} can be generated using,

$$\phi_{ij0} = \begin{cases} \prod_{j \in T_x} q_{ij} & \text{if } T_x \neq \{\} \\ 1 & \text{otherwise,} \end{cases} \tag{2.3}$$

where $T_x = \{j \mid V_{ij} = 1\}$. The last condition (when $T_x = \{\}$) corresponds to the assumptions that all possible causes are given and that when all causes are absent, the effect is absent; i.e., $P(V_i = 0 \mid V_{i1} = 0, \ldots, V_{i|\Pi_i|} = 0) = 1$. Of course, $\phi_{ij1} = 1 - \phi_{ij0}$. The set $\phi_i := \left\{\phi_{ijk} \mid j \in \{1, \ldots, r_{\Pi_i}\}, k \in \{0, 1\}\right\}$ is referred to as the *noisy-OR CPT* of node V_i.

The above assumptions are not as restrictive as may first appear. One can always introduce an additional random variable V_{i0} that is a parent of V_i but itself has no parents. The variable V_{i0} represents all of the other reasons that could cause V_i to occur. The node V_{i0} and the prior probability $P(V_{i0})$ are referred to as a *leak node* and the *leak probability* [52], respectively. In this work we assume that all the causes are known.

2.4.2 Decision Trees

Boutilier et al. [9] formalized the concept of context-specific conditional independence that only holds given a certain assignment of a subset of the variables.

Definition 2.3 (Context-specific conditional independence)**.** *Let A, B, and C be random variables. A is context-specific conditionally independent of B given $C = c$ if $P(A \mid B, C = c) = P(A \mid C = c)$ holds for all values that the variables A and B can take from their associated domains.*

While such an independence relation cannot be captured by the underlying DAG, a natural representation for capturing common elements in such a CPD is via a decision tree, where the leaves of the tree represent different possible conditional distributions over a variable V_i, and where the path to each leaf dictates the context in which this distribution is used.

A	B	C	D	$P(D \mid A, B, C)$
0	0	0	0	0.8
0	0	0	1	0.2
0	0	1	0	0.8
0	0	1	1	0.2
0	1	0	0	0.8
0	1	0	1	0.2
0	1	1	0	0.8
0	1	1	1	0.2
1	0	0	0	0.9
1	0	0	1	0.1
1	0	1	0	0.4
1	0	1	1	0.6
1	1	0	0	0.1
1	1	0	1	0.9
1	1	1	0	0.1
1	1	1	1	0.9

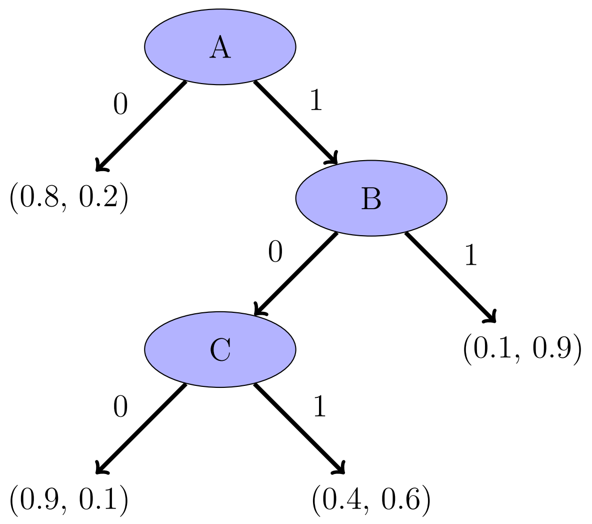


Figure 2.6: Two representations of the same CPD $P(D \mid A, B, C)$ for a child node D and a parent set $\{A, B, C\}$: tabular CPT (left) and decision tree representation (right).

A decision tree representation of a CPD for variable V_i is a rooted tree; each tree-node in the tree is either a leaf tree-node or an interior tree-node. Each leaf is labeled with a distribution $P(V_i)$. Each interior tree-node is labeled with some variable $V_j \in \Pi_i, j \neq i$. Each interior tree-node has a set of outgoing arcs to its children, each one associated with a unique variable assignment $V_j = v_{jl}$ for $j \neq i$ and $v_{jl} \in \Omega_j, l \in [1, \ldots, r_j]$. A branch through a decision tree is a path beginning at the root and proceeding to a leaf node. We assume that no branch contains two interior nodes labeled by the same variable. The parent context induced by a branch is the set of variable assignments $V_j = v_{jl}$ encountered on the arcs along the branch.

Example 2.4. *Figure 2.6 shows a comparison of a CPD represented by a CPT and a decision tree. The branch $A = 0$ in the decision tree encodes the context-specific independence relation $P(D \mid A = 0, B, C) = P(D \mid A = 0)$. The branch $A = 1, B = 1$ in the decision tree encodes the context-specific independence relation $P(D \mid A = 1, B = 1, C) = P(D \mid A = 0, B = 1)$.*

2.5 Performance Evaluation Metrics

As previously noted, our interest in this thesis is in BNs as a knowledge discovery or data analysis tool, where the BN is learned automatically from data and the resulting BN is then studied for the insights that it provides on the domain such as possible cause-effect relationships, probabilistic dependencies, and conditional independence relationships. In this section, we discuss the performance metrics we use to evaluate our proposals to show that our proposals improve the state of the art.

2.5.1 DAG Performance Metrics: Skeleton

We first define performance metrics over the skeleton of the learned DAG, where we consider the edges undirected.

Definition 2.4 (Skeleton of a DAG)**.** *The skeleton of a DAG G is the undirected graph with the same vertices and edges as G, regardless of their directions.*

Let G^* be the skeleton of the ground truth DAG—i.e., the correct answer—and G be the skeleton of the learned DAG. A true positive is the case where an edge occurs in G^* and also occurs in G, a false positive is the case where an edge is missing from G^* but occurs in G, and a false negative is the case where an edge occurs in G^* but is missing from G. We use two performance measures over skeletons of the learned DAG: misclassification cost (see, e.g., [68]) and F_β score (see, e.g., [4]).

Definition 2.5 (Misclassification cost)**.** *The weighted misclassification cost over the skeleton of a ground truth DAG and the skeleton of a learned DAG is defined as,* $\alpha \times FN + FP$*, where* $\alpha > 0$*, FN is the number of false negatives, and FP is the number of false positives.*

Definition 2.6 (F_β score)**.** *The* F_β *score over the skeleton of a ground truth DAG and the skeleton of a learned DAG is defined as,*

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

where $\beta > 0$*, precision is* $TP/(TP + FP)$ *and recall is* $TP/(TP + FN)$*.*

The F_β score is between 0 and 1, with 1 being the best value. Common values of α and β are $\frac{1}{4}, \frac{1}{2}, 1, 2$, and 4. When $\alpha > 1$, the misclassification cost gives more weight to FN and thus penalizes missing edges over extra edges; vice-versa when $\alpha < 1$. When $\beta > 1$, the F_β score gives more weight to recall over precision; vice-versa when $\beta < 1$. The F_β score and the weighted misclassification cost allow us to evaluate the performance on a spectrum with different tradeoffs between precision and recall and between FP and FN.

2.5.2 DAG Performance Metrics: CPDAG

We now define performance metrics over the CPDAG representation of the learned DAG. Comparing two DAGs directly is problematic as BNs fall into equivalence classes where some edges in the DAG have their direction compelled—they are oriented in that direction in all DAGs in the equivalence class—and some edges can be "flipped" and the resulting DAG is equivalent to the original DAG. The concept of CPDAG captures this intuition [104].

Definition 2.7 (Markov equivalence class (MEC))**.** *Two DAGs over a set of random variables* V *belong to the same Markov equivalence class (MEC) if and only if they encode the same set of conditional independence assertions. The set of all DAGs over* V *is partitioned into a set of mutually exclusive and exhaustive MECs.*

Definition 2.8 (Completed partially directed acyclic graph (CPDAG))**.** *All DAGs in the same MEC can be uniquely represented by a partially completed directed acyclic graph (CPDAG), which is the graph with directed and undirected edges such that an edge is directed if all DAGs in the equivalence class agree on the direction and undirected otherwise.*

Chickering [17, 18] provides a local graph transformation characterization of Markov equivalence that can be used to determine the CPDAG representation of a DAG.

Example 2.5. *Consider the BNs shown in Figure 2.7. The blue network on the left encodes the assertion that A and C, for example, are independent. In the green networks on the right, this assertion does not hold indicating that they belong to different MECs. All of the green networks belong to the same MEC as they encode the same set of conditional independence assertions:* $\{P(A \mid C, B) = P(A \mid B), P(C \mid A, B) = P(C \mid B)\}$.

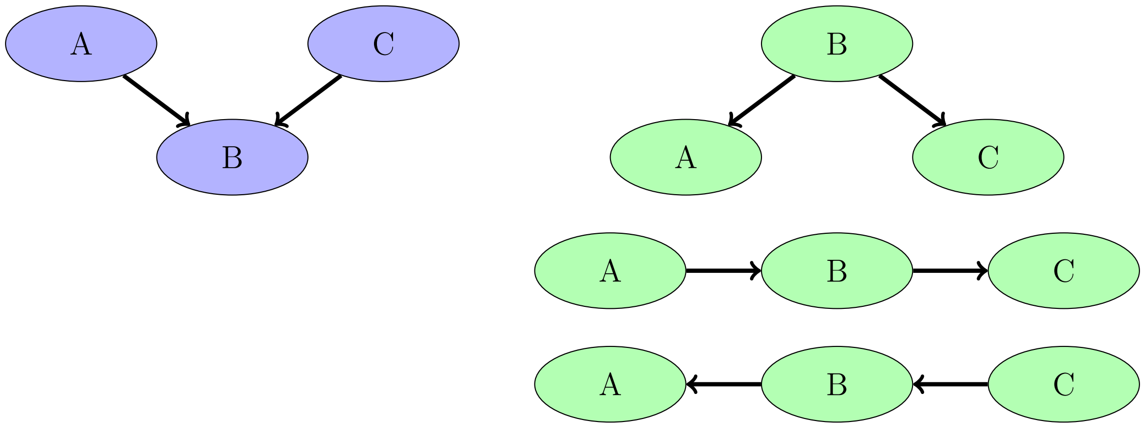


Figure 2.7: An example of Markov equivalence classes for a three node network. The blue network and the green networks are in different Markov equivalence classes.

Let G^* be the CPDAG representation of the ground truth DAG and G be the CPDAG representation of the learned DAG. A true positive is the case where an edge occurs in G^* and also occurs in G with the correct orientation (either it is undirected in both or the direction of the edge is the same in both), a false positive is the case where an edge is missing from G^* but occurs in G, a false negative is the case where an edge occurs in G^* but is missing from G, and a wrong direction is the case where an edge occurs in G^* and also occurs in G but with the incorrect orientation. We use two performance measures over CPDAG representations of the learned DAG: structural Hamming distance [104] and multi-class F_β score (see, e.g., [4]).

Definition 2.9 (Structural Hamming distance (SHD))**.** *The structural Hamming distance (SHD) over the CPDAG representation of a ground truth DAG and the CPDAG representation of a learned DAG is defined as* $FN + FP + WD$*, where* FN *is the number of false negatives, and* FP *is the number of false positives, and* WD *is the number of wrong directions.*

Although widely used in the BN literature, the SHD has some significant limitations. As each of summands for SHD are weighted equally, a low SHD does not necessarily indicate that the learned network is useful. For example, it is possible for two learned networks to have the same SHD, but one network could have learned few or none of the edges present in the ground truth, while the other network could have learned all or nearly all of the ground truth edges. Depending on the application, the former network might not be a desirable outcome, even if it has a low SHD.

The multi-class F_β score generalizes the binary F_β score to the case where the number of classes is greater than two. In the context of CPDAGS, rather than just a binary prediction as in the skeleton—the edge is either present or not present—there are now four possible class labels to predict: $\leftarrow$, $\rightarrow$, $-$, and no edge. In the multi-class F_β score, the weighted average of the binary F_β score of each class is determined, where the binary F_β scores are weighted by the number of edges in each class.

Definition 2.10 (multi-class F_β score)**.** *Let y be the set of true (edge, class-label) pairs and $\hat{y}$ be the set of predicted (edge, class-label) pairs. Let y_l and $\hat{y}_l$ be y and $\hat{y}$ restricted to class label l, respectively. The multi-class F_β score over the CPDAG representation of a ground truth DAG and the CPDAG representation of a learned DAG is defined as,*

$$\textit{multi-class } F_\beta = \frac{1}{\sum_{l \in L} |y_l|} \sum_{l \in L} |y_l| F_\beta(y_l, \hat{y}_l),$$

where $F_\beta(y_l, \hat{y}_l)$ is the binary F_β score for the class label l.

The multi-class F_β score is between 0 and 1, with 1 being the best value.

2.5.3 Density Estimation and Inference Metric

To measure the difference between two probability distributions over the same set of random variables, we adopt a widely-used measure called the Kullback-Leibler divergence.

Definition 2.11 (KL-divergence)**.** *The KL divergence between two* conditional *probability distributions, $P(A \mid B)$ and $Q(A \mid B)$ is given by,*

$$D_{KL}(P(A \mid B) \mid\mid Q(A \mid B)) = \sum_{b \in B} P(B = b) \sum_{a \in A} P(A = a \mid B = b) \log \frac{P(A = a \mid B = b)}{Q(A = a \mid B = b)}.$$

The KL-divergence is a measure of the information lost when $Q(A \mid B)$ is used to approximate $P(A \mid B)$. Note that when $P(A = a \mid B = b) \neq 0$ and $Q(A \mid B) = 0$, the

KL divergence is defined to be ∞; i.e., absolutely different. This is undesirable. Given that $Q(A \mid B)$ is approximated from data, it is not reasonable to predict that an event is impossible since the data is by necessity incomplete and the event may be rare. We thus use a smoothed version of KL-divergence where $Q(A \mid B) = 0$ is replaced by $Q(A \mid B) = \delta$, for some small $\delta > 0$, when $P(A = a \mid B = b) \neq 0$.

Chapter 3

The Credible Set Approach

When one is using BNs for knowledge discovery with limited data, learning a single model may be misleading as there may be several BNs with scores that are very close to optimal and the posterior probability of even the best-scoring BN is often close to zero. An alternative to committing to a single model is to perform some form of Bayesian or frequentist model averaging [23, 53]. In the context of knowledge discovery, Bayesian model averaging allows one to estimate, for example, the posterior probability that an edge is present, rather than just knowing whether the edge is present in the best-scoring DAG. Previous work has proposed Bayesian and frequentist model averaging approaches to BN structure learning based on score-and-search. Unfortunately, existing global (optimal) approaches for model averaging either severely restrict the structure of the BN, or have only been shown to scale to networks with fewer than 30 random variables (see the review of related work and discussion in Section 1.1). In this chapter, we propose a novel global approach to model averaging inspired by performance guarantees in approximation algorithms. Our approach only considers *credible* models in that they are optimal or near-optimal in score. We show empirically that our approach improves on the accuracy of a widely used local (heuristic) search approach and is more efficient and scales to significantly larger Bayesian networks than existing state-of-the-art global search approaches. The keys to the scalability of our approach are generalized pruning rules that retain optimality.

3.1 Overview

Let OPT be the score of an optimal BN and assume that the optimization problem that we want to approximate is to find the minimum-score BN structure. Instead of finding an optimal network or the k-best networks for some fixed value of k, we propose to find all Bayesian networks $\mathcal{G}$ that are within a factor ρ of optimal; i.e.,

$$OPT \leq score(\mathcal{G}) \leq \rho \cdot OPT,$$

for some given value of $\rho \geq 1$, or equivalently,

$$OPT \leq score(\mathcal{G}) \leq OPT + \epsilon, \tag{3.1}$$

for $\epsilon = (\rho - 1) \cdot OPT$. For the two commonly used scoring functions BIC and BDeu, it can be shown that a good choice for the value of ϵ is closely related to the Bayes factor [67], a model selection criterion summarized in [58] and discussed below.

The idea is to learn only *credible* models that have optimal or near-optimal scores. This is in contrast to approaches that enumerate or sample from the space of all feasible DAGs, which can learn networks with scores that can be far from optimal; for example, for the BIC scoring function the ratio of worst-scoring to best-scoring network can be four or five orders of magnitude[1]. A similar but more restricted concern arises with the k-best approaches since there is no *a priori* method to set the parameter k to learn only the credible networks. Importantly, our approach is significantly more scalable, allowing us to learn BNs with around 70 random variables without restricting the structure of the BN. We leverage the significant body of pruning rules from previous methods for finding a single optimal BN structure. These pruning rules remove from consideration many candidate parent sets both before and during the search and we show that many of these pruning rules can be naturally generalized to preserve the Bayesian networks that are within a factor of optimal. We modify GOBNILP [6, 26], a state-of-the-art algorithm for finding an optimal BN, to implement our generalized pruning rules and to find all credible networks. Empirically, we show that the modified GOBNILP scales to significantly larger networks without any structural constraints on the learned BNs.

The remainder of this chapter proceeds as follows.

- We formalize the concept of credible networks for a dataset; i.e., networks that are close to optimal for a given scoring function, and define the problem of learning all credible networks for model averaging purposes. We discuss how the distance from the optimum that should be considered is closely related to the Bayes factor for the BIC and BDeu scoring functions (Section 3.2).

[1]Madigan and Raftery [71] deem such models *discredited* when they make a similar argument for not considering models whose probability is greater than a factor from the most probable.

- As finding all credible networks is computationally expensive, it is important to reduce the search space as much as possible without discarding credible networks. We generalize existing pruning rules to the problem of learning all credible networks (Section 3.3).

- We describe how learning all credible networks can be used for model averaging. We provide implementation details of solving the problem of learning all credible networks with an ϵ based on the Bayes factor approach, and evaluate and compare its performance with published k-best solvers [102, 16] as well as the bootstrapping method from the widely used bnlearn library [90] (Section 3.4).

3.2 Credible Sets

In this section, we formalize the concept of a credible network—a network that is optimal or near-optimal—and define the problem of learning all credible networks, a problem we refer to as the ϵ-Bayesian Network Structure Learning problem (ϵBNSL). Furthermore, we discuss what a good value for ϵ is in order to use credible networks for model averaging purposes.

We begin by defining the concept of a credible network, given a dataset and a scoring function.

Definition 3.1 (Credible network)**.** *Given a dataset $I = \{I_1, \ldots, I_N\}$, scoring function σ, and a constant $\epsilon \geq 0$, a credible network G is a network that has a score $\sigma(G)$ such that $OPT \leq \sigma(G) \leq OPT + \epsilon$, where OPT is the score of an optimal Bayesian network. A network that is not credible is discredited.*

We are interested in finding the set of all credible networks for a dataset, in order to use model averaging on the set.

Definition 3.2 (ϵBNSL)**.** *For a dataset $I = \{I_1, \ldots, I_N\}$, a scoring function σ, and a constant $\epsilon \geq 0$, the ϵ-Bayesian Network Structure Learning (ϵBNSL) problem is to find all credible networks.*

The problem of finding all optimal networks is the special case of ϵBNSL with $\epsilon = 0$.

It is important to use a good value for ϵ in the ϵBNSL problem. Liao et al. [67] showed that the Bayes factor (BF), a model selection criterion proposed by Jeffreys [57] as an alternative to significance tests, can guide the choice of ϵ. The Bayes factor was rigorously evaluated and examined by Kass and Raftery [58] as a model selection tool, and can be defined in our context as follows.

Definition 3.3 (Bayes factor)**.** *Let G_0 and G_1 be DAGs of BNs that are defined over the same set of random variables V and let $I = \{I_1, \ldots, I_N\}$ be a dataset. The Bayes factor $BF(G_0, G_1)$ is defined by,*

$$BF(G_0, G_1) = \frac{P(I \mid G_0)}{P(I \mid G_1)}.$$

Thus, the Bayes factor $BF(G_0, G_1)$ is the quotient of the probability of the data predicted by network G_0 and G_1, respectively. Using Bayes' theorem, the posteriors of the DAGs can be obtained by,

$$\frac{P(G_0 \mid I)}{P(G_1 \mid I)} = \frac{P(I \mid G_0)}{P(I \mid G_1)} \cdot \frac{P(G_0)}{P(G_1)} = BF(G_0, G_1) \cdot \frac{P(G_0)}{P(G_1)}.$$

Assuming that the prior over DAGs is uniform (a typical assumption), the BF can then be used to directly assess the ratio of the structure posteriors.

Next, we show how the Bayes factor relates to comparisons of two networks via BIC or BDeu scores.

BIC score. The difference of the BIC scores for the DAGs G_0 and G_1, both over the same set of variables, can be used as a rough approximation to $\log BF$. The difference of two BIC scores converges for $N \to \infty$ against the true value of $\log BF$,

$$\frac{\sigma_{BIC}(G_1) - \sigma_{BIC}(G_0) - \log BF(G_0, G_1)}{\log BF(G_0, G_1)} \to 0,$$

as described by Kass and Raftery [58]. To see this, consider the following. The logarithm of the marginal likelihood for a network G can be approximated as follows [86] (a similar derivation can be found in [23]),

$$\log P(I \mid G) = L_{G,I}(\hat{\theta}) - t(G) \cdot \frac{\log N}{2} + t(G) \cdot \frac{\log 2\pi}{2} - \frac{1}{2} \log |J_{G,I}(\hat{\theta})| + \log P(\hat{\theta} \mid G),$$

with $J_{G,I}(\hat{\theta})$ being the evaluation of the Hessian matrix at the maximum likelihood estimate $\hat{\theta}$. It follows that,

$$\log P(I \mid G) = -\sigma_{BIC}(G) + O(1),$$

showing that the BIC score was designed to approximate the log marginal likelihood. Dropping the $O(1)$ term yields,

$$\sigma_{BIC}(G_1) - \sigma_{BIC}(G_0) = \log \frac{P(I \mid G_0)}{P(I \mid G_1)} = \log BF(G_0, G_1).$$

Some papers define the BIC score to be twice as large as the BIC defined here, but this does not affect the above relationship.

BDeu score. The difference of the BDeu scores for the DAGs G_0 and G_1, both over the same set of variables, can be expressed in terms of the logarithm of the Bayes factor as well. The BDeu score is the log marginal likelihood where there are Dirichlet distributions over the parameters [12, 51],

$$\log P(I \mid G) = -\sigma_{BDeu}(G).$$

It follows that,

$$\sigma_{BDeu}(G_1) - \sigma_{BDeu}(G_0) = \log \frac{P(I \mid G_0)}{P(I \mid G_1)} = \log BF(G_0, G_1).$$

The observations on BIC and BDeu scores are consistent with the results of Kass and Raftery [58], who note that the logarithm of the Bayes factor $\log BF$ can be seen as a measure for the *relative success* of two models at predicting data. This is sometimes referred to as "weight of evidence" without the assumption that either model is true.

