DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

PhD THESIS

Learning Tractable Bayesian Networks

Author

Marco Benjumeda
MS in Artificial Intelligence

PhD supervisors

Concha Bielza
PhD in Computer Science

Pedro Larrañaga
PhD in Computer Science

2019

Thesis Committee

President: Serafín Moral

External Member: Johan Kwisthout

Member: Antonio Salmerón

Member: José Miguel Puerta

Secretary: Bojan Mihaljevic

Acknowledgements

Firstly, I would like to thank my supervisors, Concha Bielza and Pedro Larrañaga, for all of their guidance, motivation and wisdom throughout this thesis. Without their support, it would not have been possible to conduct this research.

My sincere thanks also go to Daniel Lowd, who provided me the opportunity to stay for three months at the University of Oregon. He was incredibly available and always provided great advice during my stay.

I am very grateful to my colleagues at the Computational Intelligence Group. It was a great pleasure to work with them and I really appreciate their help and good humour. Special thanks go to Sergio Luengo, who participated decisively in some of the main contributions of this thesis.

This work has been possible thanks to the financial support of the following projects and institutions: Cajal Blue Brain (C080020-09), TIN2013-41592-P and TIN2016-79684-P projects, S2013/ICE-2845-CASI-CAM-CM project, Fundación BBVA grants to Scientific Research Teams in Big Data 2016, European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 604102 (Human Brain Project) and European Union's Horizon 2020 research and innovation programme under grant agreement No. 720270 (HBP SGA1). During the whole PhD period I have been supported by a predoctoral contract for the formation of doctors from the Spanish Ministry of Science, Innovation and Universities (BES-2014-068637).

Personally, I thank my family and friends for their constant support at all times and for encouraging me in all of my pursuits in life. This work is dedicated to them.

Abstract

We are in the era of machine learning and the automatic discovery of knowledge from data is increasingly used to solve problems in our daily life. A key to successfully design useful intelligent algorithms is to be able to model the uncertainty that is present in the real world. Bayesian networks are a powerful tool that models uncertainty according to probability theory. Although the literature contains approaches that learn Bayesian networks from high dimensional datasets, traditional methods do not bound the inference complexity of the learned models, often producing models where exact inference is intractable.

This thesis focuses on learning tractable Bayesian networks from data, and contains the following five contributions. First, we propose strategies for learning Bayesian networks in the space of elimination orders (i). In this manner, we can efficiently bound the inference complexity of the networks during the learning process. This is specially useful in problems where data is incomplete. In these cases, the most common approach to learn Bayesian networks is to apply the structural expectation-maximization algorithm, which requires performing inference at each iteration of the learning process. Taking advantage of the reduced inference complexity of the models, we propose a new method with the purpose of guaranteeing the efficiency of the learning process while improving the performance of the original algorithm (ii).

Additionally, we study the complexity of multidimensional classification, a generalization of multilabel classification, in Bayesian networks. We provide upper bounds for the complexity of the most probable explanations and marginals of class variables conditioned to an instantiation of all predictor variables (iii). We use these bounds to propose efficient strategies for limiting the complexity of multidimensional Bayesian network classifiers during the learning process. With the objective of improving the performance of these models, we also propose methods for the discriminative learning of multidimensional Bayesian network classifiers (iv).

Finally, we address the problem of predicting seizure freedom in patients that have undergone temporal lobe epilepsy surgery (v). For that, we use a multidimensional Bayesian network classifier, which is specially well fitted to model the relationships among the class variables, i.e., seizure freedom at one, two and five years after surgery.

Resumen

Estamos en la era del aprendizaje automático y el descubrimiento automático de conocimientos a partir de datos se utiliza cada vez más para resolver problemas en nuestra vida diaria. Una clave para diseñar con éxito algoritmos inteligentes útiles es poder modelar la incertidumbre que está presente en el mundo real. Las redes bayesianas son una herramienta poderosa que modela la incertidumbre de acuerdo con la teoría de la probabilidad. Aunque la literatura contiene métodos que aprenden redes bayesianas a partir de conjuntos de datos con alta dimensionalidad, los métodos tradicionales no limitan la complejidad de inferencia de los modelos aprendidos, y a menudo producen modelos en los que la inferencia exacta es intratable.

Esta tesis se centra en el aprendizaje de redes bayesianas tratables a partir de datos y contiene la siguientes cinco contribuciones: Primero, proponemos estrategias para aprender redes bayesianas en el espacio de órdenes de eliminación (i). De esta manera, podemos acotar de manera eficiente la complejidad de inferencia de las redes durante el proceso de aprendizaje. Esto es especialmente útil en problemas donde los datos están incompletos. En estos casos, el enfoque más común para aprender redes bayesianas es aplicar el algoritmo de esperanza-maximización estructural, que requiere realizar inferencia en cada iteración del proceso de aprendizaje. Aprovechando la reducida complejidad de inferencia de los modelos, proponemos un nuevo método con el propósito de garantizar la eficiencia del proceso de aprendizaje y a la vez mejorar el rendimiento del algoritmo original (ii).

Además, estudiamos la complejidad de la clasificación multidimensional, una generalización de la clasificación multi-etiqueta, en redes bayesianas. Obtenemos cotas superiores para la complejidad del cómputo de las explicaciones más probables y de probabilidades marginales de las variables clase dado el valor de todas las variables predictoras (iii). Utilizamos estos límites para proponer estrategias eficientes para acotar la complejidad de los clasificadores multidimensionales basados en redes bayesianas durante el proceso de aprendizaje. Con el objetivo de mejorar el rendimiento de estos modelos, proponemos métodos para el aprendizaje discriminativo de clasificadores multidimensionales basados en redes bayesianas (iv).