However, how far from the optimal model we should consider networks is often dependent on the study and subject in question, and thus requires domain knowledge; e.g., a BF of 1,000 is more appropriate in forensic science. Heckerman et al. [51] proposed the following scale to interpret BF values: a BF of 1 to 3 bears only anecdotal evidence that G_0 is better, a BF of 3 to 20 suggests some positive evidence in favor of G_0 being better, a BF of 20 to 150 suggests strong evidence in favor of G_0, and a BF greater than 150 indicates very strong evidence. Let G^* be an optimal network. If we choose 150 to be the desired BF in ϵBNSL, e.g., $G_0 = G^*$ and $\epsilon = \log(150)$, then any network with a score less than or equal to $\log(150)$ away from the optimal score would be credible; otherwise it would be discredited. Note that the ratio of posterior probabilities was defined as λ in [102, 16] and was used as a metric to assess arbitrary values of k in finding the k-best networks.

Finally, given a desired Bayes factor, we can rewrite the score criterion of the ϵBNSL problem using the BIC or BDeu scoring function as,

$$OPT \leq \mathit{score}(\mathcal{G}) \leq OPT + \log BF. \tag{3.2}$$

3.3 Pruning Rules

To find all optimal and near-optimal BNs given a BF, the local score $\sigma(\Pi_i)$ for each candidate parent set $\Pi_i \subseteq 2^{V-\{V_i\}}$ and each random variable V_i must be considered. As this is very cost prohibitive, it is important that the search space of candidate parent sets be pruned, provided that global optimality constraints are not violated. In this section, we generalize existing pruning rules from the literature such that the generalized rules hold when solving the ϵBNSL problem.

Without loss of generality, we assume in stating the rules and the generalized rules that a lower score represents a better-quality network structure. A subtle but important point is that the statements of the pruning rules in the literature sometimes assume that one wishes to only preserve *an* optimal network and other times that one wishes to preserve *all* optimal networks. We state the pruning rules such that all optimal networks are preserved. In this way, when the pruning rules are generalized to ϵBNSL, the original rule and the generalized rule coincide when $\epsilon = 0$.

Definition 3.4 (Safely pruned). *Given a vertex variable V_i and a non-negative constant $\epsilon \in \mathbb{R}^+$, a candidate parent set Π_i for V_i can be safely pruned if Π_i cannot be the parent set of V_i in any network in the set of credible networks.*

We discuss the original rules and their generalization below along with proofs for each of the results.

Teyssier and Koller [100] give a pruning rule for all decomposable scoring functions. This rule compares the score of a candidate parent set to those of its subsets.

Theorem 3.1 (Teyssier and Koller [100]). *Given a vertex variable V_j, and candidate parent sets Π_j and Π'_j, if $\Pi_j \subset \Pi'_j$ and $\sigma(\Pi_j) < \sigma(\Pi'_j)$, Π'_j can be safely pruned if σ is a decomposable scoring function.*

We relax this pruning rule below.

Lemma 3.1. *Given a vertex variable V_j, candidate parent sets Π_j and Π'_j, and some $\epsilon \in \mathbb{R}^+$, if $\Pi_j \subset \Pi'_j$ and $\sigma(\Pi_j) + \epsilon < \sigma(\Pi'_j)$, Π'_j can be safely pruned if σ is a decomposable scoring function.*

Proof. (Lemma 3.1) Consider networks G and G' that are the same except for the parent set of V_j, where G has the parent set Π_j for V_j and G' has the parent set Π'_j for V_j.

$$\begin{aligned}
\sigma(G') &= \sigma(\Pi'_j) + \sum_{i \neq j} \sigma(\Pi_j) && [\sigma() \text{ is decomposable}] \\
&> \sigma(\Pi_j) + \epsilon + \sum_{i \neq j} \sigma(\Pi_j) && [\text{given}] \\
&= \sigma(G) + \epsilon \\
&\geq OPT + \epsilon.
\end{aligned}$$

Thus, G' cannot be in the set of credible networks. □

Besides this generic pruning rule, there are also specific rules for BIC and BDeu scoring with discrete data for finding the optimal structure. We generalize these pruning rules to the ϵBNSL problem below.

3.3.1 Pruning with BIC Score

A pruning rule comparing the BIC score and the penalty associated with a candidate parent set to those of its subsets was introduced by de Campos and Ji [29]. Recall that $t(\Pi_i)$ is the penalty term, $w = \log N/2$, and $\max_{\theta_i} L(\Pi_i)$ is the log likelihood term in the BIC scoring function (see Section 2.2.1).

Theorem 3.2 (de Campos and Ji [29])**.** *Given a vertex variable V_i, and candidate parent sets Π_i and Π'_i, if $\Pi_i \subset \Pi'_i$ and $\sigma(\Pi_i) - t(\Pi'_i) \cdot w < 0$, Π'_i and all supersets of Π'_i can be safely pruned if σ is the BIC scoring function.*

The following theorem gives a relaxed version of that pruning rule.

Theorem 3.3. *Given a vertex variable V_i, candidate parent sets Π_i and Π'_i, and some $\epsilon \in \mathbb{R}^+$, if $\Pi_i \subset \Pi'_i$ and $\sigma(\Pi_i) - t(\Pi'_i) \cdot w + \epsilon < 0$, Π'_i and all supersets of Π'_i can be safely pruned if σ is the BIC scoring function.*

Proof. (Theorem 3.3)

$$\begin{aligned}
&\sigma(\Pi_i) - t(\Pi'_i) \cdot w + \epsilon < 0 && \text{[given]}\\
&\Rightarrow -\sigma(\Pi_i) + t(\Pi'_i) \cdot w - \epsilon > 0 &&\\
&\Rightarrow -\sigma(\Pi_i) + t(\Pi'_i) \cdot w - \max_{\theta_i} L(\Pi'_i) - \epsilon > 0 && [\max_{\theta_i} L(\Pi'_i) < 0]\\
&\Rightarrow -\max_{\theta_i} L(\Pi'_i) + t(\Pi'_i) \cdot w > \sigma(\Pi_i) + \epsilon &&\\
&\Rightarrow \sigma(\Pi'_i) > \sigma(\Pi_i) + \epsilon. &&
\end{aligned}$$

By Lemma 3.1, Π'_i cannot be in the set of credible networks. Using the fact that penalties increase with increase in parent set size, supersets of Π'_i also cannot be in the set of credible networks. □

Another pruning rule for BIC is given by de Campos and Ji [29]. This provides a bound on the number of possible instantiations of subsets of a candidate parent set. Recall that r_i is the cardinality of the domain of variable V_i and r_{Π_i} is the number of possible instantiations of the parent set Π_i of V_i (see Section 2.1).

Theorem 3.4 (de Campos and Ji [29])**.** *Given a vertex variable V_i and candidate parent set Π_i such that $r_{\Pi_i} > \frac{N}{w}\frac{\log r_i}{r_i - 1}$, if $\Pi_i \subset \Pi'_i$, then Π'_i can be safely pruned if σ is the BIC scoring function.*

The following theorem gives a relaxed version of that pruning rule.

Theorem 3.5. *Given a vertex variable V_i, and a candidate parent set Π_i such that $r_{\Pi_i} > \frac{N}{w}\frac{\log r_i}{r_i - 1} + \epsilon$, for some $\epsilon \in \mathbb{R}^+$, if $\Pi_i \subset \Pi'_i$, then Π'_i can be safely pruned if σ is the BIC scoring function.*

Proof. (Theorem 3.5) The parent set Π'_i contains at least one additional variable; denote this variable as a. The difference in the scores $\sigma(\Pi_i') - \sigma(\Pi_i)$ is given by,

$$\begin{aligned}
&\sigma(\Pi_i') - \sigma(\Pi_i) \\
&\overset{0}{=} -\max_{\theta_i} L(\Pi_i') + t(\Pi_i') \cdot w + \max_{\theta_i} L(\Pi_i) - t(\Pi_i) \cdot w \\
&\overset{1}{\geq} \max_{\theta_i} L(\Pi_i) + t(\Pi_i') \cdot w - t(\Pi_i) \cdot w \\
&\overset{2}{=} \sum_{j=1}^{r_{\Pi_i}} n_{ij} \left(\sum_{k=1}^{r_i} \frac{n_{ijk}}{n_{ij}} \log \frac{n_{ijk}}{n_{ij}} \right) + t(\Pi_i') \cdot w - t(\Pi_i) \cdot w \\
&\overset{3}{=} -\sum_{j=1}^{r_{\Pi_i}} n_{ij} H(\theta_{ij}) + t(\Pi_i') \cdot w - t(\Pi_i) \cdot w \\
&\overset{4}{\geq} -\sum_{j=1}^{r_{\Pi_i}} n_{ij} \log r_i + r_{\Pi_i'} \cdot (r_i - 1) \cdot w - r_{\Pi_i} \cdot (r_i - 1) \cdot w \\
&\overset{5}{\geq} -\sum_{j=1}^{r_{\Pi_i}} n_{ij} \log r_i + r_{\Pi_i} \cdot (r_a - 1) \cdot (r_i - 1) \cdot w \\
&\overset{6}{\geq} -\sum_{j=1}^{r_{\Pi_i}} n_{ij} \log r_i + r_{\Pi_i} \cdot (r_i - 1) \cdot w \\
&\overset{7}{\geq} -N \cdot \log r_i + r_{\Pi_i} \cdot (r_i - 1) \cdot w \\
&\overset{8}{>} \epsilon.
\end{aligned}$$

Step 0 uses the definition of BIC. Step 1 uses that $\max_{\theta_i} L(\Pi_i')$ is negative. Step 2 uses the definition of the maximum likelihood estimate (see Equation 2.1) and algebra (multiplying by n_{ij}/n_{ij}). Step 3 uses the definition of (the sample estimate of) entropy, $H(X) = -\sum_{x \in X} P(X = x) \log P(X = x)$. Step 4 uses that $H(X) \leq \log |X|$ and the definition of the penalty function t. Step 5 uses that $r_{\Pi_i'} \geq r_{\Pi_i} \cdot r_a$, as Π_i' extends Π_i by at least one variable a, and algebra to simplify the expression. Step 6 uses that $r_a \geq 2$. Step 7 follows from the definition of n_{ij}. Step 8 uses the assumption of the theorem. By Lemma 3.1, Π_i' cannot be in the set of credible networks. □

The following corollary of Theorem 3.4 gives a useful upper bound on the size of a candidate parent set, where N is the number of instances in the dataset I.

Corollary 3.1 (de Campos and Ji [29])**.** *Given a vertex variable V_i and candidate parent set Π_i, if Π_i has more than $\log_2 N$ variables, Π_i can be safely pruned if σ is the BIC scoring function.*

Using Theorem 3.5, we generalize Corollary 3.1 to Corollary 3.2 and prove it below.

Corollary 3.2. *Given a vertex variable V_i and candidate parent set Π_i, if Π_i has more than $\lceil \log_2(N+\epsilon) \rceil$ variables, for some $\epsilon \in \mathbb{R}^+$, Π_i can be safely pruned if σ is the BIC scoring function.*

Proof. (Corollary 3.2) Take a variable V_i and a parent set Π_i with $|\Pi_i| = \lceil \log_2(N+\epsilon) \rceil$ variables. Because every variable has at least two states, we know that $r_{\Pi_i} \geq 2^{|\Pi_i|} \geq N + \epsilon > \frac{N}{w} \frac{\log r_i}{r_i - 1} + \epsilon$, because $w = \log \frac{N}{2}$ gives us $\frac{\log r_i}{w(r_i-1)} < 1$. By Theorem 3.5 no proper superset of Π_i can be in the set of credible networks. $\square$

Corollary 3.2 provides an upper-bound on the size of parent sets based solely on the dataset size N. The following table summarizes such an upper-bound given different amounts of data N and a BF of 20.

N	100	500	10^3	5×10^3	10^4	5×10^4	10^5
$\lvert\Pi\rvert$	7	9	10	13	14	16	17

The entropy of a candidate parent set is also a useful measure for pruning. A pruning rule, given by de Campos et al. [30], provides an upper bound on the conditional entropy of candidate parent sets and their subsets. First, we note that the sample estimate of entropy for a variable V_i is given by,

$$H(V_i) = -\sum_{k=1}^{r_i} \frac{n_{ik}}{N} \log \frac{n_{ik}}{N},$$

where n_{ik} represents how many instances in the dataset contain v_{ik}, where v_{ik} is an element in the domain Ω_i of V_i. Similarly, the sample estimate of entropy for a candidate parent set Π_i is given by,

$$H(\Pi_i) = -\sum_{j=1}^{r_{\Pi_i}} \frac{n_{ij}}{N} \log \frac{n_{ij}}{N}.$$

Conditional entropy is given by,

$$H(X \mid Y) = H(X \cup Y) - H(Y).$$

Lemma 3.2 (de Campos et al. [30]). *Given a vertex variable V_i, and candidate parent sets Π_i, Π'_i such that $\Pi'_i = \Pi_i \cup \{V_j\}$ for some variable $V_j \notin \Pi_i$, we have $L(\Pi'_i) - L(\Pi_i) \leq N \cdot \min\{H(V_i \mid \Pi_i), H(V_j \mid \Pi_i)\}$.*

Theorem 3.6 (de Campos et al. [30]). *Given a vertex variable V_i, and candidate parent set Π_i, let $V_j \notin \Pi_i$ such that $N \cdot \min\{H(V_i \mid \Pi_i), H(V_j \mid \Pi_i)\} < (1 - r_j) \cdot t(\Pi_i) \cdot w$. Then the candidate parent set $\Pi'_i = \Pi_i \cup \{V_j\}$ and all its supersets can be safely pruned if σ is the BIC scoring function.*

We relax Theorem 3.6 and prove its generalization below.

Theorem 3.7. *Given a vertex variable V_i, and candidate parent set Π_i, let $V_j \notin \Pi_i$ such that $N \cdot \min\{H(V_i \mid \Pi_i), H(V_j \mid \Pi_i)\} < (1 - r_j) \cdot t(\Pi_i) \cdot w + \epsilon$, for some $\epsilon \in \mathbb{R}^+$. Then the candidate parent set $\Pi'_i = \Pi_i \cup \{V_j\}$ and all its supersets can be safely pruned if σ is the BIC scoring function.*

Proof. (Theorem 3.7)

$$\begin{aligned}
\sigma(\Pi'_i) &\overset{0}{=} -L(\Pi'_i) + t(\Pi'_i) \cdot w \\
&\overset{1}{\geq} -L(\Pi_i) + N \cdot \min\{H(V_i \mid \Pi_i), H(V_j \mid \Pi_i)\} + t(\Pi'_i) \cdot w \\
&\overset{2}{>} -L(\Pi_i) + (1 - r_j) \cdot t(\Pi_i) \cdot w + \epsilon + t(\Pi'_i) \cdot w \\
&\overset{3}{=} -L(\Pi_i) + t(\Pi_i) \cdot w - r_j \cdot t(\Pi_i) \cdot w + \epsilon + t(\Pi'_i) \cdot w \\
&\overset{4}{=} -L(\Pi_i) + t(\Pi_i) - t(\Pi'_i) + \epsilon + t(\Pi'_i) \\
&\overset{5}{=} \sigma(\Pi_i) + \epsilon.
\end{aligned}$$

Step 1 uses Lemma 3.2. Step 2 uses the assumptions of the theorem. Step 4 uses $\Pi'_i = \Pi_i \cup \{V_j\}$. By Lemma 3.1, Π'_i cannot be in the set of credible networks. Using the fact that penalties increase with increase in parent set size, supersets of Π'_i also cannot be in the set of credible networks. □

3.3.2 Pruning with BDeu Score

A pruning rule for the BDeu scoring function is given by de Campos et al. [30] and a more general version is given by Cussens and Bartlett [26].

Lemma 3.3 (Cussens and Bartlett [26]). *Let n_{ij} be a positive integer and α' be a positive real number. Then,*

$$\log \frac{\Gamma(n_{ij} + \alpha')}{\Gamma(\alpha')} = \sum_{i=0}^{n_{ij-1}} \log(i + \alpha').$$

Lemma 3.4 (Cussens and Bartlett [26]). *Let $\{n_{ijk}\}$, $k = 1, ...r_i$, be non-negative integers with a positive sum, $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$, and α'' be a positive real number. Then,*

$$\sum_{k=1}^{r_i} \log \frac{\Gamma(n_{ijk} + \alpha'')}{\Gamma(\alpha'')} \leq \log \frac{\Gamma(n_{ij} + \alpha'')}{\Gamma(\alpha'')}.$$

Theorem 3.8 (Cussens and Bartlett [26]).

$$\sum_{j=1}^{r_{\Pi_i}} \left(\frac{\Gamma(\alpha')}{\Gamma(n_{ij} + \alpha')} + \sum_{k=1}^{r_i} \log \frac{\Gamma(n_{ijk} + \frac{\alpha'}{r_i})}{\Gamma(\frac{\alpha'}{r_i})} \right) \leq \sum_{\{j:n_{ij}>0\}} \sum_{i=0}^{n_{ij}-1} \log \left(\frac{i + a'/r_i}{i + \alpha'} \right).$$

Corollary 3.3 (Cussens and Bartlett [26]). *Define $r_i^+ := |\{j : n_{ij} > 0\}|$. Then,*

$$\sum_{j=1}^{r_{\Pi_i}} \log \frac{\Gamma(\alpha')}{\Gamma(n_{ij} + \alpha')} + \sum_{k=1}^{r_i} \log \frac{\Gamma(n_{ijk} + \frac{\alpha'}{r_i})}{\Gamma(\frac{\alpha'}{r_i})} \leq -r_i^+ \log r_i.$$

Corollary 3.4 (Cussens and Bartlett [26]). *Given a vertex variable V_i and candidate parent sets Π_i and Π'_i such that $\Pi_i \subset \Pi'_i$ and $\Pi_i \neq \Pi'_i$, let $r_i^+(\Pi'_i) := |\{j : n_{ij} > 0, j \in \Omega_{\Pi'_i}\}|$ be the total number of instantiations of Π'_i that appear in the dataset. If $\sigma(\Pi_i) < r_i^+(\Pi'_i) \log r_i$ then Π'_i and the supersets of Π'_i can be safely pruned.*

We generalize Corollary 3.4 to Theorem 3.9 and prove it below.

Theorem 3.9. *Given a vertex variable V_i and candidate parent sets Π_i and Π'_i such that $\Pi_i \subset \Pi'_i$ and $\Pi_i \neq \Pi'_i$, let $r_i^+(\Pi'_i) := |\{j : n_{ij} > 0, j \in \Omega_{\Pi'_i}\}|$ be the total number of instantiations of Π'_i that appear in the dataset. If $\sigma(\Pi_i) + \epsilon < r_i^+(\Pi'_i) \log r_i$, for some $\epsilon \in \mathbb{R}^+$, then Π'_i and the supersets of Π'_i can be safely pruned if σ is the BDeu scoring function.*

Proof. (Theorem 3.9) Let G' be a Bayesian network where Π'_i or one of its supersets is a parent set for V_i. Let G be another Bayesian network where Π_i is the parent set for V_i.

Consider the LHS of Corollary 3.3. It is the local BDeu score for a parent set Π'_i which has r_{Π_i} counts n_{ij} in its contingency table and counts n_{ijk} in the contingency table for $\Pi'_i \cup \{V_i\}$, where $\alpha' = \alpha / r_{\Pi_i}$ for some equivalent sample size α (see Equation 2.2). If $r_i^+(\Pi'_i) \log r_i > \sigma(\Pi_i) + \epsilon$ then $\sigma(\Pi_i) + \epsilon$ is lower than the local BDeu score for Π'_i due to Corollary 3.3. Take a candidate parent set Π''_i. If $\Pi'_i \subset \Pi''_i$ then $r_i^+(\Pi''_i) \leq r_i^+(\Pi'_i)$ and so $r_i^+(\Pi''_i) \log r_i \leq r_i^+(\Pi'_i) \log r_i$, as $r_i \geq 2$. From this it follows that the local score for Π''_i must also be more than $\sigma(\Pi_i) + \epsilon$. Using Lemma 3.1, the result follows as desired. □

3.4 Experimental Evaluation

In this section, we provide implementation details of the BF approach, as well as evaluate and compare its performance with published k-best solvers and the bootstrapping method from the widely used bnlearn library [90].

Our experiments show that the ϵBNSL approach is more memory efficient compared to the k-best based solvers in BDeu scoring and often collects more networks in less time. Used with the pruning rules generalized above, our ϵBNSL method can scale up to datasets with around 70 variables in BIC scoring, providing a significant improvement to the earlier state of the art results reported on a network of 29 variables using the k-best approach with score pruning [15].

The scoring computations were conducted on the Graham cluster of SHARCNET[2] and the structure learning experiments were conducted on either SHARCNET, a shared server with 346 GB RAM and Intel Xeon Gold 6148 CPUs at 2.4 GHz, or dedicated machines with 128 GB RAM and Intel Xeon Silver 4214R CPUs at 2.4 GHz. For scoring the datasets memory usage was limited to 64 GB and for structure learning a limit of 128 GB was imposed. For both scoring and learning, a computation time limit of 24 hours was imposed for each instance.

3.4.1 The Credible Set Approach

In this section, we study the scaling of our ϵBNSL approach and the effect of the BF when using the BIC scoring function. To evaluate our credible set approach, we considered a

[2]https://www.sharcnet.ca

wide selection of datasets from the UCI repository[3] and networks from the bnlearn Bayesian network repository[4] (see Table 3.1). We preprocessed the UCI datasets using a k-nearest neighbor imputation algorithm, with $k = 5$, to fill in missing values and a supervised discretization method [35] based on the MDL principle to discretize continuous variables. For the BN from the bnlearn repository, we used the logic sampling function `rbn` from the bnlearn R package [90] to generate datasets of sizes N = 50, 100, 500, 1,000, 5,000, and 10,000 from the bif files. The number of variables n used in these experiments, ranging from 10 to 76, pushes the limits of the ϵBNSL approach, especially when using the BDeu scoring that does not have effective pruning rules compared to BIC.

Table 3.1: UCI datasets (*left*, *middle*) and bnlearn Bayesian networks (*right*), where n is the number of variables in the dataset or network, and N is the number of instances in the original UCI dataset.