Finalmente, abordamos el problema de predecir la ausencia de ataques en pacientes con epilepsia en el lóbulo temporal tras someterse a cirugía (v). Para ello, utilizamos un clasificador multidimensional basados en redes bayesianas, que están especialmente diseñados para modelar las relaciones entre las variables clase, es decir, ausencia de ataques epilépticos uno, dos y cinco años después de la cirugía.

Contents

List of Figures

List of Tables

Acronyms

ACC Accuracy

AIC Akaike information criterion

AUC Area under the receiver operating characteristic curve

BDe Bayesian Dirichlet for likelihood-equivalence

BIC Bayesian information criterion

BN Bayesian network

CB Class-bridge

CDN Conditional dependency network

CLL Conditional log-likelihood

CPD Conditional probability distribution

DAG Directed acyclic graph

E I Engel score I

E II-IV Engel score II, III, or IV

EEG Electroencephalography

EFD Equal frequency discretization

ELR Extended logistic regression

EM Expectation-maximization

EO Elimination order

ESS Expected sufficient statistics

ET Elimination tree

FFD Fixed frequency discretization

JT Junction tree

LEX Lexicographic breadth-first search algorithm

LL Log-likelihood

MAP Maximum a posteriori hypothesis

MB Markov blanket

MCS Maximum cardinality search

MDL Minimum description length

MEG Magnetoencephalography

MLE Maximum likelihood estimation

MNI Montreal Neurological Institute

MP Message passing

MPE Most probable explanation

MRI Magnetic resonance imaging

NP Nondeterministic polynomial time

ROC Receiver operating characteristic curve

SEM Structural expectation-maximization

SPECT Single-photon emission computed tomography

SVM Support vector machine

TJT Thin junction tree

TLE Temporal lobe epilepsy

TNR True negative rate

TPR True positive rate

TSEM Tractable structural expectation-maximization

UCSF University of California, San Francisco

VE Variable elimination

Y1 One year after surgery

Y2 Two years after surgery

Y5 Five years after surgery

Chapter 1

Introduction

Machine learning is the science of getting computers to learn by themselves by generalizing from datasets. During the last years, machine learning algorithms have been extensively used to aid humans in all sort of real-world problems, including medical diagnosis, neuroscience, speech recognition, computer vision and sports analytics. The models built from the data can be used to get predictions of the outcome of certain events and to reason about the phenomenas collected in the data. These models are also useful to hide the complexity of the real world, and therefore simplify the representation of the problems that we are facing.

As most of the application domains of machine learning involve uncertainty, robustly dealing with uncertainty is a must if we aim to produce models that are useful in practice. Bayesian networks (BNs) address this problem applying probability theory, that is long established, using probabilities to indicate different degrees of certainty. A BN represents a joint probability distribution as a product of conditional distributions. It is an intuitive graphical representation in the form of a directed acyclic graph (DAG), where the nodes represent the variables of the model and the arcs represent the probabilistic relationships between the variables. This explicit representation allows BNs to be consulted or modified by experts. Moreover, BNs are self-explanatory, i.e., they do not only give solutions to problems, but they also provide a justification.

One of the main limitations of BNs is that applying them in high dimensional domains is sometimes intractable. Learning BNs from data is an NP-complete problem [Chickering, 1996], but there are several heuristics that can be used to learn these models from large datasets [Spirtes et al., 2000; Gámez et al., 2011; Scanagatta et al., 2018a]. Although it is NP-hard to perform exact [Cooper, 1990] or approximate [Dagum and Luby, 1993] inference in general BNs, exact inference becomes tractable in BNs with bounded treewidth. Thus, bounding the treewidh of the models during the learning process would ensure the tractability of the output BNs.

Tractable inference also allows us to address other problems in which the complexity of inference is the bottleneck, such as learning models from incomplete datasets. During inference, BNs can deal with missing values via marginalization. The use of the structural expectation-maximization (SEM) algorithm [Friedman, 1997] is the most common approach

toward learning BNs from incomplete datasets. However, its main limitation is its demanding computational cost, caused mainly by the need to perform inference at each iteration of the algorithm. Therefore, ensuring the tractability of the models is essential for the efficiency of SEM.

Other problem that can directly benefit from learning BNs of bounded treewidth is multi-dimensional classification, which consists of obtaining the most probable explanation (MPE) of the class variables conditioned on the value of the predictor features. In this problem, identifying the relationships between the class variables is key to correctly classify all the class variables simultaneously. Multidimensional BN classifiers (MBCs) [van der Gaag and de Waal, 2006] are a generalization of the well-known BN classifiers. They are BNs with a restricted network topology, and their structures explicitly represent the relationships among the class and feature variables. Moreover, they can compute the posterior probability of the class variables given the features, which is specially useful when point estimates (for the classes) are not informative enough.

1.1 Hypothesis and objectives

For all the methods proposed in this thesis we assume that the input datasets are drawn from an existing joint probability distribution over a set of categorical variables. All the instances in the datasets are assumed to be independent and identically distributed. By default, we assume that the input datasets are fully observed. The algorithms that specifically address incomplete datasets assume that values are either missing completely at random or missing at random.

This work is motivated by the next hypothesis:

- The use of efficient algorithms for searching in the combined space of directed acyclic graphs and elimination orders will allow us to learn bounded treewidth Bayesian networks from high dimensional datasets.

The main objectives of the thesis are formulated according to the above hypothesis:

- Describe a framework to efficiently move in the space of elimination orders (EOs) during BN structure search.
- Introduce new methods in a BN structure learning algorithm to obtain bounded treewidth BNs.
- Propose a tractable method for learning BNs from incomplete datasets.
- Study the complexity of inference in MBCs and apply the results to learn tractable MBCs.
- Propose strategies for the discriminative learning of MBCs.
- Use MBCs to model surgery outcome at different time scales for patients with temporal lobe epilepsy (TLE).