UCI dataset	n	N
shuttle	10	58,000
census income	14	48,842
letter	17	20,000
online shopping	18	12,330
lymphography	19	148
hepatitis	20	155
parkinsons	23	195
credit card	24	30,000

UCI dataset	n	N
robot navigation	25	5,456
horse colic	27	368
steel	28	1,941
flags	29	194
breast cancer	31	569
soybean	36	683
biodeg	42	1,055
spectf heart	45	267

network	n
sachs	11
child	20
insurance	27
water	32
mildew	35
alarm	37
barley	48
hailfinder	56
heparII	70
win95pts	76

We modified the development version (version denoted 9c9f3e6) of GOBNILP, denoted hereafter as GOBNILP_dev, to apply the pruning rules presented above and supplied appropriate parameter settings for collecting near-optimal networks[5]. This version of GOBNILP is compiled with SCIP 6.0.0 and CPLEX 12.8.0. GOBNILP handles the acyclicity constraint for DAGs by extending the SCIP Optimization Suite [44] with the addition of a constraint handler. To learn multiple BNs, GOBNILP_dev uses SCIP calls for collecting feasible solutions. In this mode, when SCIP finds a solution, the solution is stored, a constraint is added to render that solution infeasible and the search continues. This differs

[3]https://archive.ics.uci.edu/ml

[4]https://www.bnlearn.com/bnrepository/

[5]Codebase is available at: https://www.cs.york.ac.uk/aig/sw/gobnilp/

from (and is much more efficient than) the method used in the current stable version of GOBNILP for finding k-best BNs where an entirely new search is started each time a new BN is found. By default when SCIP is asked to collect solutions it turns off all cutting plane algorithms. This led to very poor GOBNILP performance since GOBNILP relies on cutting plane generation. Therefore, this default setting is overridden in GOBNILP_dev to allow cutting planes when collecting solutions. To find only solutions with objective no worse than $(OPT + \epsilon)$, SCIP's `SCIPsetObjlimit` function is used.

Since GOBNILP_dev uses an objective limit $OPT + \epsilon$ to enumerate possible networks, we start by utilizing it to determine the optimal score OPT. For BDeu, no limit was placed on the possible parent set sizes when scoring the datasets. For BIC, we set the limit on the size of the parent set based on Corollary 3.2 that assures optimality. Then, with a 250,000-counting limit, all networks within the limit are collected.

Table 3.2 reports the search time T and the number of collected networks $|\mathcal{G}|$ for BF of 3, 20 and 150 using BIC, where n is the number of random variables in the dataset, N is the number of instances in the dataset, and the datasets are from the UCI repository. Table 3.3 reports the same metrics for the bnlearn repository. The three thresholds were determined using the interpretation scale from Heckerman et al. [51], where 3 denotes the transition between anecdotal and positive evidence, 20 between positive and strong evidence, and 150 between strong and very strong evidence. The network size, sample size, and the number of networks at a particular BF all together have an impact on how long it takes to complete a search. Smaller networks with a big sample size, such as shuttle and census, are solved considerably more quickly than other relatively large networks, such as soybean and spectf-heart.

The findings also show that there are significant differences between different datasets in the number of collected networks at the three BF levels. Since near-optimal networks have similar posterior probabilities to the optimal network, datasets with fewer instances typically have more networks collected at a particular BF. Despite the fact that the required level of BF for a study, like the p-value, is frequently determined by domain knowledge, the proposed approach will, given enough samples, yield significant results that may be used for further analysis. One methodology that we can recommend is to set the BF as large as possible such that the number of networks can still be reasonably handled. We found that a BF of 1000 was often feasible for larger datasets, which is large enough to provide exceedingly strong guarantees on the final model averaged network and the resulting data analysis.

Table 3.2: The search time T and the number of collected networks $|\mathcal{G}|$ for the credible set method (BF = 3, 20, and 150) on the UCI datasets using BIC scoring, where n is the number of random variables in the dataset and N is the number of instances in the dataset.

			BF = 3		BF = 20		BF = 150	
UCI dataset	n	N	T_3 (s)	$\|\mathcal{G}_3\|$	T_{20} (s)	$\|\mathcal{G}_{20}\|$	T_{150} (s)	$\|\mathcal{G}_{150}\|$
shuttle	10	58,000	0.3	6	0.3	6	0.3	6
census	14	48,842	6.1	26	5.6	26	5.5	26
letter	17	20,000	0.9	68	0.9	68	0.9	68
online-shopping	18	12,330	11.6	10	8.0	10	8.3	30
lymphography	19	148	0.9	140	4.1	1,682	50.9	16,624
hepatitis	20	155	4.8	564	126.8	29,378	1,023.8	250,000
parkinsons	23	195	220.2	15,776	3,717.7	250,000	3,823.7	250,000
credit-card	24	30,000	224.0	42	183.6	42	231.7	84
robot	25	5,456	2.4	25	2.4	25	2.3	25
horse-colic	27	368	2.7	32	5.7	366	15.5	2,017
steel	28	1,941	19.0	1,152	34.6	2,304	81.2	4,608
flags	29	194	3.7	295	58.4	10,325	874.0	199,541
breast-cancer	31	569	8.7	40	22.5	294	65.7	1,437
soybean	36	683	33.8	73	91.6	488	392.3	3,200
biodeg	42	1,055	118.5	6	130.1	15	156.3	72
spectf-heart	45	267	15.4	207	125.7	4,350	1,920.2	86,275

Table 3.3: The search time T and the number of collected networks $|\mathcal{G}|$ for the credible set method (BF = 3, 20, and 150) on the bnlearn networks using BIC scoring, where n is the number of random variables in the dataset, N is the number of instances in the dataset, and OM = Out of Memory.

			BF = 3		BF = 20		BF = 150	
network	n	N	T_3 (s)	$\lvert\mathcal{G}_3\rvert$	T_{20} (s)	$\lvert\mathcal{G}_{20}\rvert$	T_{150} (s)	$\lvert\mathcal{G}_{150}\rvert$
		50	0.3	84	0.7	404	2.4	2,343
sachs	11	500	0.4	168	0.7	336	1.8	796
		5000	1.9	336	1.7	336	1.8	336
		50	1.2	821	20.6	20,840	419.4	250,000
child	20	500	0.5	48	0.8	257	3.1	1,156
		5000	3.0	24	2.7	24	2.6	24
		50	29.8	21,104	281.7	250,000	370.5	250,000
insurance	27	500	1.7	42	3.1	272	7.7	1,489
		5000	6.2	50	9.2	50	10.0	200
		50	0.3	840	1.8	5,004	7.2	19,810
water	32	500	4.0	4,050	7.6	8,991	20.8	23,409
		5000	1.0	48	1.0	48	1.4	87
		50	0.0	16	0.0	66	0.1	106
mildew	35	500	0.6	729	0.7	972	0.9	1,620
		5000	2.4	513	2.4	513	3.4	1,026
		50	225.8	165,072	296.0	250,000	586.2	250,000
alarm	37	500	3.8	238	26.6	6,468	428.1	110,398
		5000	32.4	16	34.9	148	48.8	2,428
		50	0.0	3	0.0	3	0.0	17
barley	48	500	3.0	2,244	7.7	8,784	14.9	11,364
		5000	5.5	12	4.9	12	3.0	24
		50	122.2	250,000	146.5	250,000	171.4	250,000
hailfinder	56	500	452.6	250,000	372.8	250,000	473.9	250,000
		5000	OM	—	OM	—	OM	—
		50	1,087.9	250,000	4,349.4	250,000	OM	—
heparII	70	500	422.8	250,000	815.0	250,000	2,981.9	250,000
		5000	OM	—	OM	—	OM	—
		50	870.5	250,000	4,006.2	250,000	OM	—
win95pts	76	500	OM	—	OM	—	OM	—
		5000	OM	—	OM	—	OM	—

3.4.2 Credible Set vs. K-Best

In this section, we compare our credible set approach to published solvers that find a subset of top k scoring networks for a given parameter k. The solvers under consideration are KBest (KBest_12b)[6] from [102] and KBestEC[7] from [16], which are based on the dynamic programming approach introduced in [95]. KBest finds the k best networks and KBestEC finds the k best MECs (see Definition 2.7). KBest and KBestEC lack support for BIC, thus only results using BDeu with an equivalent sample size of one are shown in the corresponding experiments.

The experimental results of KBest, KBestEC, and GOBNILP_dev (our implementation of our credible set approach) using the BDeu scoring function are reported in Table 3.4, where n is the number of random variables in the dataset, N is the number of instances in the dataset, and k is the number of top scoring networks. The search time is reported for KBest, KBestEC, and GOBNILP_dev. The number of DAGs within the k MECs $|\mathcal{G}_k|$ is reported for KBestEC. The last two columns are the number of found networks $|\mathcal{G}_{20}|$ and the number of MECs $|\mathcal{M}_{20}|$ using the BF approach with a BF of 20.

The search times for KBest and KBestEC both increase exponentially with the number of requested networks k. The KBest and KBestEC do not perform well with larger datasets as they are built to solve instances with an n under 20[8]. Additionally, KBestEC fails to produce accurate scoring files for the datasets shuttle, census, letter, robot, and horse-colic. While KBestEC incurs some overhead for equivalence class checking, it seems to be able to increase the coverage of DAGs. For some cases, such as online-shopping and credit-card, KBestEC takes substantially longer than KBest, and the number of DAGs covered by the discovered MECs is inconsistent. The credible set approach is consistently much faster than the k-best approaches, which is due to the generalized pruning rules being very effective in reducing the search space and allowing GOBNILP_dev to solve the problem. Comparing to the results in [13, 14], our approach can scale to larger networks if the set of pruned scores can be computed for a given BF[9].

[6]http://web.cs.iastate.edu/~jtian/Software/UAI-10/KBest.htm

[7]http://web.cs.iastate.edu/~jtian/Software/AAAI-14-yetian/KBestEC.htm

[8]Obtained through correspondence with the author.

[9]We are unable to generate BDeu score files for datasets with 30 or more variables within our experimental constraints.

Table 3.4: The search time T and the number of collected networks k, $|\mathcal{G}_k|$ and $|\mathcal{G}_{20}|$ for KBest, KBestEC, and credible set (BF = 20) methods on the UCI datasets using BDeu scoring, where n is the number of random variables in the dataset, N is the number of instances in the dataset, OT = Out of Time, and ES = Error in Scoring. Note that $|\mathcal{G}_k|$ is the number of DAGs covered by the k-best MECs in KBestEC and $|\mathcal{M}_{20}|$ is the number of MECs for the networks collected by the credible set method.

			KBest		KBestEC		Credible set		
UCI dataset	n	N	T_k (s)	k	T_{EC} (s)	$\|\mathcal{G}_k\|$	T_{20} (s)	$\|\mathcal{G}_{20}\|$	$\|\mathcal{M}_{20}\|$
shuttle	10	58,000	1.8	10	ES	—	17.5	44	1
			3.7	100	ES	—			
			60.1	1,000	ES	—			
census	14	48,842	3.6	10	ES	—	3.5	52	1
			73.3	100	ES	—			
			1,549.8	1,000	ES	—			
letter	17	20,000	16.1	10	ES	—	5.5	47	1
			477.5	100	ES	—			
			21,480.2	1,000	ES	—			
online-shopping	18	12,330	18.3	10	43.3	11	12.9	30	2
			468.3	100	86,406.7	112			
			18,533.0	1,000	OT	—			
lymphography	19	148	147.6	10	383.2	42	126.5	17,909	3,531
			5,812.1	100	8,397.2	464			
			OT	1,000	OT	—			
hepatitis	20	155	38.4	10	96.2	42	87.6	13,984	2,525
			1,045.7	100	1,169.4	446			
			34,374.8	1,000	50,592.0	5,309			
parkinsons	23	195	4,171.5	10	OT	—	8,925.2	250,000	46,156
			OT	100	OT	—			
			OT	1,000	OT	—			
credit-card	24	30,000	133.5	10	4,507.7	10	263.2	6	1
			6,724.6	100	OT	—			
			OT	1,000	OT	—			
robot	25	5,456	OT	10	ES	—	6.6	5	1
			OT	100	ES	—			
			OT	1,000	ES	—			
horse-colic	27	368	935.2	10	ES	—	15.4	1,143	154
			39,097.0	100	ES	—			
			OT	1,000	ES	—			

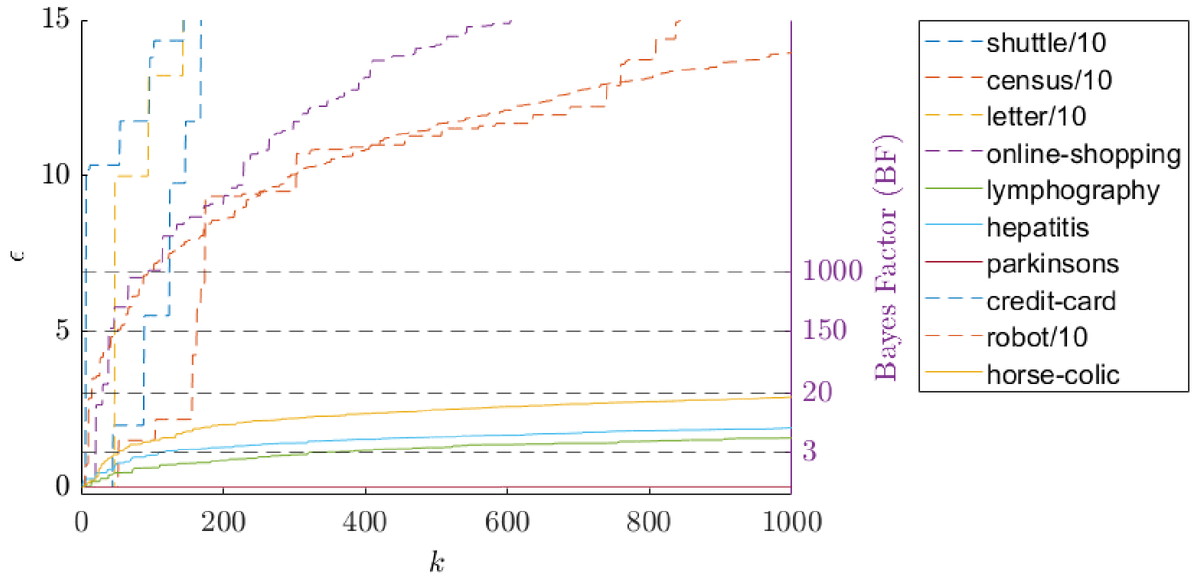


Figure 3.1: The deviation ϵ from the optimal BDeu score by k, where k is the number of best scoring networks retrieved. The corresponding values of the BF ($\epsilon = \log(\text{BF})$) are presented on the right. For example, if the desired BF value is 20, then all networks falling below the dash line at 20 are credible.

Figure 3.1 shows score patterns within the 1,000 best networks for some datasets in the KBest experiment. A plot for a dataset indicates the deviation ϵ from the optimal BDeu score by the kth-best network learned from that dataset and for reference the black dashed lines represent different levels of BFs calculated by $\epsilon = \log BF$ (See Equation 3.2). The plots show that it is difficult to pick a value for k *a priori* to capture the appropriate set of top scoring networks. For datasets like shuttle and online-shopping, it takes fewer than 50 networks to reach a BF of 20, whereas hepatitis needs almost 14,000 networks. In cases with limited data, several BNs with identical probabilities can be learned at a given BF, indicating that the sample size significantly affects the number of networks learned. When there is a lack of data, it would be a reasonable choice to use a high value for k in the model, and vice versa. However, only our credible set can automatically identify a suitable and reliable set of networks for additional study.

3.4.3 Credible Set vs. Bootstrapping

In this section, we compare our global (optimal) credible set approach for model averaging to a local (heuristic) solver for model averaging. We compare against the bootstrapping approach from bnlearn [90] with the tabu search algorithm, as it is known to be among the best local solvers [91] and it is widely popular (the R implementation of the bnlearn package has been downloaded more than 56.4k times). The number of bootstrap replicates is the default value of 200. When comparing our credible set approach against the KBest approaches, we compared on solving time, as these approaches are all optimal (see Section 3.4.2). Now when comparing against a non-optimal solver, we wish to quantify the improvement in accuracy gained by using our optimal approach.

Model averaging in structure learning for BNs is used in the following way: (i) learn a set of networks with good scores that fit the given data, (ii) perform model averaging over the set of networks and compute a real-valued confidence value $P(e \mid I)$ for each edge e in the learned set of networks , and (iii) set a minimum threshold value, t and report all significant edges, which are edges e such that their confidence values exceed the threshold $P(e \mid I) > t$. In this manner, a model-averaged network can be constructed from the significant edges, and used to analyze causal and probabilistic dependencies. Two methods are available to form a confidence value $P(e \mid I)$: each network in the set of networks is equally weighted and each network is weighted by its score, where networks closer to optimal receive more weight than those further from optimal. Here we use equal weighting and a fixed threshold of $t = 0.5$. In our experiments, we found that the difference between the two methods was negligible.

For evaluating the two model averaging approaches on the task of BN structure learning, we used a total of 90 ground truth BNs: 10 ground truth BNs came from the bnlearn repository and a further 80 ground truth BNs were constructed following a similar approach to Liu et al. [69] by (i) scoring each of the 16 UCI datasets using each of the five scoring functions AIC, BDeu, BIC, qBDJ, and qNML (ii) learning an optimal network structure from each scored dataset, and (iii) and fitting the parameters to each structure to give a final Bayesian network. Given these ground truth BNs, we used the logic sampling function `rbn` from the bnlearn `R` package [90] to generate random samples of sizes $N = 50$, 100, 500, 1,000, 5,000, and 10,000 from the bif files. We collected three samples for each dataset size N, for a total of 18 samples for each ground truth BN. Thus, there are 90×18 instances all together, each associated with a ground truth DAG. Having ground truth DAGs allows us to measure the accuracy of each approach. The number of variables n used in our experiments, ranging from 11 to 76, pushes the limits of both the bootstrap and the credible set approaches, especially when using the BDeu scoring that does not have effective pruning rules compared to BIC.

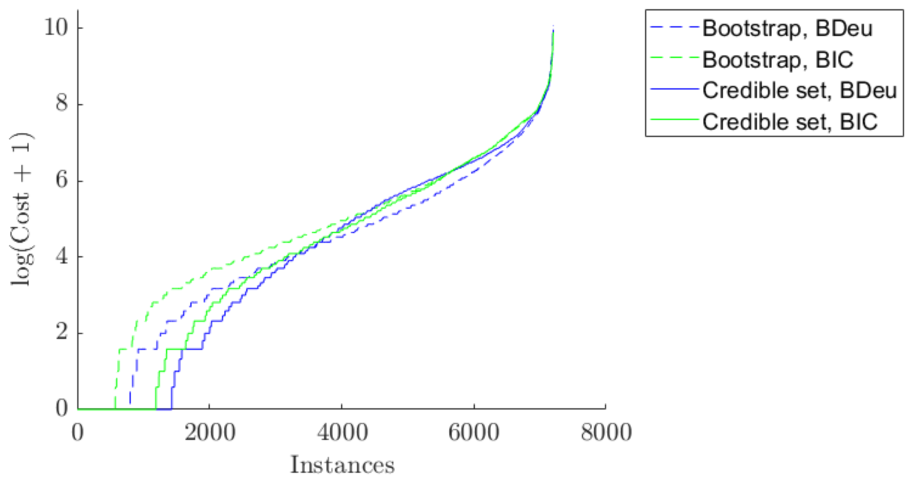
We compare the results of knowledge discovery using the credible set approach against the bootstrap approach in Figures 3.2 & 3.3. For each method, we plot results for the BIC and BDeu scoring functions. Each "performance curve" is obtained by solving each of the 90×18 instances, evaluating the learned DAG against ground truth using the performance measure, and sorting the results into ascending order. For the performance measures (e.g., F_β) that take an additional parameter, we use $\alpha, \beta = \frac{1}{4}, \frac{1}{2}, 1, 2$, and 4. The performance curves can then be compared and we look for whether one method dominates another for a given scoring function.

Figure 3.2 compares the credible set and bootstrap methods using the misclassification cost and F_β score performance measures over the skeletons. For the misclassification cost, the credible set method dominates the bootstrap method for the BIC score, always being better or equal in performance, while the results for the BDeu score are more mixed (note the log scale). For the F_β score, the credible set method again dominates the bootstrap method for the BIC score, while again the results for the BDeu are more mixed.

Figure 3.3 compares the credible set and bootstrap methods using the SHD and multi-class F_β score performance measures over CPDAGs. For the SHD, the credible set method offers significant accuracy improvements over the bootstrap method for both the BIC score and the BDeu score (note the log scale). For the multi-class F_β score, the credible set method dominates and again offers significant accuracy improvements over the bootstrap method for both the BIC score and the BDeu score.

3.5 Summary

Existing optimal methods to apply model averaging in Bayesian network structure learning are limited by constraints on the Bayesian network structure, or do not scale to networks with more than 30 random variables. In this chapter, we provided a novel model averaging method for learning all networks within a factor of optimal in Bayesian network structure learning. Our method improves on the existing methods in three ways. First, only *credible* models—those whose scores are optimal or close to optimal—are learned by our method. Second, our method is substantially more efficient and scales to far larger Bayesian networks. Empirically, our results demonstrate that the modified GOBNILP scales to significantly larger networks without restricting the structure of the learned Bayesian networks. Finally, our empirical results demonstrate that our credible set method can offer significant accuracy improvements over a bootstrapping model averaging method that is far and away the most widely used model averaging method in practice.



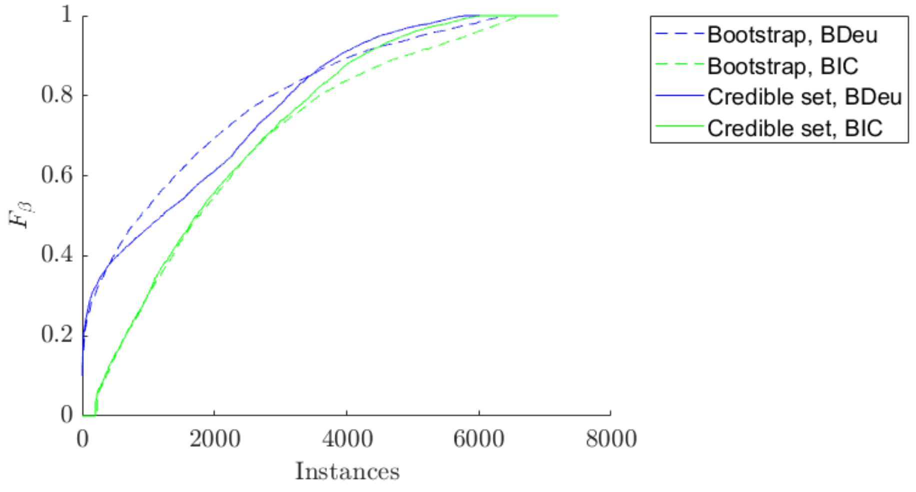
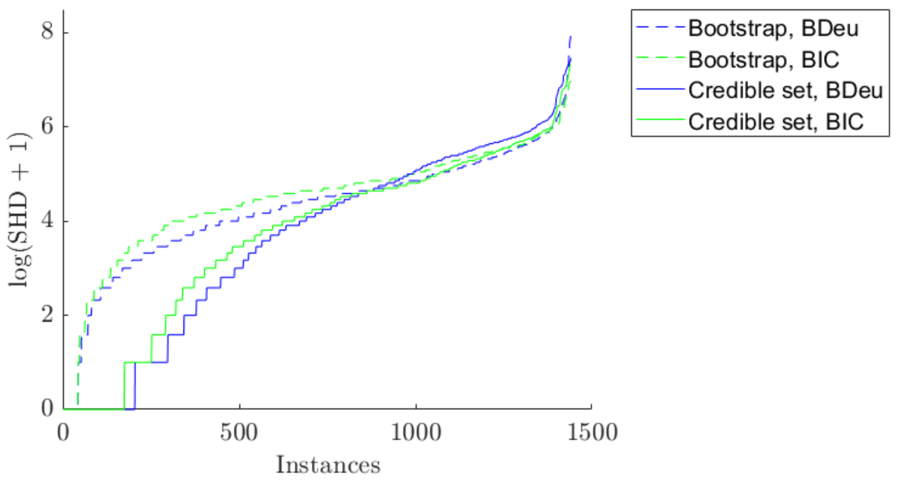


Figure 3.2: Comparison of bootstrap and credible set model averaging methods, for various scoring functions and performance measures for undirected edges (skeleton): misclassification cost (top, lower values are better); F_β score (bottom, higher values are better). All methods used a fixed threshold of 0.5.