1.2 Document organization

This thesis is organized as follows:

- **Chapter 1** gives an introduction to the problem of learning tractable BNs from data and shows the motivations and organization of this work.
- **Chapter 2** contains some definitions on the topic of BNs, and reviews the background of the topics addressed in this thesis. Namely, it includes an overview of inference complexity in BNs, structure learning algorithms, methods for learning BNs from incomplete datasets, multidimensional classification with BNs and discriminative learning of BNs.
- **Chapter 3** provides our proposal for learning tractable BNs from data by searching in the space of EOs.
- **Chapter 4** addresses the problem of efficiently learning BNs from incomplete datasets.
- **Chapter 5** analyzes the computational complexity of most probable explanations in MBCs and proposes a new algorithm for learning tractable MBCs.
- **Chapter 6** proposes strategies for the discriminative learning of MBCs.
- **Chapter 7** provides an MBC for predicting seizure freedom in patients with TLE one, two, and five years after surgery.
- **Chapter 8** gives our conclusions, suggests future research lines, and lists the published contributions derived from this thesis.

Chapter 2

Background

2.1 Bayesian networks

A BN $\mathcal{B}$ represents a joint probability distribution over a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. A BN is a pair $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$, where $\mathcal{G}$ is the structure and $\boldsymbol{\theta}$ is the vector of parameters $\mathcal{P}(X_i|\mathbf{Pa}_{X_i}^{\mathcal{G}})$ that represent the conditional probability distributions (CPDs) of each variable $X_i \in \mathcal{X}$ given its parents in the structure. Structure $\mathcal{G}$ encodes conditional independences among triplets of variables (local Markov property), i.e., each variable X_i is independent of its non-descendants given its parents $\mathbf{Pa}_{X_i}^{\mathcal{G}}$. Hence, the joint probability distribution can be factorized as

$$\mathcal{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathcal{P}(X_i|\mathbf{Pa}_{X_i}^{\mathcal{G}}).$$

Figure 2.1 shows the BN Earthquake. It models the behaviour of an alarm that can be activated by burglars or by an earthquake, and the reactions of John and Mary (if they call or not) to the activation of the alarm. All the variables in the BN are Boolean, and they represent the next events: there is a burglary (B), there is an earthquake (E), the alarm activates (A), John calls (J), and Mary calls (M).

The parameters of the network are represented in conditional probability tables. For example, the parameters of A show that the probability of activation of the alarm is higher if there is a burglary and there is not an earthquake ($\mathcal{P}(A = t|B = t, E = g) = 0.94$) than if there is an earthquake and there is not a burglary ($\mathcal{P}(A = t|B = f, E = t) = 0.29$), or that there is more probability that John calls ($\mathcal{P}(J = t|A = t) = 0.9$) than that Mary calls ($\mathcal{P}(M = t|A = t) = 0.7$) if the alarm is activated. The structure of a BN can be used to discover conditional independences among subsets of variables. Conditional independence is defined as:

Definition 2.1. ***(Conditional independence):*** *Let $\mathcal{X}_A$, $\mathcal{X}_B$ and $\mathcal{X}_C$ be three disjoint sets of nodes in a DAG $\mathcal{G}$. Given a probability distribution $\mathcal{P}$, $\mathcal{X}_A$ and $\mathcal{X}_B$ are conditionally independent given $\mathcal{X}_C$ if and only if:*

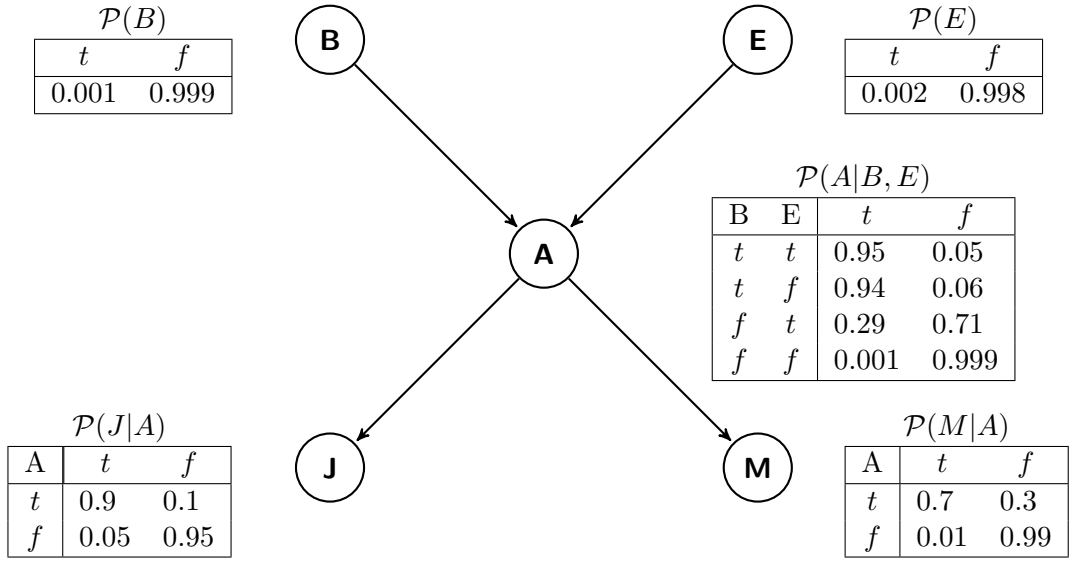


Figure 2.1: BN Earthquake.

$$\mathcal{P}(\mathcal{X}_A, \mathcal{X}_B|\mathcal{X}_C) = \mathcal{P}(\mathcal{X}_A|\mathcal{X}_C) \cdot \mathcal{P}(\mathcal{X}_B|\mathcal{X}_C).$$

We use $(\mathcal{X}_A \perp \mathcal{X}_B|\mathcal{X}_C)_{\mathcal{P}}$ *to denote that in* $\mathcal{P}$, $\mathcal{X}_A$ *and* $\mathcal{X}_B$ *are conditionally independent given* $\mathcal{X}_C$.

A way of inducing the independences encoded in a BN is to analyze its graph structure using the concept of d-separation [Geiger et al., 1990; Pearl, 1995, 1988].

Definition 2.2. ***(d-separation):*** *Let* $\mathcal{X}_A$, $\mathcal{X}_B$ *and* $\mathcal{X}_C$ *be three disjoint sets of nodes in a DAG* $\mathcal{G}$. *Let* $\mathcal{T}$ *be the set of possibles trials from any node* $X_a \in \mathcal{X}_A$ *to any node* $X_b \in \mathcal{X}_B$, *where a trial in the network is a succession of edges in* $\mathcal{G}$, *no matter their directions. Then* $\mathcal{X}_C$ *blocks a trial* $T_I \in \mathcal{T}$ *if one of the following holds:*

1. T_I *contains a chain* $T_{i-1} \rightarrow T_i \rightarrow T_{i+1}$ *such that* $T_i \in \mathcal{X}_C$.
2. T_I *contains a fork* $T_{i-1} \leftarrow T_i \rightarrow T_{i+1}$ *such that* $T_i \in \mathcal{X}_C$.
3. T_I *contains a collider* $T_{i-1} \rightarrow T_i \leftarrow T_{i+1}$ *such that* T_i *and any of its descendants do not belong to* $\mathcal{X}_C$.

If all the trials in $\mathcal{T}$ *are blocked by* $\mathcal{X}_C$, *then* $\mathcal{X}_C$ *d-separates* $\mathcal{X}_A$ *and* $\mathcal{X}_B$, *which is expressed by* $(\mathcal{X}_A \perp \mathcal{X}_B|\mathcal{X}_C)_{\mathcal{G}}$.

The relation between the topological properties of directed graphs and the conditional independences of their underlying probability distribution is defined by the concept of *I-map* [Pearl, 1988].

Definition 2.3. ***(I-map):*** *Let* $\mathcal{X}_A$, $\mathcal{X}_B$ *and* $\mathcal{X}_C$ *be three disjoint sets of nodes in a DAG* $\mathcal{G}$. *Then* $\mathcal{G}$ *is an I-map of a probability distribution* $\mathcal{P}$ *if:*

$$(\mathcal{X}_A \perp \mathcal{X}_B | \mathcal{X}_C)_{\mathcal{G}} \implies (\mathcal{X}_A \perp \mathcal{X}_B | \mathcal{X}_C)_{\mathcal{P}}$$

This means that if $\mathcal{G}$ is an I-Map of $\mathcal{P}$, if two set of nodes $\mathcal{X}_A$ and $\mathcal{X}_B$ are d-separated by another set of nodes $\mathcal{X}_C$ in $\mathcal{G}$, then $\mathcal{X}_A$ and $\mathcal{X}_B$ are conditionally independent in $\mathcal{P}$ given $\mathcal{X}_C$. Also, any variable in $\mathcal{P}$ is conditionally independent of the rest of the variables given its Markov blanket [Pearl, 1988].

Definition 2.4. ***(Markov blanket):*** *Let $\mathcal{B}$ be a BN over $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$. The Markov blanket of any variable $X_i \in \mathcal{X}$ in $\mathcal{B}$ ($MB(X_i)$) is the set of variables composed by the parents of X_i, its children, and the parents of its children.*

BNs can be formally defined as follows:

Definition 2.5. ***(Bayesian network):*** *Let $\mathcal{P}$ be a probability distribution over a set of variables $\mathcal{X}$, then a Bayesian network $\mathcal{B}$ is composed of a DAG $\mathcal{G}$ and a set of conditional probability distributions $\boldsymbol{\theta}$ such that:*

- *Every node X_i in $\mathcal{G}$ represents a variable in $\mathcal{X}$, and has a conditional probability distribution $\boldsymbol{\theta}_{X_i} = \mathcal{P}(X_i | \boldsymbol{Pa}^{\mathcal{G}}_{X_i})$ associated to it.*
- *$\mathcal{G}$ is a minimal I-map of $\mathcal{P}$. That is, no arcs can be removed from $\mathcal{G}$ without negating the I-map property.*

2.2 Inference in Bayesian networks

Probabilistic inference can be used to refer to multiple problems in BNs. Some well-known inference problems are: evidence propagation, finding the maximum a posteriori (MAP) hypothesis, and computing the MPE. Evidence propagation entails finding the posterior probability $\mathcal{P}(\mathcal{X}_q | \boldsymbol{e})$ of a set of query variables $\mathcal{X}_q$ conditioned on evidence $\boldsymbol{e}$. It can be used for some key tasks such as prediction and diagnosis. Finding the MAP consists of searching the most probable configuration of a set of variables in a BN for a given evidence. The MPE is a special case of MAP that involves searching the most probable configuration of all variables not instantiated in a BN for a given evidence. Kwisthout [2011] provides a thorough overview of the complexity of many MPE and MAP variants.

In this thesis, we use inference complexity to refer to the complexity of evidence propagation in BNs. Exact inference in BNs is generally NP-hard [Cooper, 1990], and approximate inference is commonly used when exact inference is intractable. Approximate inference in BNs is also NP-hard [Dagum and Luby, 1993], and, although it has been useful for solving some otherwise intractable problems, it has some major drawbacks. It degrades the responses output by the model, and hardly any of these algorithms offer any guarantees of convergence.