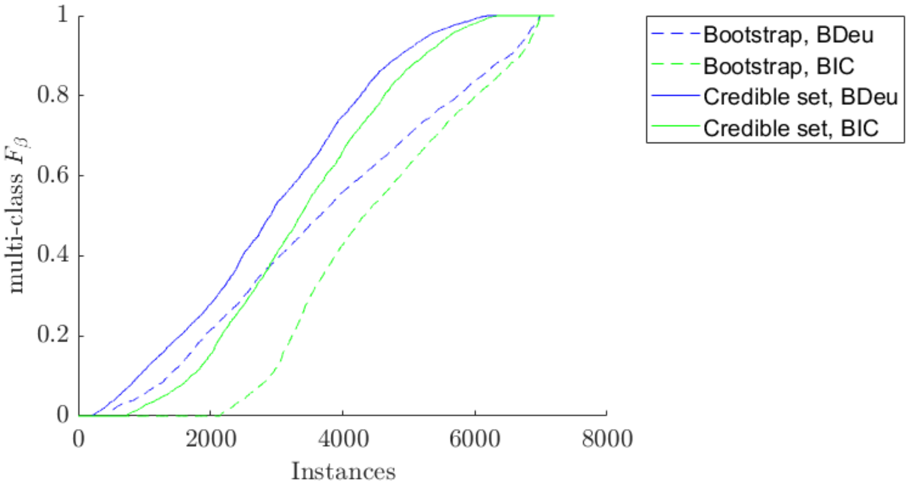


Figure 3.3: Comparison of bootstrap and credible set model averaging methods, for various scoring functions and performance measures for directed edges (CPDAGs): structural Hamming distance (top, lower values are better); multi-class F_β score (bottom, higher values are better). All methods used a fixed threshold of 0.5.

Chapter 4

Local Structure: Noisy-OR

A Bayesian network can be learned from data using the well-known score-and-search approach, and within this approach a key consideration is how to simultaneously learn the global structure in the form of the underlying DAG and the local structure in the CPDs. Several useful forms of local structure have been identified in the literature but thus far the score-and-search approach has only been extended to handle simultaneously local structure in the form of context-specific independence (see the review of related work and discussion in Section 1.1.3). In this chapter, we show how to extend the score-and-search approach to the important and widely useful case of noisy-OR relations. We provide an effective gradient descent algorithm to score a candidate noisy-OR using the widely used BIC score and we provide pruning rules that allow the search to successfully scale to medium sized networks. Our empirical results provide evidence for the success of our approach to learning Bayesian networks that incorporate noisy-OR relations.

4.1 Overview

A widely used local structure is the noisy-OR relation [45, 80], which models the CPD over causes (parents) and effects (children). The noisy-OR assumes a form of causal independence and allows one to specify a CPT with parameters linear in the number of parents. Given the DAG, previous work has shown how to learn the parameters of a noisy-OR relation from data or from a CPT [113, 108]. Here, we propose the first score-and-search approach to simultaneously learn the DAG and learn both CPT and noisy-OR relations as possible representations for the CPDs. Importantly, we place no a priori constraints on the global structure and we exactly determine all networks within a given factor of optimal.

Our approach has two primary advantages. First, our approach only replaces a CPT with a noisy-OR relation when it is appropriate. Converting an arbitrary proportion of CPTs to structured representations can lead to significant degradation of the expressive power of the model, and it is difficult to determine the optimal proportion a priori. We control the degradation by specifying an approximation factor that measures how far a BN can deviate from the optimal network, and so only near-optimal networks with both CPTs and noisy-OR relations are learned in a principled manner. Second, our approach provides pruning rules that can scale to BNs of moderate sizes in contrast to the state of the art using other structured representations. We empirically demonstrate that our approach can learn these *mixed* BNs in a principled manner that takes advantage of a reduced complexity.

The remainder of this chapter proceeds as follows.

- We provide an algorithm to learn the parameters of a noisy-OR relation for a candidate parent set, as well as score the noisy-OR parent set simultaneously (Section 4.2).
- We provide pruning rules for noisy-OR parent sets that retain optimality while reducing the search space (Section 4.2).
- We extend our credible set score-and-search approach with model averaging to choose between noisy-OR and CPTs automatically. Our approach controls the degradation by specifying a Bayes factor (BF) [58] that measures how far a BN can deviate from the optimal network, and so only near-optimal networks with both CPTs and noisy-OR relations are learned in a principled manner (Section 4.2).
- We show empirically that our approach to learning parameters of noisy-OR relations is more accurate than the state-of-the-art, and the structure learning with our pruning rules can scale to BNs of moderate sizes (Section 4.3).

4.2 Score-and-Search with Noisy-OR

In this section, we present our credible set score-and-search approach for learning all BNs given local scores that are within a given factor of optimal, where the networks can contain both full CPT and noisy-OR relations as possible representations for the CPDs. For ease of reference, we begin by recalling the notation and definition for the noisy-OR structured representation (Section 2.4).

Recall that with the noisy-OR relation, one assumes that there are a set of causes $\Pi_i := \{V_{i1}, \dots, V_{i|\Pi_i|}\}$ leading to an effect V_i, where $V_i, V_{ij} \in V$ for all $j \in \{1, ...|\Pi_i|\}$ and

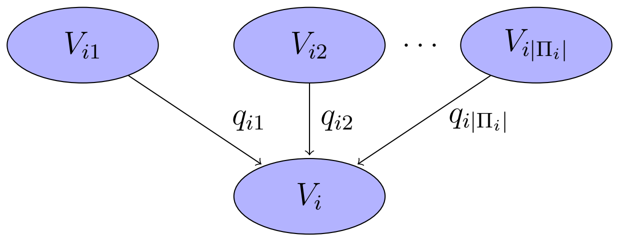


Figure 4.1: Causal structure for a BN with a noisy-OR relation, where the set of causes $\Pi_i := \{V_{i1}, \ldots, V_{i|\Pi_i|}\}$ leads to effect V_i and there is a noisy-OR relation at variable V_i.

$V_i \notin \Pi_i$ (see Figure 4.1). Each cause $V_{ij} \in \Pi_i$ is either present or absent, and each V_{ij} in isolation is likely to cause V_i and the likelihood is not diminished if more than one cause is present. Further, one assumes that all possible causes are given and when all causes are absent, the effect is absent. Finally, one assumes that the mechanism or reason that inhibits a V_{ij} from causing V_i is independent of the mechanism or reason that inhibits a $V_{ij'}$, $j' \neq j$, from causing V_i.

For a binary variable V_i and parent set Π_i, a noisy-OR relation specifies a CPT using $|\Pi_i|$ parameters, $\mathbf{q}_i = q_{i1}, \ldots, q_{i|\Pi_i|}$, one for each parent, where q_{ij} is the probability that V_i is false given that V_{ij} is true and all of the other parents are false,

$$P(V_i = 0 \mid V_{ij} = 1, V_{ij'} = 0_{[\forall j', j' \neq j]}) = q_{ij}.$$

From these parameters, the full CPT representation of size 2^{n+1} can be generated using,

$$\phi_{ij0} = \begin{cases} \prod_{j \in T_x} q_{ij} & \text{if } T_x \neq \{\} \\ 1 & \text{otherwise,} \end{cases} \tag{4.1}$$

where $T_x = \{j \mid V_{ij} = 1\}$. The last condition (when $T_x = \{\}$) corresponds to the assumptions that all possible causes are given and that when all causes are absent, the effect is absent; i.e., $P(V_i = 0 \mid V_{i1} = 0, \ldots, V_{i|\Pi_i|} = 0) = 1$. Of course, $\phi_{ij1} = 1 - \phi_{ij0}$. The set $\phi_i := \big\{\phi_{ijk} \mid j \in \{1, \ldots, r_{\Pi_i}\}, k \in \{0, 1\}\big\}$ is referred to as the *noisy-OR CPT* of variable V_i.

In general, a score-and-search approach scores candidate parent sets for the variables in the network and searches for the choice of a parent set, one for each variable, that leads to the best overall score while ensuring that the network is acyclic. In what follows, we present our overall approach for solving ϵBNSL (our credible-set model-averaging approach) extended to noisy-OR relations. We first describe an effective gradient descent algorithm to score a candidate noisy-OR relation using the widely used BIC score and pruning rules that allow the search to scale to larger networks.

4.2.1 BIC Score for Noisy-OR Relations

The BIC score consists of a maximum likelihood term and a penalty term (see Equation 2.2.1). We present a gradient descent algorithm that is based on minimizing a KL divergence as it is known that minimizing the KL divergence results in maximizing the likelihood (see, e.g., [76]). Recall that the elements of θ_i in the log likelihood term of the BIC score are conditional probabilities computed from the dataset I. Given a variable V_i, we must compute maximum likelihood estimates for the noisy-OR CPT ϕ_i for every candidate parent set Π_i, such that the conditional KL divergence between the full CPT θ_i and the resulting noisy-OR CPT ϕ_i that is determined by the $\mathbf{q}_i$ (see Equation 4.1), is minimized. The KL divergence between two *conditional* probability distributions, $P(A \mid B)$ and $Q(A \mid B)$ is given by,

$$D_{KL}(P(A|B) \mid\mid Q(A|B)) = \sum_{b \in B} P(B=b) \sum_{a \in A} P(A=a|B=b) \log \frac{P(A=a|B=b)}{Q(A=a|B=b)}.$$

We note that an alternative approach to estimate noisy-OR parameters is to maximize the log-likelihood using the expectation-maximization (EM) technique, which was derived in [31] and applied to noisy-OR in [108]. We perform an experimental comparison of the two alternative approaches in Section 4.3.

To derive our gradient descent algorithm, we begin with the definition of KL divergence for the two conditional probability distributions, θ_i and ϕ_i, and rewrite it into a more convenient form:

$$\begin{aligned}
D_{KL}(\theta_i \mid\mid \phi_i) &\overset{0}{=} \sum_{j=1}^{r_{\Pi_i}} P(\pi_{ij}) \sum_{k \in \{0,1\}} \theta_{ijk} \log \frac{\theta_{ijk}}{\phi_{ijk}} \\
&\overset{1}{=} \sum_{j=1}^{r_{\Pi_i}} \frac{n_{ij}}{N} \sum_{k \in \{0,1\}} \theta_{ijk} \log \frac{\theta_{ijk}}{\phi_{ijk}} \\
&\overset{2}{=} \frac{1}{N} \sum_{j=1}^{r_{\Pi_i}} \sum_{k \in \{0,1\}} n_{ijk} \cdot \theta_{ijk} \log \frac{\theta_{ijk}}{\phi_{ijk}} \\
&\overset{3}{=} \frac{1}{N} \sum_{j=1}^{r_{\Pi_i}} \sum_{k \in \{0,1\}} n_{ijk} \cdot \theta_{ijk} \log \theta_{ijk} - \frac{1}{N} \sum_{j=1}^{r_{\Pi_i}} \sum_{k \in \{0,1\}} n_{ijk} \cdot \theta_{ijk} \log \phi_{ijk},
\end{aligned}$$

where N is the number of instances in our dataset. To find ϕ_i such that $D_{KL}(\theta_i \mid\mid \phi_i)$ is minimized, note that the first term in Step 3 is constant. So, we must determine,

$$\underset{\mathbf{q_i}}{\text{argmin}}\ D_{KL}(\theta_i \mid\mid \phi_i) = -\sum_{j=1}^{r_{\Pi_i}} \sum_{k \in \{0,1\}} n_{ijk} \cdot \theta_{ijk} \log \phi_{ijk},$$

where the $\mathbf{q}_i$ that minimizes the KL divergence are the maximum likelihood estimates for ϕ_i that are determined by the $\mathbf{q}_i$ (Equation 4.1). The penalty term in the BIC score can be computed in constant time; specifically, the number of parents in the candidate parent set. Thus, fitting these noisy-OR parameters gives us the BIC score for the noisy-OR for a candidate parent set. To find these noisy-OR parameters, we propose Algorithm 4.1 which performs gradient descent for the derivative,

$$\Delta_{KL}^{\mathbf{q}_i} = \frac{d}{d\mathbf{q}_i} \sum_{j=1}^{r_{\Pi_i}} \sum_{k \in \{0,1\}} n_{ijk} \cdot \log \phi_{ijk}. \tag{4.2}$$

We start with an initial guess for the set of noisy-OR parameters $\mathbf{q}_i$ and evaluate term $\Delta_{KL}^{\mathbf{q}_i}$ for these values (Equation 4.2). The initial guess uses hot starts in that the solution for a smaller candidate parent set is used as the starting point when estimating the parameters for a candidate parent set that is a superset. We perform gradient descent over $\mathbf{q}_i$, where each step update is found by a simple geometric line search algorithm (see Step 5, Algorithm 4.1). Geometric line search is a backtracking line search procedure, where we first choose a descent direction and then determine the maximum amount to move along that direction.

4.2.2 Pruning Rules

To find all near-optimal BNs given an approximating factor ϵ for a dataset I, we propose to compute two different sets of local scores for each variable. The first set is the BIC scores when the conditional probability distributions for the candidate parents sets are represented by full CPTs. The second set is the BIC scores when the conditional probability distributions for the candidate parent sets are represented by noisy-OR relations. However, computing the local scores for all variables is quite cost prohibitive—we would need a set of $n \cdot 2^{n-1}$ local scores for each of the two BIC scores. A solution is to prune the search space of candidate parent sets, provided that global optimality constraints of the full network structure are not violated. Recall that we say that a candidate parent set Π_i can be *safely pruned* given a non-negative constant $\epsilon \in \mathbb{R}^+$ if Π_i cannot be the parent set of V_i in any network in the set of credible networks (see Definition 3.4).

Algorithm 4.1 Computing Noisy-OR Parameters for a Candidate Parent Set

Input: Variable V_i, candidate parent set Π_i, a dataset I of N instances.
Parameter: Threshold t, maximum iterations *maxIter*
Output: A set of noisy-OR parameters : $\mathbf{q}_i = q_{i1}, \ldots, q_{i|\Pi_i|}$

```
1:  Initialize $\mathbf{q}_i = q_{i1}, \ldots, q_{i|\Pi_i|} = hotstarts()$
2:  Initialize $l = 0, \mathbf{mq}_i = \mathbf{q}_i, \delta = \infty$
3:  while $l < maxIter$ do
4:      $\mathbf{q}'_i = \mathbf{q}_i$
5:      $step = GeometricLineSearch(\mathbf{q}'_i, \Delta_{KL}^{\mathbf{q}'_i})$
6:      $\mathbf{q}_i = \mathbf{q}'_i - step * \Delta_{KL}^{\mathbf{q}'_i}$
7:      $\delta q_i = \Delta_{KL}^{\mathbf{q}_i} - \Delta_{KL}^{\mathbf{q}'_i}$
8:      if $\delta q_i < \delta$ then
9:          $\mathbf{mq}_i = \mathbf{q}_i$
10:         $\delta = \delta q_i$
11:     end if
12:     if $\delta q_i < t$ then
13:         break
14:     end if
15:     $l = l + 1$
16: end while
17: return $\mathbf{mq}_i$
```

For computing BIC scores for full CPTs, we employ the pruning rules given in Section 3.3 to find all near-optimal BNs given an approximating factor ϵ. For computing BIC scores for noisy-OR relations, we introduce new pruning rules.

Lemma 4.1. *A candidate parent set Π_i of a variable V_i such that Π_i is consistently instantiated to $(0, \ldots, 0)$ throughout the dataset whenever the variable V_i is set to one can be safely pruned.*

Proof. The candidate parent set Π_i cannot explain V_i in this configuration as there is no instance in the dataset to indicate that Π_i affects the values of V_i. □

Theorem 4.1. *Given a vertex variable V_i, a candidate parent set Π_i, and some $\epsilon \in \mathbb{R}^+$, if $t(\Pi_i) \cdot w > \sigma_i(\{\}) + \epsilon$, Π_i can be safely pruned.*

Proof. The null parent set is a subset of all candidate parent sets and by Lemma 3.1 any candidate parent set with a score exceeding the score of the null parent set plus ϵ

can be safely pruned. Consider the definition of BIC for a parent set Π_i for variable V_i, $\sigma(\Pi_i) = -L(\theta_i)+t(\Pi_i)\cdot w$. Let us have a candidate parent set such that $t(\Pi_i)\cdot w > \sigma_i(\{\})+\epsilon$. Then, since log-likelihood is negative, $t(\Pi_i) \cdot w > \sigma_i(\{\}) + \epsilon \Rightarrow -L(\theta_i) + t(\Pi_i) \cdot w > \sigma_i(\{\}) + \epsilon \Rightarrow \sigma(\Pi_i) > \sigma_i(\{\}) + \epsilon$. □

Theorem 4.2. *Given a vertex variable V_i, a candidate parent set Π_i can be safely pruned if there exists an instance $I_j = (v_1, \dots, v_n) \in I$, such that $v_i = 1$ and $v_k = 0$ for all $V_k \in \Pi_i$.*

Proof. By Equation (4.1), we must have $P(V_i = 1 \mid V_k = 0 \ \forall k \in \Pi_i) = 0$ for a noisy-OR node, which the existence of instance I_j contradicts. □

This theorem has particular applications for sparse instances, as the following corollaries show.

Corollary 4.1. *For a vertex variable V_i with an instance $I_j = (v_1, \dots, v_n) \in I$, such that $v_i = 1$ and $v_k = 0$ for all $V_k \in V \setminus \{V_i\}$, all candidate parent sets except the empty set can be safely pruned.*

Corollary 4.2. *For a vertex variable V_i with an instance $I_j = (v_1, \dots, v_n) \in I$, such that $v_i = v_k = 1$ for some vertex variable V_k and $v_\ell = 0$ for all $V_\ell \in V \setminus \{V_i, V_k\}$, all candidate parent sets can be safely pruned that are not the empty set and do not contain V_k.*

4.2.3 Overall Algorithm for Structure Learning

Algorithm 4.2 shows our overall algorithm for ϵBNSL, a principled way to automatically select between full CPTs and noisy-OR relations, given a dataset and an approximation factor ϵ. Note that Step 1 and 2 can be performed in parallel.

4.3 Experimental Evaluation

In this section, we show the accuracy of Algorithm 4.1[1] in computing the noisy-OR parameters for synthetic BNs with embedded noisy-OR relations. We also show significant presence of noisy-OR relations in standard benchmark networks. Finally, we test the performance of our learned networks against ground truth networks. All experiments are conducted on computers with 2.2 GHz Intel E7-4850V3 CPUs. Each experiment is limited to 64 GB of memory and 24 hours of CPU time.

[1]Code available at https://github.com/CharupriyaSharma/eBNSLNoisyOR

Algorithm 4.2 Noisy-OR BN Structure Learning

```
    Input: A dataset $I$ over a set of random variables $\mathbf{V}$ with $N$ instances, and an $\epsilon \in \mathbb{R}^+$.
    Output: Set of credible networks.
1:  Step 1: Full CPT representations. Determine the BIC scores when fitting a full CPT for all candidate parent sets that could not be pruned with pruning rules from Section 3.3 using Equation 2.2.1.
2:  Step 2: Noisy-OR Representations. Determine the BIC scores when fitting a noisy-OR relation for all candidate parent sets that could not be pruned using our pruning rules in Section 4.2.2. Here, the noisy-OR parameters are fit using Algorithm 4.1, which minimizes the KL divergence between the full-CPT and the noisy-OR CPT. These parameters are used to compute the noisy-OR BIC score.
3:  Step 3: Merge. Merge the sets of scores from Step 1 and Step 2, using pruning rules Lemma 3.1 and Theorem 3.3, into a set of scores for candidate parent sets for each variable in the dataset. During merging scores of a variable $V_j$, we have to examine only cases where a candidate parent set $\Pi_j$ belongs to the set of BIC scores and its superset $\Pi'_j$ belongs to the noisy-OR BIC scores and vice versa.
4:  Step 4: Find credible networks. The scores obtained in Step 3 are used to learn the set of credible networks using a developmental version of GOBNILP [26], gobnilp_dev (see Section 3.4.1), which can be used to solve the $\epsilon$BNSL problem and collect all the networks in the credible set for the given approximation factor $\epsilon$.
```

4.3.1 Recovery of Noisy-ORs in Synthetic Datasets

To evaluate the accuracy of Algorithm 4.1 in finding the noisy-OR parameters and minimizing conditional KL divergence, we used synthetic BNs which consisted of a single noisy-OR. The parent set sizes were in the range $\{2, \ldots, 7\}$, all parent nodes had priors of 0.5, and the noisy-OR parameters $\mathbf{q} = q_1, \ldots, q_{|\Pi|}$ in the ground truth were uniformly sampled from the set $\{0.01, 0.02, \ldots, 0.99\}$. Thirty tests were performed at each parent set size.

We randomly generated datasets from the synthetic BNs with 100, 500 and 1000 instances, respectively. Algorithm 4.1 was applied to a dataset and the noisy-OR parameters estimated by the algorithm were compared against the parameters in the ground truth network (see Table 4.1). As well, the conditional KL divergence was computed between the noisy-OR CPT for the estimated parameters and the noisy-OR CPT for the ground truth parameters (see Equation 4.1). We also compared our results against the expectation-maximization algorithm for noisy-OR proposed by [108], the code for which was supplied by the author. As shown in Table 4.1, Algorithm 4.1 estimated the ground truth param-

Table 4.1: (Top) Median relative error in noisy-OR parameters and (bottom) median conditional KL divergence of noisy-OR CPTs learned by our Algorithm 4.1, denoted KL, and Vomlel's [108] expectation-maximization algorithm, denoted EM, from ground truth for various parent set sizes.

Parent	$N = 100$		$N = 500$		$N = 1000$	
Size	KL	EM	KL	EM	KL	EM
2	**0.16**	1.07	**0.07**	1.07	**0.05**	1.00
3	**0.21**	1.18	**0.09**	1.11	**0.07**	1.07
4	**0.27**	1.04	**0.11**	1.27	**0.07**	1.26
5	**0.25**	1.50	**0.11**	1.54	**0.08**	1.57
6	**0.34**	1.99	**0.16**	2.04	**0.10**	2.06
7	**0.41**	2.09	**0.24**	2.03	**0.16**	1.99

Parent	$N = 100$		$N = 500$		$N = 1000$	
Size	KL	EM	KL	EM	KL	EM
2	**0.04**	0.05	**0.01**	**0.01**	**0.00**	0.01
3	**0.13**	0.32	**0.02**	0.05	**0.01**	0.03
4	**0.33**	1.32	**0.06**	0.24	**0.03**	0.12
5	**1.02**	4.91	**0.18**	1.20	**0.07**	0.55
6	**1.33**	9.02	**0.33**	3.48	**0.16**	2.06
7	**2.54**	12.08	**1.07**	11.68	**0.60**	6.55

eters with significantly higher accuracy than the EM algorithm. Algorithm 4.1 also had much lower conditional KL divergence.