The message-passing (MP) algorithm [Pearl, 1982; Kim and Pearl, 1983] can perform exact linear time inference in the number of variables of any BN $\mathcal{B}$ when its topology is a polytree [1]. However, there are many situations where polytrees are not representative enough,

[1] A polytree is a directed acyclic graph whose underlying undirected graph is a tree.

and this restriction is therefore too strict in practice. Inference in BNs with loops is far from straightforward, and we cannot use MP to perform exact inference in this type of networks. Although MP has been adapted to deal with loops, the resultant method, called loopy belief propagation [Pearl, 1988], provides only approximate results. Most exact inference methods for graphs with loops are based on variable elimination (VE) [Shachter, 1990; Zhang and Poole, 1994], recursive conditioning [Pearl, 1985; Darwiche, 2001], and junction tree (JT) belief propagation [Shenoy and Shafer, 1990; Jensen et al., 1990].

2.2.1 Elimination orders, treewidth and inference complexity

VE is one of the most straightforward methods for inference in BNs. It successively eliminates the variables of a network until it yields the answer to a specified query. This algorithm is typically defined in terms of factors. A factor is a function that maps value assignments of a set of random variables to real positive numbers; CPDs are an example of factors. The elimination of a variable X_i includes outputting the product of all the factors containing X_i and marginalising out X_i, producing a new factor ϕ_i. The order in which the variables are removed is called the elimination order. An EO of a set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ is a permutation $\pi = (\pi(X_1), \ldots, \pi(X_n))$ of $\mathcal{X}$. We use $(X_i < X_j)_\pi$ to denote that X_i must be eliminated before X_j given π. The sequence of factors induced by an EO π in graph $\mathcal{G}$ are the factors obtained after eliminating each node X_i in $\mathcal{G}$ acording to the EO π.

Inference complexity is influenced by the selection of the EO [Dechter, 1999]. To provide a formal definition of optimal EO, we must first introduce the concept of cluster [Darwiche, 2009]:

Definition 2.6. ***(Cluster)*** *Let $\phi_1, \ldots, \phi_n$ be the sequence of factors induced by an EO π in graph $\mathcal{G}$. Cluster C_i is defined as the set of random variables in the domain of factor ϕ_i.*

The optimality of an EO depends on its width.

Definition 2.7. ***(Optimal EO)*** *Let $\mathbf{C} = (C_1, \ldots, C_n)$ be the sequence of clusters induced by an EO π in graph $\mathcal{G}$. The width of π in $\mathcal{G}$, which we denote as width$(\mathcal{G}, \pi)$, is the size (number of variables) of its largest cluster in $\mathbf{C}$ minus one. We refer to the EO with the minimal width for $\mathcal{G}$ as the optimal EO.*

Note that our use of optimal EO is analogous to optimal graph triangulations [Darwiche, 2009], and we do not consider the best EO for specific inference queries. Inference complexity in BNs is typically evaluated in terms of their treewidth.

Definition 2.8. ***(Treewidth)*** *The treewidth of a graph $\mathcal{G}$ is the width of the optimal EO for $\mathcal{G}$.*

The notion of treewidth was introduced by Robertson and Seymour [1986]. Intuitively, the treewidth of a BN $\mathcal{B}$ can be understood as a measure of similarity between $\mathcal{B}$ and a tree (e.g., a tree has treewidth one). The computational cost of VE, recursive conditioning, and

JTs is exponential in the treewidth of $\mathcal{B}$. Thus, bounding the treewidth of $\mathcal{B}$ entails limiting its inference complexity [Kwisthout et al., 2010].

The literature also includes approaches that are not always exponential in the treewidth. In this case, tractable exact inference does not necessarily call for models with a low treewidth. These methods exploit the local network structures [Boutilier et al., 1996; Darwiche, 2003], or the exchangeability between the model variables [Niepert and van den Broeck, 2014]. Nevertheless, it is extremely challenging to consider the above properties during the learning process. Therefore, in these thesis we focus on learning bounded treewidth BNs.

The combined space of EOs and DAGs is redundant. This means that there may be multiple EOs that induce the same factors (using VE) for the same BN. We define the equivalence of two EOs as:

Definition 2.9. ***(Equivalence of EOs)*** *Let $\mathcal{B}$ be a BN over $\mathcal{X}$, and π_1 and π_2 two EOs of $\mathcal{X}$. Let $\boldsymbol{Cls}_{\pi_1}(X_i)$ and $\boldsymbol{Cls}_{\pi_2}(X_i)$ be the clusters induced by visiting node X_i during VE using the EOs π_1 and π_2, respectively. π_1 and π_2 are equivalent for $\mathcal{B}$ if, for each $X_i \in \mathcal{X}$, $\boldsymbol{Cls}_{\pi_1}(X_i) = \boldsymbol{Cls}_{\pi_2}(X_i)$.*

The completeness of a set of EOs $\mathcal{S}$ for $\mathcal{B}$ ensures that if an EO π_i belongs to $\mathcal{S}$ all the EOs that are equivalent to π_i for $\mathcal{B}$ also belong to $\mathcal{S}$. Note that the completeness of $\mathcal{S}$ does not imply that all the EOs in $\mathcal{S}$ are equivalent for $\mathcal{B}$.

Definition 2.10. ***(Completeness of a set of EOs)*** *A set of EOs $\mathcal{S}$ is complete for $\mathcal{B}$ if there are no two equivalent EOs π_i, π_j, with $\pi_i \in \mathcal{S}$ and $\pi_j \notin \mathcal{S}$, for $\mathcal{B}$.*

For example, assume a BN $\mathcal{B}$ over variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ that represents the product of marginals $\mathcal{P}(X_1, \ldots, X_n) = \mathcal{P}(X_1)\mathcal{P}(X_2) \cdots \mathcal{P}(X_n)$. Given $\mathcal{B}$, VE induces the same factors for any EO of $X_1, \ldots, X_n$. Hence, all the $n!$ possible EOs are equivalent for $\mathcal{B}$, and there is a single complete set of EOs that contains all the permutations of $\mathcal{X}$.