4.3.2 Presence of Noisy-OR Relations in Standard Benchmarks

To evaluate the ability of our overall algorithm for ϵBNSL (Algorithm 4.2) to learn networks with noisy-OR relations, we used standard datasets from the UCI Machine Learning Repository (`https://archive.ics.uci.edu/`). The datasets used were all binary or made binary.

The overall algorithm for ϵBNSL was applied to a dataset to learn the set of credible networks using a Bayes factor, BF, of 20. Out of the 13 (in a total of 16) benchmarks the algorithm was able to solve, nine benchmarks showed a presence of noisy-OR relations

Table 4.2: Total number of variables where a noisy-OR relation is selected (m), and average (ave.) and maximum (max.) percentage of networks in the set of credible networks that select noisy-OR relations for these nodes, for various benchmarks, where n is the number of variables in the dataset or network and N is the number of instances in the dataset. OT indicates a dataset that could not be solved within the time limit.

Dataset	n	N	m	ave.	max.
adult	14	32,561	0	0.0	0.0
nltcs	16	3,236	0	0.0	0.0
msnbc	17	58,265	0	0.0	0.0
zoo	17	101	7	41.9	99.4
letter	17	20,000	OT	OT	OT
hepatitis	20	155	10	76.9	100.0
parkinsons	23	195	11	51.2	100.0
sensors	25	5,456	OT	OT	OT
autos	26	159	13	76.0	100.0
horse	28	300	3	97.4	100.0
flag	29	194	10	77.6	100.0
wdbc	31	569	OT	OT	OT
soybean	36	266	9	86.1	100.0
alarm	37	1,000	2	28.8	56.4
bands	37	277	8	63.4	100.0
spectf	45	267	0	0.0	0.0

(see Table 4.2). Specifically, these nine benchmarks had two or more nodes that were assigned noisy-OR relations in at least 28.8% of the networks in the credible set. Also, seven benchmarks had at least one node that was assigned a noisy-OR relation in all of the networks in the credible set. Note that some benchmarks, such as hepatitis and parkinsons, select noisy-OR relations for around half of their nodes, which shows that using only full CPTs could have resulted in overfitting. Further, optimal BNs containing noisy-OR relations were consistently found to have better scores than that of optimal networks found using only full CPTs. We also examined the effectiveness of the pruning rules (Steps 2 and 3 of the algorithm). On these benchmarks, the rules safely pruned away from 89.17% to 99.99% of the candidate parent sets, showing that the pruning rules are highly effective.

4.3.3 Performance on Ground Truth Networks

To further evaluate our overall algorithm for ϵBNSL (Algorithm 4.2), we used real-world BNs from the Bayesian Network Repository[2]. The variables in the networks were made binary and their corresponding CPTs compressed by combining similar categories. For example, for a variable taking one of {LOW, MEDIUM, HIGH}, the MEDIUM category was combined with LOW, and the variable could now take one of {LOW, HIGH} (see Table 4.3; BNs without a _b suffix were already binary). From each ground truth network, we randomly generated datasets with 100, 500, and 1000 samples. We then ran our structure learning algorithm on the datasets to learn the set of credible networks, fixed the CPT parameters using maximum likelihood estimation and measured relative inference error against the ground truth network.

Table 4.3 shows the median relative inference error of the best scoring and the worst scoring networks in the set of credible networks, as well as that of the best-scoring network with full CPTs (i.e., not containing noisy-OR relations), against that of the ground truth network. Overall the inference error of the best scoring network is comparable to that of the full CPT. Somewhat surprisingly, the error for the worst scoring network can be smaller than for the best scoring network or the full CPT.

Table 4.3: Median relative inference error for the best and worst scoring network in the set of credible networks learned by Algorithm 4.1 and the full CPT against the ground truth network. The datasets with N instances were generated from various ground truth BNs with n nodes.

Bayesian		$N = 100$			$N = 500$			$N = 1,000$		
network	n	best	worst	CPT	best	worst	CPT	best	worst	CPT
earthquake	5	0.03	0.91	0.00	0.02	0.98	0.00	0.26	0.53	1.00
survey_b	6	0.05	0.69	0.00	0.02	0.74	0.00	0.01	0.75	0.00
asia	8	0.04	0.13	0.92	0.04	0.92	0.02	0.02	0.90	0.08
sachs_b	11	0.43	0.68	0.18	0.70	0.60	0.21	0.68	0.62	0.01
child_b	20	0.05	0.91	0.01	0.05	0.88	0.07	0.05	0.85	0.04
insurance_b	27	0.67	0.72	0.70	0.65	0.71	0.68	0.65	0.68	0.68
alarm_b	37	0.04	0.99	0.01	0.08	0.99	0.05	0.05	OT	0.06

To perform inference on our learned set of credible networks, we generated evidence

[2] www.bnlearn.com/bnrepository

for 10% of the variables in the network. The variables were randomly selected. For one trial, we selected a state of every variable in the evidence, which was set according to the variable's posterior probability distribution in the model, conditional on the evidence observed up till this point. Then, we computed the posterior probability distributions over the non-evidence variables for our learned network and for the ground truth network. The inference errors were the differences between these values. We repeated the described procedure 1000 times for each of the networks. Inference was performed using JavaBayes[3], which was extended to take in an evidence file and two BNs for comparison. Our results are consistent with [113], who show that in three real-world Bayesian networks, noisy-OR/MAX relations were a good fit for up to 50% of the CPTs in these networks and that converting some CPTs to noisy-OR/MAX relations gave good approximations when answering probabilistic queries.

4.4 Summary

Existing successful approaches for learning Bayesian networks from data use the well-known score-and-search approach. In this chapter, we extended the score-and-search approach to simultaneously learn the best global structure and the best local structure when the choice is either a full CPT or a noisy-OR relation for a candidate parent set of a variable in the network. We showed how to score a causal noisy-OR relation for a candidate parent set by fitting the best possible noisy-OR to the data, and we showed how to effectively prune the search space while maintaining the optimality of the networks that are learned. Our experimental results provide evidence of the effectiveness of our approach. In particular, it was found that noisy-OR relations appeared in a significant proportion of the learned networks, for well known datasets.

[3] www.cs.cmu.edu/~javabayes

Chapter 5

Local Structure: Neural Networks

Previous work has shown that incorporating structured representations of CPDs into the score-and-search approach can improve the accuracy of the learned graph. Several useful forms of local structure have been identified in the literature but thus far the score-and-search approach has only been extended to handle variations of decision-tree representations of local structure (see the review of related work and discussion in Section 1.1.3). In the previous chapter, we showed how to learn CPDs with a widely used local structure, noisy-OR, within the score-and-search framework and showed that these relations can appear often in standard benchmarks (Section 4.3). However, linear models like noisy-OR can fail to model non-linear relationships between variables. In this chapter, we propose the first score-and-search approach for learning Bayesian networks with neural network relations as possible representations for CPDs, which enable us to learn non-linear relationships without imposing any rigid constraints on the structure. Importantly, we simultaneously learn both the global structure in the form of the underlying DAG and the local structure in the CPDs, we place no a priori constraints on the global structure, and we exactly determine all networks within a given factor of optimal.

The remainder of this chapter proceeds as follows.

- We provide an algorithm to learn neural network relations for candidate parent sets, as well as score them simultaneously (Section 5.1).
- We extend our credible set score-and-search approach to include neural-network relations. Our approach controls the degradation by specifying a Bayes factor (BF) [58] that measures how far a BN can deviate from the optimal network, and so only near-optimal networks with neural-network relations are learned in a principled manner (Section 5.1).

- We show empirically that our approach to learning parameters of neural networks to represent CPDs compares favourably with approaches based on decision trees and performs particularly well in instances with low amounts of data (Section 5.2).

5.1 Neural Networks to Represent CPDs

Let us consider the definition of the BIC score for a candidate parent set Π_i of V_i in a discrete Bayesian network. We can associate a score to a candidate parent set Π_i of V_i, when the associated full CPT θ_i is given, as follows,

$$\sigma_{BIC}(\Pi_i) = -L(\theta_i) + t(\Pi_i) \cdot w \tag{5.1}$$

where the $L(\theta_i)$ term measures the log likelihood of the data given the candidate parent set and the term $t(\Pi_i) = r_{\Pi_i}(r_i - 1)$ is the number of parameters needed to specify the full CPT for the candidate parent set. As the size of the parent set and the number of free parameters grow, BIC places heavy penalties on the local scores, which can prevent finding the ground truth for networks with larger parent sets. As has been previously shown, there may be structure in the CPDs and this structure might be representable with far fewer parameters than the exponential number of parameters needed for the full CPT.

5.1.1 Learning CPDs with Neural Networks

For modeling CPDs, we will use *multilayer perceptrons (MLPs)* (see, e.g., [46]). MLPs are a fully connected class of feedforward neural networks where the mapping between inputs and output is non-linear. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a non-linear activation function, which defines the output of that node given its inputs. Data moves forward through the layers, and the modeling of the input instance within the MLP involves changing connection weights after each instance of data is processed, based on the error in the output compared to the expected result using a loss function. The ability of MLPs to model data that is not linearly separable can be leveraged to find complex structures within a CPD for a candidate parent set Π_i of V_i. Note that in an MLP representation of a CPD, the input layer and the output layer are determined by the parent set Π_i and the child V_i variable, respectively.

Let $\mathcal{N}$ be the set of all MLPs over a variable V_i and a candidate parent set Π_i. The goal is to find the best scoring neural network model of the data; i.e.,

$$\sigma_{BIC}(\Pi_i) = \min_{nn \in \mathcal{N}} (-L(nn, \Pi_i) + t(nn, \Pi_i) \cdot w), \tag{5.2}$$

where $L(nn, \Pi_i)$ is the log likelihood of the data given the neural network model nn and $t(nn, \Pi_i)$ is the number of parameters (weights) in nn. There are both practical and theoretical issues with determining Equation 5.2, all the more so as some of the pruning rules no longer apply as they depend on the assumption that the CPD is represented as a full CPT. We discuss this further in Section 5.2 below.

Algorithm 5.1 Neural Network Fit

Input: Node V_i, candidate parent set Π_i, and a one-hot encoded dataset I of instances.
Parameter: Accuracy threshold δ.
Output: Size of the best fitting neural network.

```
1:  $CPTSize$ = size of a full CPT representation
2:  $baseline$ = mean accuracy of a classifier that predicts the most frequent child value
3:  $nnSize = 0$
4:  $nnNodes = 1$
5:  $accuracy = 0$
6:  while $nnSize < CPTSize$ do
7:      train an MLP $nn$ with a single hidden layer of $nnNodes$ neurons
8:      $prevAccuracy = accuracy$
9:      $accuracy$ = mean score of $nn$ on training data
10:     $nnSize$ = number of parameters in $nn$
11:     if $accuracy > baseline$ and $|accuracy - prevAccuracy| < \delta$ then
12:         break
13:     end if
14:     $nnNodes = nnNodes + 1$
15: end while
16: return min($CPTSize$, $nnSize$)
```

The alternative that we pursue here is to approximate Equation 5.2 by first fixing the $L(\theta_i)$ term in Equation 5.1 to the maximum likelihood estimation given by the full CPT and then minimizing the $t(\Pi_i)$ term by fitting the smallest neural network to represent the CPT. If we reduce the number of free parameters with our neural network, the penalty term reduces, making the candidate parent sets score better, as well as enabling us to find more structured representations of the CPTs. Here, given a parent set for a child variable that we wish to score and associated data, we attempt to fit a neural network to the data such that the network is as small as possible yet the parent set predicts the child variable with maximum accuracy. The minimum of the number of entries in the CPT and the number of weights in the neural network is then used as the number of free parameters in the $t(\Pi_i)$ term of the BIC score.

We propose a simple greedy algorithm (see Algorithm 5.1) to find the size of the neural network that best fits the data in that the network is as small as possible yet the parent set predicts the child variable with maximum accuracy. Given a node V_i and candidate parent set Π_i, and a one-hot encoded dataset I, we train a simple neural network. The neural networks is constructed as an MLP with a single hidden layer where each node uses the logistic function, and a softmax is placed on the output layer. Starting with zero nodes, we keep on adding nodes to the hidden layer of the neural network while we can improve the mean accuracy of the neural network and the total number of weights needed for the neural network CPT does not exceed the number of parameters needed by the full CPT.

5.1.2 Learning CPDs with Decision Trees

Decision trees have been used to represent context specific conditional independence in CPDs (see the discussion in Section 1.1.3 and Section 2.4). Let $\mathcal{T}$ be the set of all decision trees over a variable V_i and a candidate parent set Π_i. The goal is to find the best scoring decision tree model of the data; i.e.,

$$\sigma_{BIC}(\Pi_i) = \min_{dt \in \mathcal{T}}(-L(dt, \Pi_i) + t(dt, \Pi_i) \cdot w), \tag{5.3}$$

where $L(dt, \Pi_i)$ is the log likelihood of the data given the decision tree model dt and $t(dt, \Pi_i)$ is the number of probabilities specified by dt (see, for example, Figure 2.6, where the decision tree there specifies eight probabilities) [39].

Algorithm 5.2 describes a decision tree algorithm that is closely similar to the algorithm given above for neural networks: the algorithm approximates Equation 5.3 by first fixing the $L(\theta_i)$ term in Equation 5.1 and then minimizing the $t(\Pi_i)$ term by fitting the smallest decision tree to represent the CPT. Given a node V_i and candidate parent set Π_i, and a one-hot encoded dataset I, we learn a decision tree. Starting with a decision tree consisting of a single leaf, we keep on adding leaf nodes to the decision tree while we can improve the mean accuracy of the decision tree. The minimum of the number of entries in the CPT and the number of probabilities specified by the decision tree is then used as the number of free parameters in the $t(\Pi_i)$ term of the BIC score.

Algorithm 5.2 Decision Tree Fit

Input: Node V_i, candidate parent set Π_i, and a one-hot encoded dataset I of instances.
Parameter: Accuracy threshold δ.
Output: Size of the best fitting decision tree.

```
1: $CPTSize$ = size of a full CPT representation
2: $baseline$ = mean accuracy of a classifier that predicts the most frequent child value
3: $dtSize = 0$
4: $dtNodes = 1$
5: $accuracy = 0$
6: while $dtSize < CPTSize$ do
7:     learn a decision tree $T$ with $dtNodes$ leaf nodes from $I$
8:     $prevAccuracy = accuracy$
9:     $accuracy$ = mean score of $dt$ on training data
10:    $dtSize$ = number of probabilities specified by $dt$
11:    if $accuracy > baseline$ and $|accuracy - prevAccuracy| < \delta$ then
12:        break
13:    end if
14:    $dtNodes = dtNodes + 1$
15: end while
16: return min($CPTSize$, $dtSize$)
```

Algorithm 5.3 describes the algorithm [39] that we use to more directly approximate the best possible BIC score of a decision tree representation for a given node and candidate parent set. We build a decision tree by branching on the values of the variables in the candidate parent set. The leaf nodes in the tree will contain a mixture of child values, which give the probabilities for each value of the child. We fit decision trees with increasing numbers of leaf nodes to the given dataset, until the BIC score no longer improves with additional child nodes.

Algorithm 5.3 Decision Tree Score

Input: Node V_i, candidate parent set Π_i, and a one-hot encoded dataset I of instances.
Output: The best BIC score of a decision tree representation

```
1: $parentInstances$ = number of parent set instantiations
2: $CPTSize$ = size of a full CPT representation
3: $treeLeaves = 0$
4: $bestScore = +\infty$
5: for $\ell \in \{1, \ldots, parentInstances\}$ do
6:    learn a decision tree $T_\ell$ with $\leq \ell$ leaf nodes from $I$
7:    $prevTreeLeaves = treeLeaves$
8:    $treeLeaves$ =number of leaves of $T_\ell$
9:    if $treeLeaves = prevTreeLeaves$ then
10:       break
11:   end if
12:   $score$ = BIC($T_\ell$, $I$)
13:   if $bestScore > score$ then
14:       $bestScore = score$
15:   end if
16: end for
17: return $bestScore$
```

5.1.3 Overall Algorithm for Structure Learning

Algorithm 5.4 shows our overall credible set algorithm for ϵBNSL, a principled way to automatically determine the BIC penalty accurately using neural networks or decision trees given a dataset and an approximation factor ϵ. The same algorithm can also be used for CPTs represented with other structures, for example, decision trees.

Note that here, we use pruning rules for BIC introduced in Section 3.3, which includes an effective pruning rule using an upper bound on the size of possible parent sets (Theorem 3.5). However, this pruning rule assumes that the CPD is represented as a CPT. Nevertheless, we can use these pruning rules in a heuristic manner. In Section 5.2 we show that this does not have a significant impact on accuracy in comparison to the performance of full CPTs.

Algorithm 5.4 BN Structure Learning with Neural Networks/Decision Trees

```
    Inputs: A choice of structured CPT learning algorithm $Alg \in${Algorithm 5.1, Algorithm 5.2}, a dataset $I$ over a set of random variables $\mathbf{V}$, and an $\epsilon \in \mathbb{R}^+$.
    Outputs: Set of credible networks.
1:  Step 1: Full CPT representations. Determine the BIC scores when using the best fitting CPT learned by $Alg$ for all candidate parent sets that could not be pruned with pruning rules from Section 3.3.
2:  Step 2: Find credible networks. The scores obtained in Step 1 are used to learn the set of credible networks using a developmental version of GOBNILP [26], gobnilp_dev [67], which can be used to solve the $\epsilon$BNSL problem and collect all the networks in the credible set for the given approximation factor $\epsilon$.
```

5.2 Experimental Evaluation

In this section we compare our neural network representation algorithm to a full CPT representation and a decision tree representation.

We considered a wide selection of datasets from the UCI repository[1] and networks from the bnlearn Bayesian network repository[2] (see Table 5.1). We preprocessed the UCI datasets using a k-nearest neighbor imputation algorithm, with $k = 5$, to fill in missing values and a supervised discretization method [35] based on the MDL principle to discretize continuous variables.

For evaluating the neural network method on the task of structure learning using BIC, we used a total of 87 ground truth BNs: 7 ground truth BNs came from the bnlearn repository and a further 80 ground truth BNs were constructed following a similar approach to Liu et al. [69] by (i) scoring each of the 16 UCI datasets using each of the five scoring functions AIC, BDeu, BIC, qBDJ, and qNML (ii) learning an optimal network structure from each scored dataset, and (iii) and fitting the parameters to each structure to give a final Bayesian network. Given these ground truth BNs, we used the logic sampling function `rbn` from the bnlearn R package [90] to generate random samples of sizes $N = 50$, 100, 500, and 1,000 from the bif files. We collected three samples for each dataset size N, for a total of 12 samples for each ground truth BN. Thus, there are 87 $\times$ 12 instances all together, each associated with a ground truth DAG. Having ground truth DAGs allows us to measure the accuracy of each approach. The number of variables n used in our experiments ranged

[1] https://archive.ics.uci.edu/ml
[2] https://www.bnlearn.com/bnrepository/

from 10 to 48. Structure learning was done to solve ϵBNSL using a fixed threshold of 0.5 using *gobnilp_dev* (See Section 3.4).

Table 5.1: UCI datasets (*left*, *middle*) and bnlearn Bayesian networks (*right*), where n is the number of variables in the dataset or network, and N is the number of instances in the original UCI dataset.

UCI dataset	n	N
shuttle	10	58,000
census income	14	48,842
letter	17	20,000
online shopping	18	12,330
lymphography	19	148
hepatitis	20	155
parkinsons	23	195
credit card	24	30,000

UCI dataset	n	N
robot navigation	25	5,456
horse colic	27	368
steel	28	1,941
flags	29	194
breast cancer	31	569
soybean	36	683
biodeg	42	1,055
spectf heart	45	267

network	n
sachs	11
child	20
insurance	27
water	32
mildew	35
alarm	37
barley	48

The scoring computations were conducted on the Graham cluster of SHARCNET[3] and the structure learning experiments were conducted on either SHARCNET, a shared server with 346 GB RAM and Intel Xeon Gold 6148 CPUs at 2.4 GHz, or dedicated machines with 128 GB RAM and Intel Xeon Silver 4214R CPUs at 2.4 GHz. For scoring the datasets memory usage was limited to 64 GB and for structure learning a limit of 128 GB was imposed. For both scoring and learning, a computation time limit of 24 hours was imposed for each instance.

Algorithm 5.1 for finding the best fitting neural network, Algorithm 5.2 for finding the best fitting decision tree, and Algorithm 5.3 for finding the best scoring decision tree were implemented using scikit-learn [81] as a basis. For neural networks, the activation function was set as logistic, as experimentation with other activation functions led to poorer or similar results. The network predicts the probabilities of each value of the child variable and the loss function to minimize in learning was the cross entropy (logarithmic loss) function, which minimizes the difference between the predicted probability distribution and the distribution of probabilities in the training dataset; i.e., the maximum likelihood estimate [46]. We found that the regularization term did not affect the performance significantly, and we used the default $\alpha = 0.0001$. As well, we used the default of 200 as the maximum number of epochs. For our experiments, we approximated the size of the neural network as follows

[3] https://www.sharcnet.ca

(Algorithm 5.1, Line 10). Let m be the number of nodes in the single hidden layer. Let d_{X_i} be the domain size of the ith parent node X_i and let d_Y be the domain size of the child node. The size of the neural network is approximated to be $m \cdot \sum_{i=1} d_{X_i} + (d_Y - 1)$. For decision trees, we used the default gini splitting criterion and scikit learn's ability to grow a decision tree with a given number of leaf nodes in a best-first fashion.

We begin by discussing what did *not* work. Our implementation of Algorithm 5.3, which directly approximates Equation 5.3 and most closely corresponds to the proposal by Friedman and Goldszmidt [39], did not scale and was unable to score most of our datasets within the memory (64 GB) and time limits (24 hours) we imposed. Friedman and Goldszmidt [39] used the method within a greedy hill-climbing search; i.e., only select parent sets were scored. In our credible set approach, we must score all possible parent sets, a much more costly endeavour. Nevertheless, the limited experimentation that we were able to perform identified an important issue that did not arise in previous work. Given a parent set Π_i and a variable $V_j \notin \Pi_i$, $\sigma_{BIC}(\Pi_i) = \sigma_{BIC}(\Pi_i \cup \{V_j\})$ when using Equation 5.3, whenever V_j is irrelevant; i.e., adding irrelevant variables to a parent set does not change its score. This is problematic when doing knowledge discovery as it leads to many extra edges or false positives. For the decision tree, we also implemented a version of Algorithm 5.2 that used minimal cost-complexity pruning [11] to avoid overfitting but this did not lead to improved performance. Finally, we also attempted to directly approximate Equation 5.2 using an algorithm closely similar to Algorithm 5.3 that used the Keras/Tensorflow implementation of weight pruning to approximate the neural network with the minimal number of parameters. However, this too did not scale to most of our datasets within the memory and time limits we imposed.

We compare the results of knowledge discovery using the credible set approach given in Algorithm 5.4 when using (i) Algorithm 5.1 for finding the best fitting neural network, (ii) Algorithm 5.2 for finding the best fitting decision tree, and (iii) the full CPT, in Figures 5.1 & 5.2. For each method, we plot results for the BIC scoring function. Each "performance curve" is obtained by solving each of the 87×12 instances, evaluating the learned DAG against ground truth using the performance measure, and sorting the results into ascending order. For the performance measures (e.g., F_β) that take an additional parameter, we use $\alpha, \beta = \frac{1}{4}, \frac{1}{2}, 1, 2,$ and 4. The performance curves can then be compared and we look for whether one method dominates another method.

Figure 5.1 compares the performance of the neural network and the decision tree methods in approximating the BIC penalty against the penalty computed using the full CPT when using the metrics SHD and multi-class F_β score over CPDAGs. For the SHD, the neural network and the CPT methods have similar results, with the neural network method showing a small degradation over the CPT method. However, the decision tree method performs poorly under this metric and the neural network and CPT methods dominate the decision tree method. A more fine-grained analysis (not shown) shows that for the SHD metric the performance of the decision tree method worsens as the size N of the dataset grows. For the multi-class F_β score, the neural network improves over the CPT method more often than it degrades performance, with the degree of improvement offered by the neural network method being on average larger than the degree of loss in performance. However, the performance of the decision tree method is mixed on this metric, with the improvements and the losses in performance being about equally matched. A more fine-grained analysis (not shown) shows that for the multi-class F_β score the performance of the decision tree method worsens as the size N of the dataset grows.

Figure 5.2 compares the performance of the neural network and the decision tree methods in approximating the BIC penalty against the penalty computed using the full CPT when using the metrics misclassification cost and F_β score over skeletons. A similar analysis holds. For the misclassification cost, the neural network and the CPT methods have similar results, with the neural network method showing a small degradation over the CPT method. However, the decision tree method performs poorly under this metric and is dominated by both the CPT and neural network methods. A more fine-grained analysis (not shown) again shows that for the misclassification cost metric the performance of the decision tree method worsens as the size N of the dataset grows. For the F_β score, the neural network improves over the CPT method considerably more often than it degrades performance, with the degree of improvement offered by the neural network method being on average larger than the degree of loss in performance. However, the performance of the decision tree method is once again mixed on this metric, with the improvements and the losses in performance being about equally matched. A more fine-grained analysis (not shown) again shows that for the multi-class F_β score the performance of the decision tree method worsens as the size N of the dataset grows.

To summarize, over the four performance metrics, two for CPDAGs and two for skeletons, the performance of the decision tree method is mixed on two of the metrics and is dominated by both the neural network method and the CPT method on the other two metrics. Comparing the neural network method and the CPT method, the neural network method slightly degrades performance on two of the metrics and offers advantages on the other two metrics. To quantify the advantage, we next do a more fine-grained comparison of the CPT and neural network methods. Tables 5.2–5.5 break down the results for the full CPT and neural network methods by size N of the dataset and by the different types of ground truth instances, with the UCI instances being separated by type of scoring function used to construct the ground truth. The tables show the quartile percentage change when comparing the two methods.

Table 5.2 compares the full CPT to the neural network method for SHD, and shows the 25th percentile (Q1), median (Q2) and 75th (Q3) percentile of the percentage change. Here, positive values favour the full CPTs method, while negative values favour the neural network method. We see the neural network method is more competitive with full CPTs in the low data case—i.e., $N = 50$ and $N = 100$—and we see a reduced overall competitiveness of the neural network method as the amount of data increases.

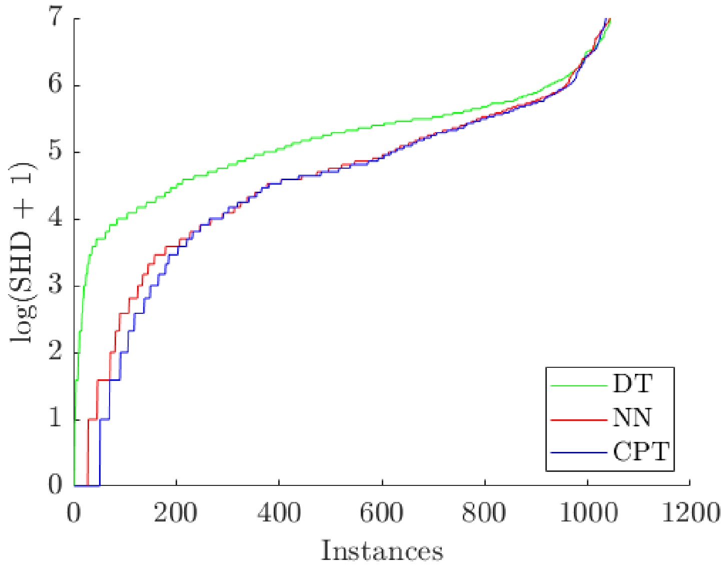
Table 5.3 compares the full CPT to the neural network method for the multi-class F_1 score. Here, negative values favour the full CPTs method, while positive values favour the neural network method. In cases of low data, i.e. $N = 50$ and $N = 100$, we see the overall results being significantly skewed in the favour of our neural network method. The advantage is somewhat reduced when we add more data, although we still see the neural network method perform well with some sets of instances.

Table 5.4 compares the full CPT to the neural network method for misclassification cost FP+FN. Here, positive values favour the full CPTs method, while negative values favour the neural network method. We see the neural network method is more competitive with full CPTs in the low data case—i.e., $N = 50$ and $N = 100$—and we see a reduced overall competitiveness of the neural network method as the amount of data increases.

Table 5.5 compares the full CPT to the neural network method for F_1 score. Here, negative values favour the full CPTs method, while positive values favour the neural network method. In cases of low data, i.e. $N = 50$ and $N = 100$, we see the overall results being significantly skewed in the favour of our neural network method. The advantage is somewhat reduced when we add more data, although we still see the neural network method perform well with some sets of instances.

5.3 Summary

In this chapter, we extended our score-and-search approach with model averaging to simultaneously learn the DAG and the local structure of the CPD using neural network representations. The use of neural networks allows modeling of high-order interactions without needing an exponential number of parameters as in the CPT. Empirically, we show that our approach compares favourably with decision tree and full CPT approaches, and performs well on instances with low amounts of data while scaling to medium sized networks.



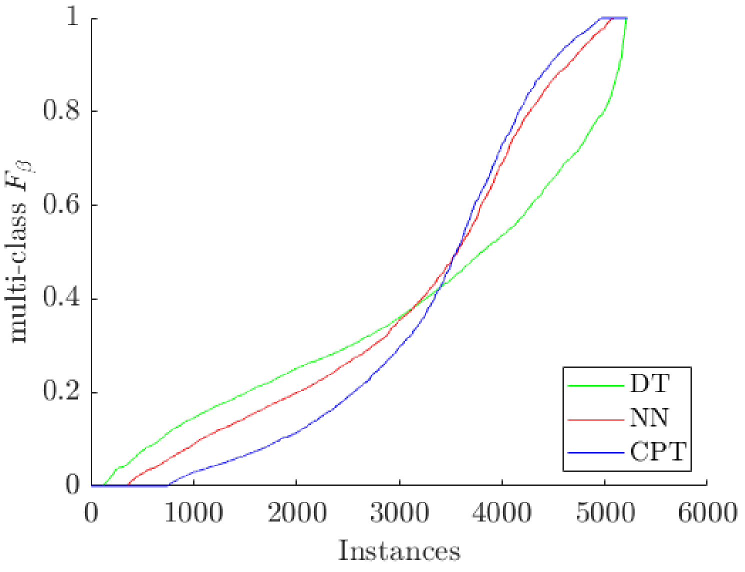
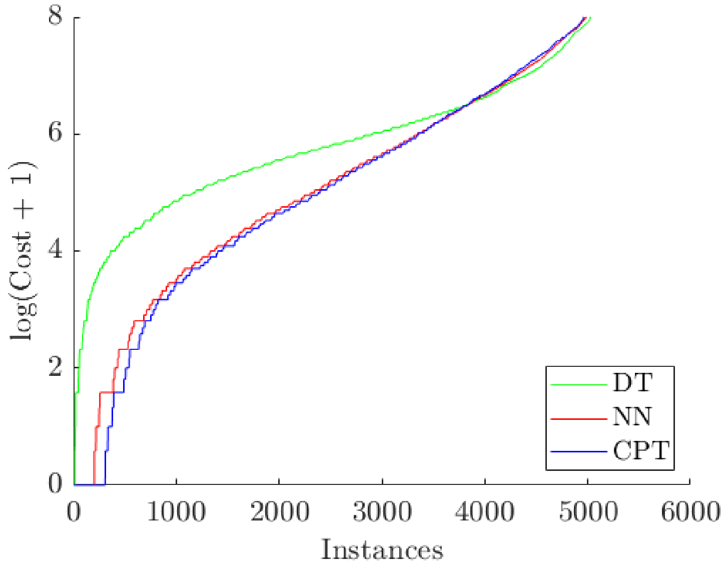


Figure 5.1: Comparison of scoring with conditional probability tables (CPT), scoring with neural networks (NN), and scoring with decision trees (DT) using structural Hamming distance over CPDAGs (top, lower values are better) and multi-class F_β score over CPDAGs (bottom, higher values are better) as the performance measure. All methods used a fixed threshold of 0.5.



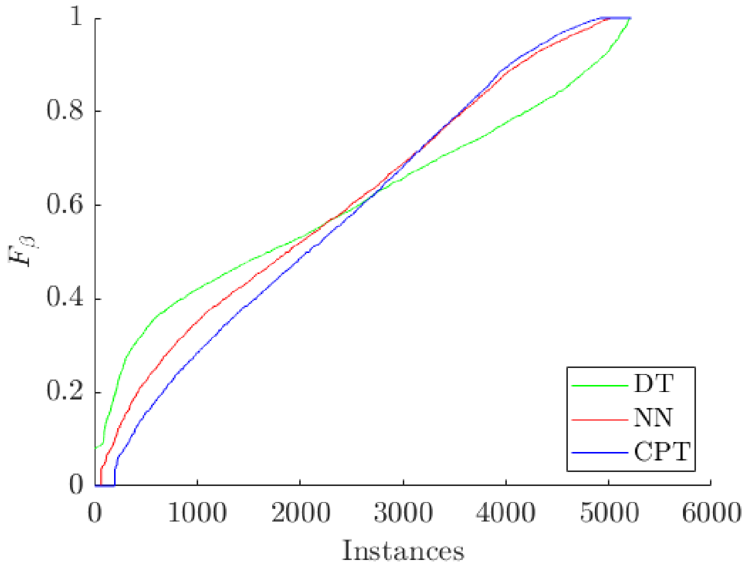


Figure 5.2: Comparison of scoring with conditional probability tables (CPT), scoring with neural networks (NN), and scoring with decision trees (DT) using misclassification cost $\alpha \cdot \mathrm{FN} + \mathrm{FP}$ over skeletons (top, lower values are better) and F_β score over skeletons (bottom, higher values are better) as the performance measure. All methods used a fixed threshold of 0.5.

Table 5.2: Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using structural Hamming distance over CPDAGs as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Positive values favor CPT, negative values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks.

Ground	$N = 50$			$N = 100$			$N = 500$			$N = 1000$		
truth	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
UCI-AIC	−2.7	0.0	5.0	−6.8	0.0	3.8	−4.1	0.0	5.1	0.0	3.4	43.8
UCI-BDeu	−4.5	0.0	6.1	−5.0	0.0	0.0	−5.9	0.0	14.1	−14.3	0.0	0.0
UCI-BIC	−4.5	0.0	12.4	−5.8	0.0	14.7	0.0	0.0	121.4	0.0	0.0	411.7
UCI-qBDJ	−3.9	0.0	3.7	−6.6	0.0	1.9	−7.3	0.0	3.2	−8.2	0.0	3.2
UCI-qNML	−4.5	0.0	4.5	−7.3	−1.1	3.3	−4.7	0.0	14.9	−4.5	0.0	18.6
bnlearn	0.0	11.3	22.0	3.0	9.6	18.2	9.0	22.9	28.9	14.7	29.5	95.0
Global	−4.2	0.0	6.4	−5.6	0.0	4.0	−2.8	0.0	16.7	−4.0	0.0	23.4

Table 5.3: Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using multi-class F_1 score over CPDAGs as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Negative values favor CPT, positive values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks.

Ground	$N = 50$			$N = 100$			$N = 500$			$N = 1000$		
truth	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
UCI-AIC	0.0	4.8	77.9	−0.9	9.6	337.7	−6.3	0.5	79.9	−10.6	−0.5	2.6
UCI-BDeu	0.0	11.5	243.7	0.0	30.5	146.5	−9.9	0.0	44.8	−1.2	1.6	63.4
UCI-BIC	−6.0	7.4	115.3	−7.1	1.9	90.1	−9.9	−2.0	0.0	−7.6	0.0	0.0
UCI-qBDJ	0.0	25.2	118.0	0.0	18.5	319.2	−1.4	0.0	33.7	−1.9	0.0	44.4
UCI-qNML	0.0	31.3	379.4	0.0	81.3	395.9	−4.8	0.0	75.3	−1.9	0.0	70.9
bnlearn	−28.3	19.3	384.3	−3.9	28.2	763.6	−33.9	0.0	254.1	−57.0	−17.6	145.5
Global	0.0	19.2	161.0	0.0	18.9	189.8	−7.7	0.0	31.3	−5.2	0.0	30.3

Table 5.4: Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using misclassification cost FP + FN over skeletons as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Positive values favor CPT, negative values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks.

Ground	$N = 50$			$N = 100$			$N = 500$			$N = 1000$		
truth	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
UCI-AIC	−5.6	0.0	3.1	−3.9	0.0	3.7	−5.4	0.0	6.5	0.0	2.0	22.6
UCI-BDeu	−2.9	0.0	6.2	−3.7	0.0	6.7	−7.9	0.0	13.2	−7.9	0.0	0.0
UCI-BIC	−8.2	0.0	4.8	−2.2	0.0	19.2	0.0	0.0	91.7	0.0	0.0	137.5
UCI-qBDJ	−6.5	−1.6	2.3	−4.1	0.0	2.0	−3.7	0.0	8.5	−8.7	0.0	8.2
UCI-qNML	−6.4	0.0	2.0	−4.2	0.0	5.0	−4.0	0.0	9.3	−6.9	0.0	11.0
bnlearn	−3.3	4.7	17.2	0.0	7.5	11.4	−3.9	16.7	36.6	0.0	25.9	36.7
Global	−5.5	0.0	4.5	−3.7	0.0	6.7	−4.0	0.0	13.8	−4.1	0.0	19.4

Table 5.5: Comparison of scoring with conditional probability tables (CPT) and scoring with neural networks (NN) using F_1 score over skeletons as the performance measure. At each row, the 25 percentile (Q1), median (Q2), and 75 percentile (Q3) of the percentage change is shown when comparing the methods on a set of ground truth networks and dataset sample sizes of $N = 50, 100, 500, 1000$. Negative values favor CPT, positive values favor NN. $N = 1000$ omits the soybean and biodeg benchmarks.

Ground	$N = 50$			$N = 100$			$N = 500$			$N = 1000$		
truth	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
UCI-AIC	0.0	10.6	28.1	−1.1	2.7	15.7	−1.7	0.0	10.0	−3.3	0.0	1.3
UCI-BDeu	−2.8	0.2	14.1	−2.6	0.0	6.4	−3.5	0.0	6.3	0.0	0.0	5.5
UCI-BIC	−0.6	3.3	24.7	−4.3	0.0	6.8	−3.0	0.0	1.5	−2.4	0.0	0.0
UCI-qBDJ	0.0	10.1	29.1	0.0	2.0	11.9	−2.1	0.0	4.8	−1.0	0.0	7.8
UCI-qNML	0.0	9.4	27.7	0.0	2.6	14.9	−1.6	0.0	4.9	−1.3	0.0	3.8
bnlearn	10.6	23.3	42.2	−3.5	3.6	31.7	−7.8	−2.1	4.7	−7.6	−3.8	0.0
Global	0.0	8.8	25.9	−1.4	1.0	11.2	−2.2	0.0	4.8	−2.0	0.0	2.6

Chapter 6

Local Structure: Multivariate Adaptive Regression Splines

The graph structure of a BN can be learned from data using the well-known score-and-search approach. Previous work has shown that incorporating structured representations of the CPDs into the score-and-search approach can improve the accuracy of the learned graph (see the review of related work and discussion in Section 1.1.3). In this chapter, we present a novel approach capable of learning the DAG of a BN and simultaneously modeling linear and non-linear local probabilistic relationships between variables. We achieve this by a combination of feature selection to reduce the search space for local relationships and extending the score-and-search approach to incorporate modeling the CPDs over variables as multivariate adaptive regression splines (MARS). MARS are polynomial regression models represented as piecewise spline functions. Our approach can effectively prune most candidate parent sets of a variable by leveraging feature learning algorithms without making any a priori assumptions on the structure of the DAG. We show on a set of discrete and continuous benchmark instances that our proposed approach can improve the accuracy of the learned graph while scaling to instances with a large number of variables.

The remainder of this chapter proceeds as follows.

- We introduce a scoring method that can model both linear and non-linear interactions between variables while scoring them simultaneously. The scoring algorithm identifies high-degree variable interactions without a limit on the number of variables or terms in the model and can compute structured representations for both discrete and continuous valued variables (Section 6.1).

- We provide a feature selection method to identify the most promising parents for a node, and scale our score-and-search approach to very large networks with more than 1000 nodes (Section 6.2).
- We show empirically on a set of discrete and continuous benchmark instances that our proposed approach can improve the accuracy of the learned DAG (Section 6.3).

6.1 MARS: Multivariate Adaptive Regression Splines

Linear models like linear and logistic regression are a simple, intuitive and fast to compute way to represent data. However, they impose linear relations on the data which limits their accuracy if the ground truth contains non-linear relationships. Multivariate adaptive regression splines (MARS) [37] are an extension of linear models that combines linear basis functions to model non-linearities and variable interactions. MARS models are of the form,

$$M(x) = c_0 + \sum_{j=1}^{k} c_j B_j(x), \tag{6.1}$$

where each *model coefficient* c_j for $j \in [0, k]$ is a constant, and each B_j is a *basis function*. Basis functions take one of the two forms:

1. A *hinge* function of the form $(x - t)_+ = \max(0, x - t)$ or $(t - x)_+ = \max(0, t - x)$, where t is a constant, called a *knot*.
2. A product of two or more unique hinge functions. This prevents terms of higher degree which are approximated by the hinge functions instead.

Hinge functions are zero over part of their range, and their product is nonzero only over the region where each of the component hinge functions have nonzero values. Thus, data can be partitioned into disjoint regions and processed independently, enabling processing of high dimensional inputs while still being able to capture complex non-linear relationships with high order interactions (see Figure 6.1).

We next show how we use MARS in Bayesian network structure learning. For a variable V_i and a candidate parent set Π_i we write the MARS model for V_i and Π_i as,

$$M(\Pi_i) := c_0 + \sum_{j=1}^{k} c_j B_j(\Pi_i). \tag{6.2}$$

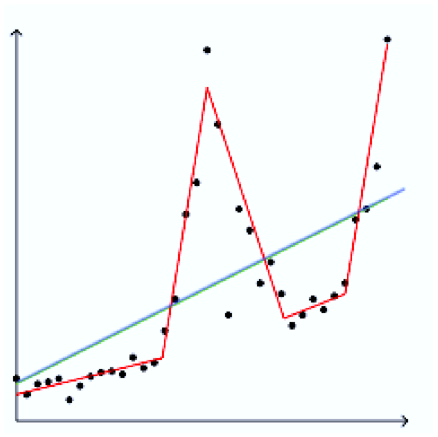


Figure 6.1: MARS (red) and linear regression (blue) models for the same set of points. MARS places a knot at every change of the slope.

For the computation of the model, we are given a dataset of N instantiations $I = \{I_1, \ldots, I_N\}$ for V_i and Π_i; i.e., each $I_j = (v, \pi_1, \ldots, \pi_{|\Pi_i|})$ with $v \in \Omega_i, \pi_k \in \Omega_{\Pi_i}$. To compute the basis functions in $M(\Pi_i)$, the MARS algorithm operates in two phases.

1. A forward stage which starts with the intercept c_0 and proceeds to generate many candidate basis functions in pairs $[(t-x)_+, (x-t)_+]$ with $x \in \Pi_i$. The model coefficients are estimated by minimizing the residual sum-of-squares (RSS),

$$RSS(M(\Pi_i)) = \sum_{j=1}^{N} (y_j - M(I_j))^2 .$$

   A candidate pair of basis functions is added to the model $M(\Pi_i)$ if it reduces the training error of $M(\Pi_i)$; i.e., if $RSS(M_{new}(\Pi_i)) - RSS(M_{old}(\Pi_i)) < 0$. This process continues until there is no significant enough improvement.

2. A backward stage that is used to address overfitting in $M(\Pi_i)$. The backward stage iterates over functions in $M(\Pi_i)$ and prunes terms until the subset of the terms that provide the best score remains. The subsets are scored by using a generalized cross-validation (GCV) score,

$$GCV(M(\Pi_i)) = \frac{RSS(M(\Pi_i))}{((1-\lambda)/N)^2}, \tag{6.3}$$

   where λ is the number of terms in $M(\Pi_i)$.

We call $GCV(M(\Pi_i))$ the local score of V_i in a BN which assigns Π_i as the parent set of V_i. GCV provides a computationally fast approximation to leave-one out cross-validation, for linear fitting under squared-error loss. The GCV score can be shown to be similar to the Akaike information criterion (AIC) score, which is frequently used for model selection [48]. GCV scores are decomposable, meaning for a BN G with parent set $\Pi_1, \ldots, \Pi_i$ assigned to variables $V_1, \ldots, V_n$, the total GCV score can be calculated by,

$$GCV(G) = \sum_{i=1}^{n} GCV(M(\Pi_i)),$$

where $M(\Pi_i)$ is the model computed for representing V_i with candidate parent set Π_i.

6.2 Scaling with Feature Selection

MARS provides a fast way to score a candidate parent set for a child. However, there are an exponential number of candidate parent sets for each node. To scale the search to large networks, it is crucial to score a small subset of candidate parent sets. In this section, we describe our algorithm BN-FS-MARS (Algorithm 6.1) for BNSL, which is based on MARS and a feature selection mechanism.