2.2.2 Treewidth estimation

It is NP-hard to exactly compute the treewidth of a BN [Arnborg et al., 1987]. There are many approaches whose time complexity is exponential in the number of network variables [Shoikhet and Geiger, 1997; Fomin et al., 2004; Bodlaender et al., 2006; Fomin and Villanger, 2012]. In practice, heuristics are most often used. As the treewidth of a graph $\mathcal{G}$ is given by the width of its optimal EO, some well-known heuristics estimate the treewidth of $\mathcal{G}$ by searching for good EOs for $\mathcal{G}$. The list below includes some popular approaches:

- Greedy search methods: Two widely used strategies are to eliminate, at each iteration, the smallest degree node (i.e., the node with fewest neighbors) in the graph [Markowitz, 1957] or the node that produces the minimum number of fill-in (min-fill) edges [Kjærulff, 1990]. In practice, the min-fill algorithm performance is generally slightly better, but its computational cost is higher.

- Graph recognition techniques: The lexicographic breadth-first search algorithm (LEX) [Rose et al., 1976] and the maximum cardinality search (MCS) algorithm [Tarjan and Yannakakis, 1984] return an optimal EO only if the input graph is chordal[2]. The chordality assumption is very restrictive in practice, but there are two variants of these methods, respectively called LEX-M [Rose et al., 1976] and MCS-M [Berry et al., 2004], that also search for a good EO if the graph is not chordal.

- Local search and evolutionary techniques: Some well-known heuristics like simulated annealing [Kjærulff, 1992], genetic algorithms [Larrañaga et al., 1997], or tabu search [Clautiaux et al., 2004] have been used to find good EOs.

- Another approach focuses on finding the best graph separators, recursively splitting the clusters of an initial tree decomposition into smaller components [Koster, 1999]. Most methods using this strategy give theoretical guarantees of the treewidth upper bound.

Bodlaender and Koster [2010] provide an overview of the different heuristics used for computing upper bounds for graph treewidth, including the above methods. Their experiments suggest that greedy search methods usually provide the best trade-off between performance and efficiency.

Sometimes it is sufficient to check if the treewidth of a model is less than or equal to k rather than exactly computing the treewidth of $\mathcal{G}$; for instance in the problem of learning models with bounded treewidth (see Section 2.3.1). Although this is an NP-complete problem [Arnborg et al., 1987], checking if $\text{tw}(\mathcal{G}) \leq k$ requires linear time in the number of variables for a fixed k. Nevertheless, the time complexity for solving the above inequality is super-exponential in the treewidth of $\mathcal{G}$ [Bodlaender, 1993], which means that it may be intractable unless k is very small.

In these thesis we focus on learning BNs with bounded treewidth. To bound the treewidth of all the BN candidates during the learning process we propose an efficient heuristic to incrementally search in the space of EOs.

2.3 Learning the structure of Bayesian networks

Learning the BN structure is typically performed by a scoring metric that evaluates each candidate network with the data (score+search). Most scoring functions can be classified as Bayesian or information theory metrics. The former compute the posterior probability distribution of the model $P(\mathcal{B}|\mathcal{D})$ given a prior distribution over the possible networks conditioned on the data $\mathcal{D}$. Some popular examples are Bayesian Dirichlet for likelihood-equivalence (BDe) [Heckerman et al., 1995] and K2 [Cooper and Herskovits, 1992] scoring functions. Information theory metrics measure the compression that can be achieved by the BN over the data. Optimizing data compression is equivalent to maximizing log-likelihood, which is

[2] A graph is chordal if all cycles of four or more nodes have an edge that connects two nodes of the cycle but is not part of the cycle.

defined as:

$$\ell(\mathcal{B}|\mathcal{D}) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} M_{ijk}\log\left(\frac{M_{ijk}}{M_{ij}}\right),$$

where q_i and r_i are number of possible configurations of $\mathbf{Pa}_{X_i}^{\mathcal{G}}$ ($\mathcal{G}$ is the structure of $\mathcal{B}$) and X_i, respectively, M_{ijk} is the number of cases of $\mathcal{D}$ where $\mathbf{Pa}_{X_i}^{\mathcal{G}}$ and X_i are in their j-th and k-th configuration, respectively, and M_{ij} is $\sum_{k=1}^{r_i} M_{ijk}$.

As optimizing the log-likelihood during the structure search leads to fully connected DAG structures, it is necessary to set constrains to avoid overfitting. A common approach is to penalize the representation complexity of BNs, which is given by the number of parameters. These metrics are usually of the following form:

$$\text{score}(\mathcal{D},\mathcal{B}) = \ell(B|\mathcal{D}) - \text{pen}(\mathcal{B},\mathcal{D}), \tag{2.1}$$

where $\ell(\mathcal{B}|\mathcal{D})$ is the log-likelihood of $\mathcal{B}$ given $\mathcal{D}$ and $\text{pen}(\mathcal{B},\mathcal{D})$ is a penalty function that can depend on $\mathcal{B}$ and $\mathcal{D}$.

In the Akaike information criterion (AIC) [Akaike, 1974] $\text{pen}(\mathcal{B},\mathcal{D}) = |\mathcal{B}|$, where $|\mathcal{B}|$ is the number of parameters of $\mathcal{B}$. The Bayesian information criterion (BIC), that is based on the Schwarz Information Criterion [Schwarz, 1978], also penalizes the length of $\mathcal{D}$:

$$\text{BIC}(\mathcal{B},\mathcal{D}) = \ell(\mathcal{B}|\mathcal{D}) - \frac{1}{2}\log(M)|\mathcal{B}|,$$

where M is the number of cases in $\mathcal{D}$. Note that BIC is equivalent to the minimum description length (MDL) [Bouckaert, 1993; Lam and Bacchus, 1994] as a BN scoring function. Carvalho [2009] provides a thorough overview of both Bayesian and information theory metrics.