BN-FS-MARS operates in two phases.

1. Compute candidate MARS models and their GCV scores for all variables and candidate parent sets. We prune candidate parents for a child V_i by using feature selection: we compute an ordering for the set of candidate parents $V \setminus \{V_i\}$ which ranks their importance and use the candidate parent importance rankings to create candidate parent sets.

2. The first phase provides multiple possible parent sets for each variable with an associated score for each parent set. This gives an optimization problem of finding the DAG G that minimizes the total score. We feed the candidate parent sets and scores from the MARS models of phase 1 into a search algorithm to construct a DAG of minimum GCV score. Algorithm 6.1 shows the pseudocode of the algorithm.

The performance of BN-FS-MARS has significant dependence on the quality of the variable importance ranking used in Step 3 of Algorithm 6.1. There are a number of feature selection methods such as random forests [10], decision trees, and the Pearson correlation coefficient that can generate such a ranking (see, e.g., [47] and references therein). The accuracy of the feature selection method used to generate this ranking depends significantly on the amount and type of data we have available. In the next section, we will show the performance of BN-FS-MARS with random forests.

Algorithm 6.1 BN-FS-MARS

```
**Inputs**: A set of random variables $V$ and a dataset $I$ over $V$.
**Outputs**: A DAG $G$ minimizing GCV($G$).
*Phase 1: Compute MARS models and GCV scores for all variables guided by feature selection*
1: $R = \emptyset$, where $R$ is used to collect the scores for Phase 2
2: for each variable $V_i \in V$ do
3:    $\mathbf{\Pi_i}$ := FilterCandidatesFS($V_i$)
4:    for $\ell = 0, \dots, |\mathbf{\Pi}_i|$ do
5:       for parent sets $\Pi_i \subseteq \mathbf{\Pi}_i$, $|\mathbf{\Pi}_i| = \ell$ do
6:          compute MARS model $M(\Pi_i)$                    (see Equation 6.2)
7:          compute $GCV(M(\Pi_i))$                          (see Equation 6.3)
8:          $R = R \cup \{(V_i, \Pi_i, GCV(M(\Pi_i)))\}$
9:          if time limit for $V_i$ reached then
10:            continue for-loop in line 2
11:         end if
12:      end for
13:   end for
14: end for
*Phase 2: Choose a MARS model for each variable to create a DAG that minimizes GCV(G)*
15: Run a search algorithm on $R$ to compute a DAG $G$ minimizing $GCV(G)$
```

Algorithm 6.2 FilterCandidatesFS

```
**Inputs**: A random variable $V_i \in V$, a constant $\ell < n$.
**Outputs**: A candidate parent set $\Pi_i$.
1: Compute a variable importance ranking for $V_i$ using $V \setminus \{V_i\}$
2: Return $\mathbf{\Pi}_i$ := $\ell$-most important variables based on feature selection.
```

6.3 Experimental Evaluation

In this section, we show the performance of our algorithm in computing a DAG for a given dataset.

We tested the performance of our algorithm on synthetic networks generated using the software Tetrad[1]. In addition, we show results on real-world Bayesian networks from the Bayesian Network Repository[2] for large and very large network sizes. We also show results on 100 node networks in the DREAM 4 challenge benchmarks, which are gene regulation networks using the genenetweaver software[3]. Gene regulation networks are a collection of biological regulators that interact with each other, and this interaction can be modeled as a directed graph.

We ran our structure learning algorithm, BN-FS-MARS, on the datasets to learn the DAG and compared the results obtained with tabular CPT using BIC scoring and Greedy Equivalent Search [22] using BIC scoring, where Greedy Equivalent Search is a score-based local search algorithm that searches over the space of equivalence classes of BNs. The tabular CPT results are computed using GOBNILP[6] with standard pruning rules applied, and the GES results are reported using the R package pcalg (v2.7-4)[4] with the score as BIC. For tabular BIC, the parent set size was limited to 2, as scoring with larger parent set sizes led to the vast majority of the experiments failing with an out of memory error. We refer to this variant of tabular CPT as BIC2. We also experimented with the neural network based DAG-GNN, however as observed in [111], we noted that the performance was weaker than that of GOBNILP, so in our experiments, we have reported the GOBNILP results. In addition, we experimented with WINASOBS from [88], however within our experimental setup it was outperformed by GES, and thus we have not reported the results in detail.

We compare the performance of BN-FS-MARS using the metrics structural Hamming distance (SHD) and F_1 score. The experiments were conducted on computers with 2.2 GHz Intel E7-4850V3 CPUs with a memory limit of 32 GB, and each score-and-search experiment is limited to a total of 24 hours.

[1]`https://github.com/cmu-phil/Tetrad`
[2]`www.bnlearn.com/bnrepository`
[3]`http://gnw.sourceforge.net`
[4]`https://cran.r-project.org/web/packages/pcalg/`

Table 6.1: Bayesian networks from bnlearn, where n is the number of variables in the network and m is the number of edges in the network.

network	n	m
pathfinder	109	195
munin1	186	273
andes	223	338
diabetes	413	602
pigs	441	592
link	724	1125
munin2	1003	1244
munin4	1038	1306
munin3	1041	1388
munin	1041	1397

6.3.1 Implementation Details

MARS models for representing parent-child relationships were computed using the earth package (v5.3.1)[5] in R. This package builds regression models using the techniques proposed by [37]. The earth package limits every term in the model to have at most ten hinge functions in one product. However, for our experiments, we did not find this to be a limitation, as the models we learned did not exceed five hinge functions in a product.

To process discrete data for generating a MARS model, earth splits a categorical variable into l indicator columns of 1s and 0s, where l is the number of unique values the variable can take. For categorical response variables with l values, earth computes l models simultaneously. While the basis functions remain constant across these models, the coefficients can differ. The forward and backward phases are performed with the GCVs and RSSs summed across all l to minimize the sum of the GCV scores across all l models.

To determine variable importance rankings required by BN-FS-MARS, we use random forests. Random forests are an ensemble learning method that operate by constructing a set of decision trees at training time and outputting the class that is the average prediction of the individual trees for regression problems. Training random forests involves applying bagging to tree learners. Given a training set, bagging repeatedly selects a random sample of the training data and fits a decision tree to that sample. Random forests can rank

[5]https://cran.r-project.org/web/packages/earth

variables by importance in a regression problem as follows: 1) Fit a random forest to the data set and record the out-of-bag error for each observation (averaged over the trees in the forest). 2) The importance of the variable V_i is then determined by perturbing the values of variable V_i in the training data and computing the out-of-bag error again on the perturbed data set. The importance score for variable V_i is then determined by averaging the difference of the out-of-bag error before and after the perturbation over all trees in the random forest and normalizing it by the standard deviation of the differences. Random forests were computed using H2O (v3.32.0.5)[6] in Python.

To solve the optimization problem of finding the DAGs with the minimum total GCV score, we use GOBNILP, which is a state-of-the-art integer programming method for finding an optimal Bayesian network given a list of candidate parent sets and local scores for each variable V_i, with the objective of solving BNSL to find a DAG G that minimizes the total score, $\sigma(G)$. GOBNILP is an anytime algorithm, and it will return the best network at a given time.

6.3.2 Performance on Discrete Variables

To compare the performance of Algorithm 6.1 on discrete data, we generated synthetic networks of 100–900 nodes with Tetrad, with the maximum in-degree and out-degree of each node set as 3. We also used real world networks from bnlearn, with sizes between 100–1000+ nodes (see Table 6.1). From both these sets of networks, we randomly generated datasets with 1000 and 5000 samples, and scored candidate parent sets of all nodes with BN-FS-MARS using the top 5 features ranked by random forests in Step 3 of Algorithm 6.1. Finally, we computed the DAG structures using GOBNILP (Step 11) and compared the performance of BN-FS-MARS with BIC2 and GES, where BIC2 and GES both use the BIC score. Due to the anytime nature of the algorithm, we are able to obtain the best network found at any given time, and we use this property to report results on the network found within the time limit of 24 hours for score-and-search.

Performance on F_1 scores: Figure 6.2 reports the F_1 score of the synthetic Tetrad networks learned by each of the three methods in column (a). We observe that BN-FS-MARS (BFM) has the highest F_1 scores, followed by GES. Figure 6.3 reports the F_1 score of the bnlearn networks learned by each of the three methods in column (a). BN-FS-MARS usually has the best F_1 scores, especially as n increases, and the second best F_1 scores are from GES. For both Tetrad and bnlearn, we observe that increasing sample size from $N = 1000$ to $N = 5000$ improves F_1 scores of BN-FS-MARS and GES, and BN-FS-MARS

[6]https://docs.h2o.ai

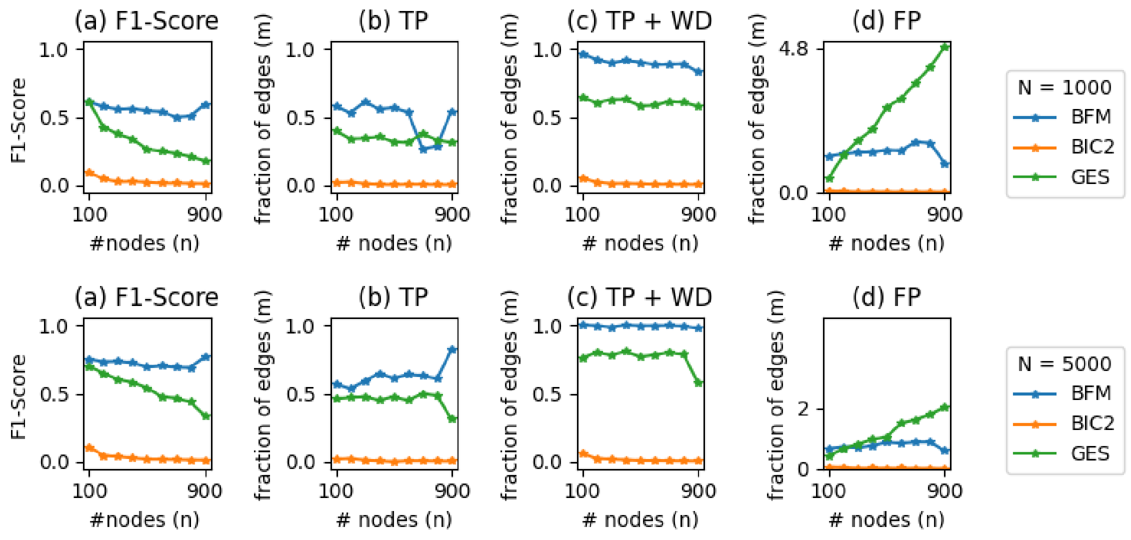


Figure 6.2: Comparison of structure learning approaches on various networks from Tetrad, where n is the number of variables and the number of edges in the ground truth network: our BN-FS-MARS (BFM), tabular CPT with maximum parent set size of two (BIC2), and GES (GES). For each ground truth network, we report the following for corresponding learned networks for sample sizes $N = 1000$ and $N = 5000$: (a) F_1 score, (b) TP (true positives), (c) sum of TP and WD (wrong direction), and (d) FP (false positives).

continues to have the highest scores. However, adding more data does not improve the F_1 scores for BIC2.

Performance on SHD: Figure 6.2 shows results for Tetrad on networks learned by each of the three methods in columns (b)-(d), and the numbers are normalized by the total number of edges m of the corresponding dataset. We see that BN-FS-MARS (BFM) almost always has the highest number of TPs as well as the highest sums of TP and WD. GES reports the second highest numbers and BIC2 reports the lowest numbers, which are close to 0. This is in line with what we observed in F_1 scores, showing that networks learned by BN-FS-MARS have a higher portion of edges present in the ground truth network, though some edges may not have the correct orientation. Note that FNs would correspond to the complement of the TP+WD plot, as they would show the number of edges that were not discovered at all.

Figure 6.2(d) shows numbers for FP. GES almost always has the highest number of FPs, indicating a tendency to learn extra edges not present in the ground truth. BN-FS-MARS has the second highest FP, but these are usually significantly lower than those for

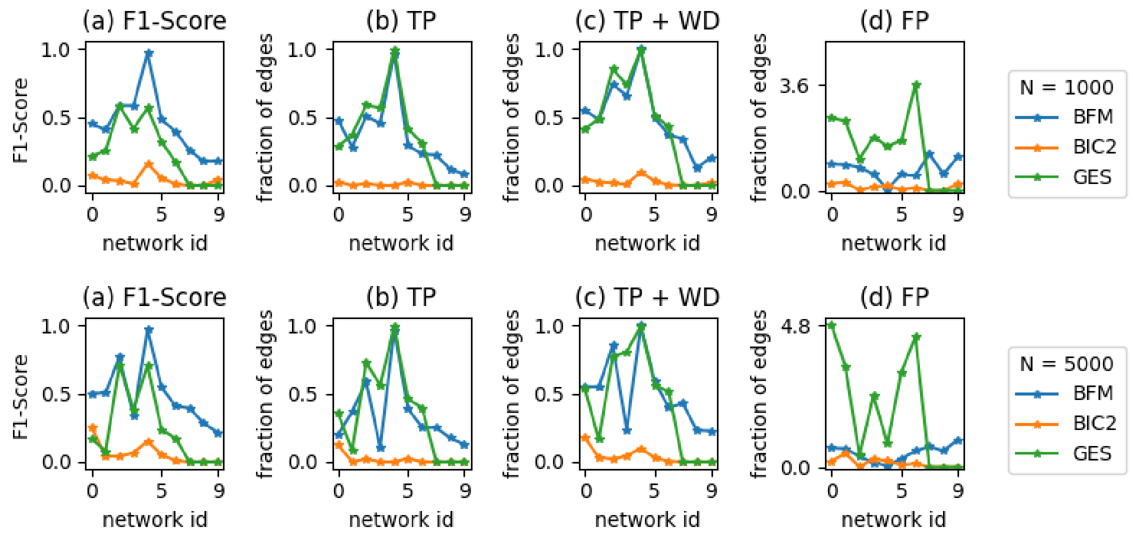


Figure 6.3: Comparison of structure learning approaches on various networks from bnlearn: our BN-FS-MARS (BFM), tabular CPT with maximum parent set size of two (BIC2), and GES (GES). For each ground truth network, we report the following for corresponding learned networks for sample sizes $N = 1000$ and $N = 5000$: (a) -score, (b) TP (true positives), (c) sum of TP and WD (wrong direction), and (d) FP (false positives).

GES. We observe that increasing sample size from $N = 1000$ to $N = 5000$ reduces the false positives significantly for both these methods. BIC2 has close to zero FPs. However, we can see that BIC2 had also failed to learn most of the edges in the network.

Figure 6.3 shows SHD results for bnlearn networks. It reports the SHD of the networks learned by each of the three methods in columns (b)-(d), and the numbers are normalized by the total number of edges m of the corresponding dataset. Here we see that GES almost always has the highest number of TPs as well as the highest sums of TP and WD. BN-FS-MARS reports the second highest numbers, which are close to the GES numbers, and BIC2 reports the lowest numbers, which are often close to 0.

For both Tetrad and bnlearn, we observe that increasing sample size from $N = 1000$ to $N = 5000$ usually improves TP and WD of networks learned using BN-FS-MARS and GES, and BN-FS-MARS is able to recover almost all of the edges in one of the instances (pigs). However, adding more data does not improve results for BIC2. For bnlearn benchmarks exceeding 1003 nodes—i.e., some of the munin sub-networks—GES and BIC2 did not converge at the end of the time limit. GES stopped with an out of time error, and BIC2 ran out of memory.

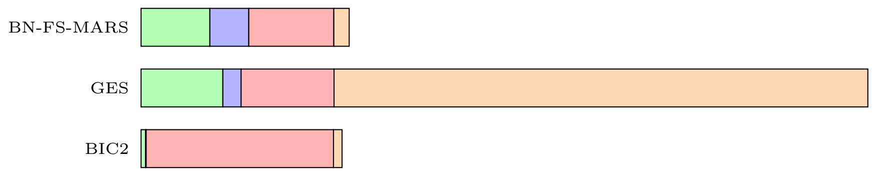


Figure 6.4: Results of BNSL on the benchmark **link** from bnlearn with N=5000 samples to show the proportion of TP=true positive, WD=wrong direction, FN=false negatives, FP=false positives in the learned DAG.

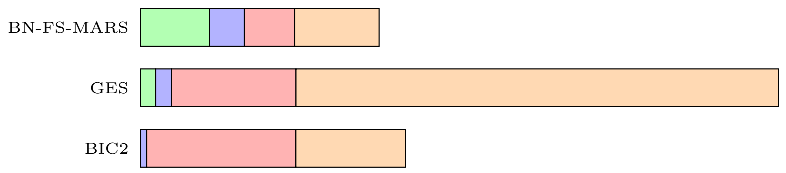


Figure 6.5: Results of BNSL on the benchmark **munin1** from bnlearn with N=5000 samples to show the proportion of TP=true positive, WD=wrong direction, FN=false negatives, FP=false positives in the learned DAG.

Figure 6.3(d) shows numbers for FP for bnlearn. As with Tetrad networks, GES almost always has the highest number of FPs, indicating a tendency to learn extra edges not present in the ground truth. BN-FS-MARS has the second highest FP, but these are usually significantly lower than those for GES. We observe that increasing sample size from $N = 1000$ to $N = 5000$ sometimes leads to an increase in FPs for GES, but almost always a reduction for BN-FS-MARS. BIC2 has the close to zero FPs. However, as with Tetrad, BIC2 also failed to recover most ground truth edges.

We show detailed results on two bnlearn benchmarks for SHD in Figure 6.4 and 6.5. For both benchamrks, we see that BIC2 has a very small number of ground truth edges, both for TP and WD. We also see that GES tends to learn a very high number of FPs. In both benchmarks, we see that BN-FS-MARS learns the largest number of TPs and WDs combined, and as a result the least number of FNs, which also leads to strong performance for F_1-scores as observed in Figure 6.3.

As noted in the discussion about performance metrics in Section 2.5.2, the SHD can have some significant limitations. As each of the summands for SHD are weighted equally, a low SHD does not necessarily indicate that the learned network is useful. As our results show, it is possible for two learned networks to have the same SHD, but one network could have learned few or none of the edges present in the ground truth, while the other

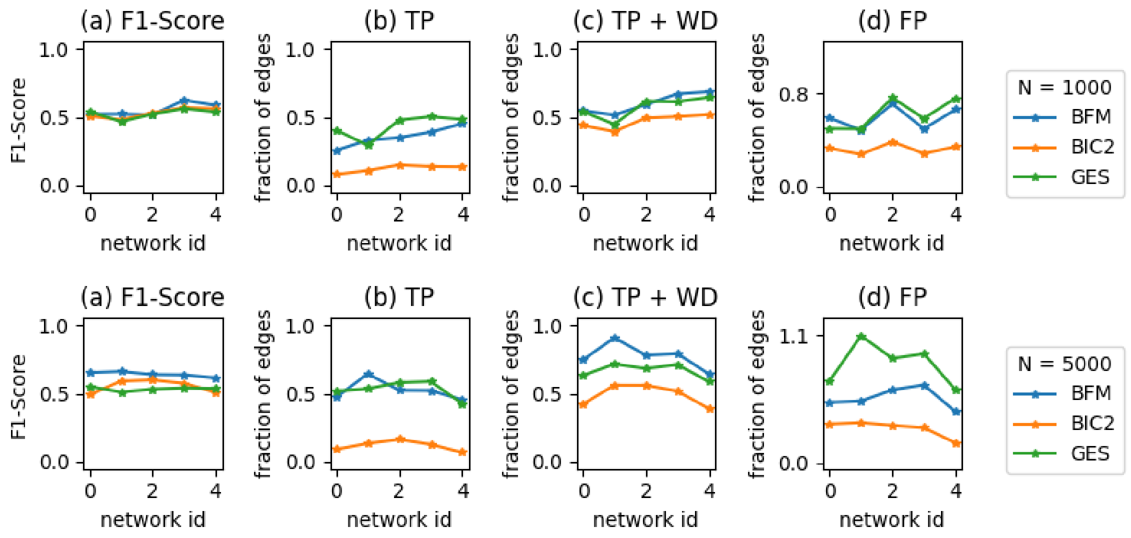


Figure 6.6: Comparison of structure learning approaches on various $n = 100$ networks from Dream4: our BN-FS-MARS (BFM), tabular CPT with maximum parent set size of two (BIC2), and GES (GES). For each ground truth network, we report the following for corresponding learned networks for sample sizes $N = 1000$ and $N = 5000$: (a) F_1 score, (b) TP (true positives), (c) sum of TP and WD (wrong direction), and (d) FP (false positives).

network could have learned all or nearly all of the ground truth edges. Depending on the application, the former network might not be a desirable outcome, even if it has a low SHD. This is the reason why we have chosen to represent our results by analyzing each of the summands for SHD, instead of simply showing the sum, i.e. the SHD itself.

We note that for one of the benchmarks (diabetes), BN-FS-MARS has decreased F_1 scores and TPs with $N = 5000$. A limitation of using the MARS method on discrete values is that it attempts to optimize the set of basis functions for all models for the child variable. If the child's domain is of large size, the results for the overall aggregated model will suffer in performance. Random forests are also known to perform poorly for such variables. The benchmark diabetes has several variables with domains of size 20+, leading to the observed degradation of performance of BN-FS-MARS. The more erratic nature of the plots for bnlearn is also because of the very different ranges of variable domain sizes across the benchmarks.

6.3.3 Performance on Continuous Variables

To test the performance of Algorithm 6.1 on continuous data, we used the five Dream4 networks, each of which have 100 nodes and between 176–249 edges. We generated datasets using simulated steady-state measurements with the genenetweaver software, with 1000 and 5000 samples. We scored candidate parent sets of all nodes with BN-FS-MARS with random forests as the feature selection method. For scoring, we used the top 5 scoring candidate parent sets in Step 3 of Algorithm 6.1. Once the scoring was complete, we computed the DAG structures using GOBNILP (Step 11). Due to the anytime nature of the algorithm, we are able to obtain the best network found at any given time, and we use this property to report results on the network found within the time limit of 24 hours for score-and-search. We compare the performance of BN-FS-MARS with tabular CPT (denoted BIC2 but computed using pyGOBNILP because of its ability to handle continuous variables) and GES, where both BIC2 and GES use the BIC score.

Performance on F_1 scores: Figure 6.6 reports the F_1 score of networks learned by each of the three methods in column (a). For sample size $N = 1000$, the methods have similar scores. Increasing sample size to $N = 5000$ improves F_1 scores of all methods slightly, and BN-FS-MARS shows the best scores.

Performance on SHD: Figure 6.6 shows SHD results for the networks learned by each of the three methods in columns (b)-(d), and the numbers are normalized by the total number of edges m of the corresponding dataset. GES almost always has the highest number of TPs, but it is close to BN-FS-MARS in sums of TP and WD. BIC2 reports the lowest numbers for both of these cases, but unlike with the discrete benchmarks, the numbers are not close to zero. For both Tetrad and bnlearn increasing sample size to $N = 5000$ improves TP and WD of networks learned by all methods, and BN-FS-MARS has the highest numbers for TP+WD.

The final column (d) shows numbers for FP. As with Tetrad and bnlearn networks, GES almost always has the highest number of FPs, indicating a tendency to learn extra edges not present in the ground truth. BN-FS-MARS has the second highest FP. We observe that increasing sample size to $N = 5000$ sometimes leads to an increase in FPs for GES, but always a reduction for BN-FS-MARS. BIC2 has the lowest number of FPs. A change in the behaviour of these three methods can be because of the smaller network size ($n = 100$). With a much smaller search space, BIC2 gives competitive results for F_1 scores, in part because of its ability to learn fewer FPs. However, the strength of BN-FS-MARS comes from its feature selection mechanism, which is harder to solve for continuous data [47].

6.3.4 Model Complexity

One of the strengths of BN-FS-MARS is its ability to identify large parent sets. Consider a variable in the benchmark pathfinder with a domain of size 4. It has five parents of domains sizes between 2–3. Tabular BIC would need almost 300 parameters to represent it in CPT form, and this would give it a large penalty, making it difficult to be able to detect it with score-and-search. BN-FS-MARS assigns it a model with 32 terms, with none of the basis functions having a product of more than two terms. The lowered penalty makes such a parent set more likely to be selected, especially if it has a good score. In our experiments we found that modeling parent sets with BN-FS-MARS enabled us to score very large parent sets easily.

6.4 Summary

In this chapter, we proposed a novel approach to score-and-search for learning a BN with MARS relations as possible representations for CPDs. This algorithm lets us model non-linear relationships between nodes with low complexity, enabling us to learn large parent set sizes, and it does not place any constraints on the global network structure. Our approach can effectively prune most candidate parent sets of a variable by leveraging variable importance results from feature learning algorithms. We show empirically that this algorithm can solve both discrete and continuous instances with a very large number of nodes.

Chapter 7

Conclusions

This thesis focused on improving the accuracy of the score-and-search approach for Bayesian network structure learning (BNSL) and in scaling the approach to datasets with larger numbers of random variables. Existing exact approaches to score-and-search for BNSL either severely restrict the structure of the Bayesian network or have only been shown to scale to networks with fewer than 30 random variables. In addition, most approaches learn only a single, optimal network, leading to poor accuracy in cases of low amounts of data. In Chapter 3, we provided a novel model averaging method for learning all networks within a factor of optimal in Bayesian network structure learning. Our method learns only *credible* models—those whose scores are optimal or close to optimal. We supplement our method with pruning rules, making it substantially more efficient and allowing scaling to far larger Bayesian networks without restricting their structure. Our empirical results also demonstrate that our credible set method can offer significant accuracy improvements over a bootstrapping model averaging method that is far and away the most widely used model averaging method in practice.

Previous work has shown that the accuracy of a data analysis can be improved by incorporating structured representations of the CPDs into the score-and-search approach for learning the DAG, although approaches to do this have not been shown to scale well and generally do not provide a principled way to incorporate these local structures within the score-and-search algorithm. In Chapter 4, we extend our novel score-and-search approach with model averaging to simultaneously learn the DAG and the local structure of the CPDs in the form of a noisy-OR representation, which is a linear model used to represent causality. Our approach is able to choose between CPTs and noisy-OR representations automatically. We provide an effective gradient descent algorithm to score a candidate noisy-OR using the widely used BIC score and we provide pruning rules that allow the search to successfully scale to medium sized networks while maintaining the optimality of the networks that are learned. Our experimental results provide evidence of the effectiveness of our approach. In particular, it was found that noisy-OR relations appeared in a significant proportion of the learned networks, for well known datasets.

To extend the scope of structured representation to non-linear models to allow representations of more complex relationships between variables, we extend our score-and-search approach with model averaging to simultaneously learn the DAG and the local structure of the CPD using neural networks representations in Chapter 5. The use of neural networks allows modeling of high-order interactions without needing an exponential number of parameters as in the CPT. Our approach compares favourably with approaches like decision trees, and performs particularly well in instances with low amounts of data.

While the neural network approach can easily represent high-order interactions between variables, the computationally intensive nature of neural networks limits scalability. In Chapter 6, we introduce a novel score-and-search approach to simultaneously learn a single DAG and model linear and non-linear local probabilistic relationships between variables. We achieve this by a combination of feature selection to reduce the search space for local relationships and extending the score-and-search approach to incorporate modeling the CPDs over variables as piecewise spline functions. We show on a set of discrete and continuous benchmark instances that our proposed approach can improve the accuracy of the learned graph while scaling to instances with over 1,000 variables. Our method is fast enough that it can be used in a bootstrapping model averaging approach in many instances.

With the combination of performance guarantees and the flexibility of our results, they can be applied to a large range of datasets to be modeled as BNs. We have seen BNs being used for analysis of data in various domains including finance [3], medicine [70, 74, 79] and the environment [36]. With the improvements in scalability and accuracy provided by the results in this thesis, such work can be extended in a principled manner to larger datasets with improved performance guarantees.

7.1 Future Work

For representing CPDs with neural networks, we can extend our work to explore pruning rules specific to the neural network CPDs, as well as analyzing the distribution of our inputs to investigate any underlying properties that can lead to provable guarantees about the learned distribution.

A natural extension to representing CPDs with MARS is the credible set model averaging approach. This can come from an adjustment in the backward stage of computing the MARS model. As explained in Section 6.1, the backward addresses overfitting by pruning terms added to the model in the forward stage. This pruning continues until we have a subset of terms leading to the best scoring model. We can add an approximation factor to this pruning step to retain near-optimal models, leading to a set of candidate parent sets that can be considered for the credible set approach discussed in Chapter 3. One challenge here would be ensuring optimality and near optimality, as basis functions are added to the model in a greedy fashion in the standard implementation.

Another alternative for extending our MARS approach to model averaging is the bootstrapping approach, where we would score using random samples with replacement from the dataset. Empirically, we observed that our score-and-search algorithm, BN-FS-MARS, converged in less than 20 minutes for the most of our datasets used in Section 6.3. Due to the highly scalable nature of our algorithm, we could extend this approach to learn a model averaged network from multiple bootstrap samples.

References

[1] Silvia Acid, Luis M. de Campos, and Javier G. Castellano. Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59(3):213–235, 2005.

[2] Hirotugu Akaike. Information theory and the maximum likelihood principle. In *Proceedings of the International Symposium on Information Theory*, pages 267–281, 1973.

[3] Ioannis Anagnostou, Javier Sanchez, Sumit Sourabh, and Drona Kandhai. Contagious defaults in a credit portfolio: A Bayesian network approach. *Journal of Credit Risk*, 16:1–26, 2020.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 2nd edition, 2011.

[5] Xue Bai, Rema Padman, Joseph Ramsey, and Peter Spirtes. Tabu search-enhanced graphical models for classification in high dimensions. *INFORMS Journal on Computing*, 20(3):423–437, 2008.

[6] Mark Bartlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Proc. of the 29th Conf. on Uncertainty in AI*, pages 182–191, 2013.

[7] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems*, pages 400–406, 1999.

[8] Remco Ronaldus Bouckaert. *Bayesian belief networks: From construction to inference*. PhD thesis, University of Utrecht, 1995.

[9] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Proc. of the 12th Conf. on Uncertainty in AI*, pages 115–123, 1996.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Leo Breiman, Jerome Friedman, R. A. Olshen, and Charles Stone. *Classification and Regression Trees*. Routledge, 1st edition, 1984.

[12] Wray Buntine. Theory refinement of Bayesian networks. In *Proc. of the Seventh Conf. on Uncertainty in AI*, pages 52–60, 1991.

[13] Eunice Yuh-Jie Chen, Arthur Choi, and Adnan Darwiche. Learning Bayesian networks with non-decomposable scores. In *Proc. 4th IJCAI Workshop on Graph Structures for Knowledge Representation and Reasoning (GKR 2015)*, pages 50–71, 2015. Available as: LNAI 9501.

[14] Eunice Yuh-Jie Chen, Arthur Choi, and Adnan Darwiche. Enumerating equivalence classes of Bayesian networks using EC graphs. In *Proc. 19th International Conference on Artificial Intelligence and Statistics*, pages 591–599, 2016.

[15] Eunice Yuh-Jie Chen, Adnan Darwiche, and Arthur Choi. On pruning with the MDL score. *International J. of Approximate Reasoning*, 92:363–375, 2018.

[16] Yetian Chen and Jin Tian. Finding the k-best equivalence classes of Bayesian network structures for model averaging. In *Proc. of the 28th Conf. on AI*, pages 2431–2438, 2014.

[17] David M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proc. of the 11th Conf. on Uncertainty in AI*, pages 87–98, 1995.

[18] David M. Chickering. Learning equivalence classes of Bayesian network structures. *J. Mach. Learn. Res.*, 2:445–498, 2002.

[19] David M. Chickering, David Heckerman, and Christopher Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proc. of the 13th Conf. on Uncertainty in AI*, pages 80–89, 1997.

[20] David M. Chickering, Christopher Meek, and David Heckerman. Large-sample learning of Bayesian networks is NP-hard. In *Proc. of the 19th Conf. on Uncertainty in AI*, pages 124–133, 2003.

[21] David M. Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *J. Mach. Learn. Res.*, 5:1287–1330, 2004.

[22] David Maxwell Chickering. Optimal structure identification with greedy search. *J. Mach. Learn. Res.*, 3:507–554, 2002.

[23] Gerda Claeskens and Nils Lid Hjort. *Model Selection and Model Averaging.* Cambridge University Press, 2008.

[24] Diego Colombo and Marloes H. Maathuis. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782, 2014.

[25] James Cussens. Bayesian network learning with cutting planes. In *Proc. of the 27th Conf. on Uncertainty in AI*, pages 153–160, 2011.

[26] James Cussens and Mark Bartlett. Gobnilp 1.6. 2 user/developer manual1. *University of York*, 2015.

[27] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks.* Cambridge University Press, 2009.

[28] Denver Dash and Gregory F. Cooper. Model averaging for prediction with discrete Bayesian networks. *J. Mach. Learn. Res.*, 5:1177–1203, 2004.

[29] Cassio P. de Campos and Qiang Ji. Efficient structure learning of Bayesian networks using constraints. *J. Mach. Learn. Res.*, 12:663–689, 2011.

[30] Cassio P. de Campos, Mauro Scanagatta, Giorgio Corani, and Marco Zaffalon. Entropy-based pruning for learning Bayesian networks using BIC. *Artificial Intelligence*, 260:42–50, 2018.

[31] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society Series B*, 39:1–38, 1977.

[32] Xiannian Fan and Changhe Yuan. An improved lower bound for Bayesian network structure learning. In *Proc. of the 29th Conf. on AI*, 2015.

[33] Xiannian Fan, Brandon Malone, and Changhe Yuan. Finding optimal Bayesian network structures with constraints learned from data. In *Proc. of the 30th Conf. on Uncertainty in AI*, pages 200–209, 2014.

[34] Xiannian Fan, Changhe Yuan, and Brandon Malone. Tightening bounds for Bayesian network structure learning. In *Proc. of the 28th Conf. on AI*, pages 2439–2445, 2014.

[35] Usama Fayyad and Keki Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[36] Wafa Feki-Sahnouna, Asma Hamzaa, Hasna Njahb, Mabrouka Barrajd, Nouha Mahfoudia, Ahmed Rebaie, and Malika Bel Hassend. A Bayesian network approach to determine environmental factors controlling Karenia selliformis occurrences and blooms in the Gulf of Gabès, Tunisia. *Harmful Algae*, 63:119–132, 2017.

[37] Jerome H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

[38] Nir Friedman and Moises Goldszmidt. Learning Bayesian networks with local structure. In *Proc. of the 12th Conf. on Uncertainty in AI*, pages 252–262, 1996.

[39] Nir Friedman and Moises Goldszmidt. Learning Bayesian networks with local structure. In *Learning in Graphical Models*, pages 421–459. Springer, 1998.

[40] Nir Friedman, Moises Goldszmidt, and Abraham Wyner. Data analysis with Bayesian networks: A bootstrap approach. In *Proc. of the 15th Conf. on Uncertainty in AI*, pages 196–205, 1999.

[41] Pilar Fuster-Parra, A. García-Mas, F. J. Ponseti, and F. M. Leo. Team performance and collective efficacy in the dynamic psychology of competitive team: A Bayesian network analysis. *Human Movement Science*, 40:98–118, 2015.

[42] José A. Gámez, Juan L. Mateo, and José M. Puerta. Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106–148, 2011.

[43] Dan Geiger and David Heckerman. Learning Gaussian networks. In *Proc. of the Tenth Conf. on Uncertainty in AI*, pages 235–243. Elsevier, 1994.

[44] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E Lübbecke, et al. The SCIP optimization suite 6.0. *Optimization Online*, 2018.

[45] Irving J. Good. A causal calculus. *The British Journal for the Philosophy of Science*, 12(45):43–51, 1961.

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[47] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[48] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[49] Alain Hauser and Peter Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *J. Mach. Learn. Res.*, 13:2409–2464, 2012.

[50] Ru He, Jin Tian, and Huaiqing Wu. Bayesian learning in Bayesian networks of moderate size by efficient sampling. *J. Mach. Learn. Res.*, 17:1–54, 2016.

[51] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20: 197–243, 1995.

[52] Max Henrion. Some practical issues in constructing belief networks. In *Proc. of the Third Conf. on Uncertainty in AI*, pages 132–139, 1987.

[53] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999.

[54] Estevam R. Hruschka Jr. and Nelson F. F. Ebecken. Towards efficient variables ordering for Bayesian networks classifier. *Data & Knowledge Engineering*, 63(2): 258–269, 2007.

[55] Steven Hwang, Linda Ng Boyle, and Ashis G. Banerjee. Identifying characteristics that impact motor carrier safety using Bayesian networks. *Accident Analysis and Prevention*, 128:40–45, 2019.

[56] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meilă. Learning Bayesian network structure using LP relaxations. In *Proc. International Conf. on Artificial Intelligence and Statistics*, pages 358–365, 2010.

[57] Sir Harold Jeffreys. *Theory of Probability: 3d Ed.* Clarendon Press, 1967.

[58] Robert E. Kass and Adrian E. Raftery. Bayes factors. *J. of the American Statistical Association*, 90(430):773–795, 1995.

[59] Mikko Koivisto. Parent assignment is hard for the MDL, AIC, and NML costs. In *J. of COLT*, pages 289–303, 2006.

[60] Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *J. Mach. Learn. Res.*, 5:549–573, 2004.

[61] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques.* The MIT Press, 2009.

[62] Kevin B. Korb and Ann E. Nicholson. *Bayesian Artificial Intelligence.* CRC press, 2010.

[63] Andrey Krutauz, Tapajit Dey, Peter C. Rigby, and Audris Mockus. Do code review measures explain the incidence of post-release defects? *Empirical Software Engineering*, pages 1–34, 2020.

[64] Wai Lam and Fahiem Bacchus. Using new data to refine a Bayesian network. In *Proc. of the Tenth Conf. on Uncertainty in AI*, pages 383–390, 1994.

[65] Pedro Larrañaga, Cindy M. H. Kuijpers, Roberto H. Murga, and Yosu Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics*, 26:487–493, 1996.

[66] Colin Lee and Peter van Beek. Metaheuristics for score-and-search Bayesian network structure learning. In *Canadian Conference on Artificial Intelligence*, pages 129–141. Springer, 2017.

[67] Zhenyu A Liao, Charupriya Sharma, James Cussens, and Peter van Beek. Finding all Bayesian network structures within a factor of optimal. In *Proc. of the 33rd Conf. on AI*, volume 33, pages 7892–7899, 2019.

[68] Charles X. Ling and Victor S. Sheng. Cost-sensitive learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining.* Springer, 2016.

[69] Zhifa Liu, Brandon Malone, and Changhe Yuan. Empirical evaluation of scoring functions for Bayesian network model selection. *BMC Bioinformatics*, 13(Suppl 15): S14, 2012.

[70] Yi Luo, Issam El Naqa, Daniel L. McShan, Dipankar Ray, Ines Lohse, Martha M. Matuszak, Dawn Owen, Shruti Jolly, Theodore S. Lawrence, Feng-Ming Kong, and

Randall K. Ten Haken. Unraveling biophysical interactions of radiation pneumonitis in non-small-cell lung cancer via Bayesian network analysis. *Radiotherapy and Onconology*, 123:85–92, 2017.

[71] David Madigan and Adrian E. Raftery. Model selection and accounting for uncertainty in graphical models using Occam's window. *J. of the Amercian Statistical Association*, 89:1535–1546, 1994.

[72] Brandon Malone and Changhe Yuan. A depth-first branch and bound algorithm for learning optimal Bayesian networks. In *Graph Structures for Knowledge Representation and Reasoning*, volume 8323 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 2014.

[73] Brandon Malone, Changhe Yuan, and Eric A. Hansen. Memory-efficient dynamic programming for learning optimal Bayesian networks. In *Proc. of the 25th Conf. on AI*, pages 1057–1062, 2011.

[74] Richard J. McNally, Patrick Mair, Beth L. Mugno, and Bradley C. Riemann. Comorbid obsessive-compulsive disorder and depression: A Bayesian network approach. *Psychological Medicine*, 47:1204–1214, 2017.

[75] Marina Meilă and Tommi Jaakkola. Tractable Bayesian learning of tree belief networks. In *Proc. of the 16th Conf. on Uncertainty in AI*, pages 380–388, 2000.

[76] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[77] Radford M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 567:71–113, 1992.

[78] Agnieszka Oniśko, Marek J. Druzdzel, and Hanna Wasyluk. Learning Bayesian network parameters from small data sets: Application of noisy-OR gates. *International Journal of Approximate Reasoning*, 27(2):165–182, 2001.

[79] Max M. Owens, Matthew D. Albaugh, Nicholas Allgaier, and (+ 70 others). Bayesian causal network modeling suggests adolescent cannabis use accelerates prefrontal cortical thinning. *Translational Psychiatry*, 12, 2022.

[80] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[81] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12: 2825–2830, 2011.

[82] Johan Pensar, Henrik Nyman, Jarno Lintusaari, and Jukka Corander. The role of local partial independence in learning of Bayesian networks. *International Journal of Approximate Reasoning*, 69:91–105, 2016.

[83] Franz Pernkopf and Jeff A. Bilmes. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. *J. Mach. Learn. Res.*, 11(Aug):2323–2360, 2010.

[84] Franz Pernkopf and Michael Wohlmayr. Stochastic margin-based structure learning of Bayesian network classifiers. *Pattern Recognition*, 46(2):464–471, 2013.

[85] Jonas Peters, Joris M. Mooij, Dominik Janzing, and Bernhard Schölkopf. Causal discovery with continuous additive noise models. *J. Mach. Learn. Res.*, 15:2009–2053, 2014.

[86] Brian D. Ripley. Pattern recognition and neural networks. *Cambridge University Press*, 1996.

[87] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[88] Mauro Scanagatta, Cassio P. de Campos, Giorgio Corani, and Marco Zaffalon. Learning Bayesian networks with thousands of variables. In *Proc. of Conference on Neural Information Processing Systems*, pages 1864–1872, 2015.

[89] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6: 461–464, 1978.

[90] Marco Scutari. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3):1–22, 2010.

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

[92] Yujia Shen, Arthur Choi, and Adnan Darwiche. A new perspective on learning context-specific independence. In *Proceedings of the Tenth International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 425–436, 2020.

[93] Shohei Shimizu, Patrik O Hoyer, Aapo Hyvärinen, Antti Kerminen, and Michael Jordan. A linear non-gaussian acyclic model for causal discovery. *J. Mach. Learn. Res.*, 7(10), 2006.

[94] Basilio Sierra, Nicolas Serrano, Pedro Larrañaga, Eliseo Plasencia, Iñaki Inza, Juan Jiménez, Pedro Revuelta, and Melfy Mora. Using Bayesian networks in the construction of a bi-level multi-classifier. a case study using intensive care unit patients data. *Artificial Intelligence in Medicine*, 22:233–48, 07 2001.

[95] Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proc. of the 22nd Conf. on Uncertainty in AI*, pages 445–452, 2006.

[96] Tomi Silander, Janne Leppä-aho, Elias Jääsaari, and Teemu Roos. Quotient normalized maximum likelihood criterion for learning Bayesian network structures. In *Proc. International Conference on Artificial Intelligence and Statistics*, 2018.

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

[98] Joe Suzuki and Jun Kawahara. Branch and bound for regular Bayesian network structure learning. In *Proc. of the 33rd Conf. on Uncertainty in AI*, 2017.

[99] Topi Talvitie, Ralf Eggeling, and Mikko Koivisto. Learning Bayesian networks with local structure, mixed variables, and exact algorithms. *International J. of Approximate Reasoning*, 2019.

[100] Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proc. of the 21st Conf. on Uncertainty in AI*, pages 548–549, 2005.

[101] Jin Tian. A branch-and-bound algorithm for MDL learning Bayesian networks. In *Proc. of the 16th Conf. on Uncertainty in AI*, pages 580–588, 2000.

[102] Jin Tian, Ru He, and Lavanya Ram. Bayesian model averaging using the k-best Bayesian network structures. In *Proc. of the 26th Conf. on Uncertainty in AI*, pages 589–597, 2010.

[103] Fulya Trösser, Simon de Givry, and George Katsirelos. Improved acyclicity reasoning for Bayesian network structure learning with constraint programming. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4250–4257, 2021.

[104] 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.

[105] Peter van Beek and Hella-Franziska Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Proc. of International Conf. on Constraint Programming*, pages 428–444, 2015.

[106] Jimmy Vandel, Brigitte Mangin, and Simon de Givry. New local move operators for Bayesian network structure learning. In *Proceedings of the 6th European Workshop on Probabilistic Graphical Models (PGM 2012)*, 2012.

[107] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proc. of the Sixth Conf. on Uncertainty in AI*, pages 220–227, 1990.

[108] Jiří Vomlel. Noisy-OR classifier. *International J. of Intelligent Systems*, pages 381–398, 2006.

[109] Ian. H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining.* Morgan Kaufmann, 3rd edition, 2011.

[110] Yang Xiang and Wanrong Sun. Learning NAT-modeled Bayesian network structures with Bayesian approach. In *Proceedings of the Thirty-fifth Canadian Conference on Artificial Intelligence*, 2022.

[111] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, 2019.

[112] Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. *J. of Artifical Intelligence Research*, 48:23–65, 2013.

[113] Adam Zagorecki and Marek J Druzdzel. Knowledge engineering for Bayesian networks: How common are noisy-MAX distributions in practice? *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(1):186–195, 2013.

[114] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Proc. of Conference on Neural Information Processing Systems*, 31, 2018.

[115] Yuan Zou, Johan Pensar, and Teemu Roos. Representing local structure in Bayesian networks by Boolean functions. *Pattern Recognition Letters*, 95:73–77, 2017.