When data is complete, the decomposability property of the above scoring functions allows for efficient learning algorithms based on local search methods [Cooper and Herskovits, 1992; Wang et al., 2004; Gámez et al., 2011; Scanagatta et al., 2018a].

Other well-known approach is to consider the learning process as a constrain satisfaction problem, trying to get the conditional independences between the variables by using a statistical hypothesis test, and then selecting the model that fits better the dependences and independences obtained in the tests [Spirtes et al., 2000; Mahdi and Mezey, 2013]. These techniques do not use an explicit score metric to test the likelihood between the network and the data, and instead they use statistical tests to get the skeleton of the network (edges without orientation) and then they orient the edges by recovering the v-structures of the network. Hybrid methods combine both approaches [Tsamardinos et al., 2006]. Usually, they use hypothesis tests to build the skeleton of the network and local search techniques to find the direction of the arcs.

2.3.1 Previous work on learning low inference complexity Bayesian networks

Most approaches that address the problem of inference complexity during the learning process use a bound k on the model treewidth (i.e., bounded treewidth models). They reject any candidate $\mathcal{G}$ for which $\text{tw}(\mathcal{G}) > k$, where $\text{tw}(\mathcal{G})$ is the treewidth of $\mathcal{G}$. Learning bounded treewidth BNs is an NP-hard problem [Korhonen and Parviainen, 2013]. The literature contains exact methods for this problem that reduce the problem to either a weighted maximum satisfiability problem [Berg et al., 2014] or mixed-integer linear programming formulations [Nie et al., 2014; Parviainen et al., 2014]. These methods scale poorly with respect to the number of model variables and model treewidth.

Elidan and Gould [2009] proposed a method that uses an incremental triangulation of BNs during the structure search to output bounded treewidth models. Their method is treewidth-friendly (i.e., each update of the triangulation does not increment its width by more than one), and it basically applies the best chain of arc additions in each iteration given a topological ordering of the variables. Its main limitation is that the method is restricted to a single topological ordering of the variables in each iteration.

Nie et al. [2014] proposed an efficient approach that samples k-trees randomly and selects the best BN structure whose moral graph is the sampled k-tree. As the convergence of the sampling process can be a problem when the number of variables is not small, Nie et al. [2017] also provided a strategy for moving in the space of k-trees and proposed a score (I-score) to measure how well a k-tree fits the data. The authors showed that this measure is correlated with the BDeu score of the learned networks.

Scanagatta et al. [2016] proposed a method (called k-greedy) for learning bounded treewidth BNs from very large datasets. Before performing the structure search, k-greedy initializes a cache of candidate parent sets for each node using the approach of Scanagatta et al. [2015]. Then, it samples the space of orderings of variables, performing the next steps for each order. First, an initial structure with the first $k+1$ variables in the order is learned. Depending on the value of k, k-greedy uses either an exact [Cussens, 2011] or an approximate [Scanagatta et al., 2015] structure learning method. Second, the structure incrementally grows according to the chosen order, ensuring that at each step the moral graph of the structure is a partial k-tree. This process is repeated until the maximum allowed execution time is met. More recently, Scanagatta et al. [2018b] improved k-greedy by proposing a heuristic score for choosing the order in which the variables are visited. This heuristic ranks the variables by comparing the highest-scoring parent set with the lowest scoring parent set that do not exceed the treewidth bound. The resultant method is called k-MAX. As the former, k-MAX requires predefining a maximum execution time to explore the space of solutions. Extensive experiments showed that both approaches consistently outperform some of the above methods [Parviainen et al., 2014; Nie et al., 2014, 2017] for learning bounded treewidth BNs. A limitation of k-greedy and k-MAX is that they only learn BNs whose reverse topological order, when used as an EO, has at most width k.

There are also several approaches that learn JTs with bounded treewidth, usually called

thin junction trees (TJTs) [Bach and Jordan, 2001]. This problem is NP-complete when the bound on the treewidth k is greater than one [Karger and Srebro, 2001]. Chechetka and Guestrin [2008] proposed a method that learns TJTs with probably approximately correct guarantees in time $O(n^k)$, which is intractable when k is not very small. Shahaf and Guestrin [2009] used the graph cuts algorithm [Ahuja et al., 1993] to pick the best separator in each iteration during the learning process, requiring polynomial time in both n and k. As mentioned above, heuristics that use separators usually perform worse in practice than heuristics that search for good EOs for estimating the treewidth of the models.

Some approaches use a penalization in the inference complexity instead of a hard constraint. Lowd and Domingos [2008] proposed the first method (LearnAC) to learn arithmetic circuits (ACs)[3] directly from data. This method penalizes the size of each circuit exploiting the local structures of the models to learn networks that can be tractable even for high treewidths. Moves in the space of ACs can be extremely computationally demanding, as circuit structure can be huge. LearnAC uses a greedy approach to address these difficulties, where the best split (i.e., conditioning the CPD of a variable to an instance of another variable) is applied at each iteration. Like EOs, the order of splits can have a major effect on network size, and this type of search process is not capable of reconfiguring the split ordering during the learning process.

2.3.2 Learning Bayesian networks from partially observed data

Evaluating a structure according to any of the scores described in Section 2.3 involves estimating the optimal parameters for each network candidate, which can be achieved efficiently when the data is complete. However, in the presence of missing values or hidden variables, it is not feasible to efficiently estimate the parameters because the network score does not decompose.

The most popular optimization method for estimating the parameters from partially observed data is the expectation-maximization (EM) algorithm [Dempster et al., 1977; McLachlan and Krishnan, 2008; Liao and Ji, 2009]. EM addresses the missing data problem by selecting a starting point, which is either an initial set of parameters or an initial assignment to the missing variables. Once we have a parameter set, we can apply inference to complete the data, or conversely, once we have the complete data, we can estimate the set of parameters using maximum likelihood estimation (MLE). EM iterates between both steps until convergence.

Assume we have a BN $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ over a set of variables $\mathcal{X}$ and a dataset $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_M\}$. Let $\mathbf{O}[m]$ be the variables whose values are observed (not missing) at the m-th instance of $\mathcal{D}$. EM uses the set of parameters $\boldsymbol{\theta}$ to complete the data probabilistically (E-step), resulting in the completed dataset $\mathcal{D}^+$. Obtaining $\mathcal{D}^+$ can be unfeasible in the majority of cases because its cost is exponential in the number of missing values. Nevertheless, the expected sufficient

[3] ACs are DAGs in which the inner nodes are addition and multiplication nodes and the leaves are numeric variables or constants. They have been adapted to perform inference in BNs [Darwiche, 2003].

statistics (ESS) required for estimating a new set of parameters are obtained as follows:

$$\mathrm{ESS}_{\boldsymbol{\theta}}[X = x, \mathbf{Pa}_X^{\mathcal{G}} = \boldsymbol{u}] = \sum_{m=1}^{M} \mathcal{P}(x, \boldsymbol{u} | \mathbf{o}[m], \boldsymbol{\theta}), \tag{2.2}$$

where $\mathbf{o}[m]$ is the assignment of $\mathbf{O}[m]$ in $\mathcal{D}$ and $\mathrm{ESS}_{\boldsymbol{\theta}}[X = x, \mathbf{Pa}_X^{\mathcal{G}} = \boldsymbol{u}]$ are the ESS, i.e., the expected counts in $\mathcal{D}$ of variable X and $\mathbf{Pa}_X^{\mathcal{G}}$ according to $\boldsymbol{\theta}$. Equation (2.2) requires the performance of inference for each of the M instances in the worst case when all the instances contain missing values. The new set of parameters is estimated from the completed dataset (M-step):

$$\hat{\boldsymbol{\theta}}_{x|\boldsymbol{u}} = \frac{\mathrm{ESS}_{\boldsymbol{\theta}}[x, \boldsymbol{u}]}{\mathrm{ESS}_{\boldsymbol{\theta}}[\boldsymbol{u}]}, \tag{2.3}$$

where $\hat{\boldsymbol{\theta}}_{x|\boldsymbol{u}}$ are the estimated parameters of the CPD $\mathcal{P}(X|\mathbf{Pa}_X^{\mathcal{G}})$. Both steps are repeated iteratively, and the likelihood of the parameters given $\mathcal{D}$ improves monotonically until convergence. EM has also been studied in the context of Bayesian learning [Ramoni and Sebastiani, 1997; McLachlan and Krishnan, 2008]. However, in Chapter 4 we focus on the frequentist approach.

In BNs [Lauritzen, 1995], the EM algorithm assumes a fixed structure $\mathcal{G}$ and optimizes only the component $\boldsymbol{\theta}$ of the pair $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$. When $\mathcal{G}$ is unknown, it is not straightforward to apply EM. The structural EM algorithm (SEM) [Friedman, 1997] extends EM to learn both the structure and parameters.

SEM includes structural learning in the M-step. Any score+search structure learning method can be used for this purpose; however, the scoring function to be maximized must be of the form described in Equation (2.1). SEM starts with a specified initial structure $\mathcal{G}_0$ and a set of parameters $\boldsymbol{\theta}_0$. At each iteration, SEM selects the model and parameters with the highest expected score given the previous assessment. The use of the expected score is motivated by the next inequality [Koller and Friedman, 2009]:

$$\mathrm{score}(\mathcal{D}, \mathcal{B}) - \mathrm{score}(\mathcal{D}, \mathcal{B}_0) \geq \mathrm{score}(\mathcal{D}^+, \mathcal{B}) - \mathrm{score}(\mathcal{D}^+, \mathcal{B}_0), \tag{2.4}$$

where $\mathcal{D}^+$ is the result of completing dataset $\mathcal{D}$ with BN $\mathcal{B}_0$. Intuitively, if $\mathcal{B}$ has a greater expected score than the model used to complete the data $\mathcal{B}_0$, then the score improvement with respect to the observed data is guaranteed. Hence, Equation (2.4) ensures that the SEM algorithm converges to a local optimum.

To compute the expected score of each BN candidate, the ESS (Equation (2.2)) are required. When the structure changes, a new set of ESS must be obtained, which requires the performance of inference in each case. This can be severely computationally demanding if the inference complexity of the models is not bounded. Hereafter, we refer to this method as soft SEM.

A less-demanding alternative is to complete the data according to its most probable assignment (E-step) and to estimate the MLE parameters from the completed dataset (M-

step) [Rancoita et al., 2016]. This strategy does not require the computation of ESS for each change in the BN. Rather, the scores of all the BN candidates can be computed directly from the completed dataset. We refer to this method as hard SEM. In hard SEM, the complete dataset $\mathcal{D}^+$ is obtained by imputing the MPE of the missing values given the observed values $\mathbf{o}[m]$ for each instance m of $\mathcal{D}$:

$$\mathbf{h}[m] = \arg\max_{\mathbf{h}[m]} \mathcal{P}(\mathbf{h}[m]|\mathbf{o}[m], \boldsymbol{\theta}), \tag{2.5}$$

where $\mathbf{h}[m]$ is an assignment of the missing values at the m-th instance of $\mathcal{D}$.

The main difference between hard and soft SEM is that the first optimizes over two objectives. In the E-step over the data completion $\mathcal{D}^+$ (see Equation (2.5)) and in the M-step over the model ($\max_{\mathcal{B}} \text{score}(\mathcal{D}^+, \mathcal{B})$). Unlike soft SEM, the model selected by hard SEM after the M-step is not guaranteed to have a greater score with respect to the observed data than the previous candidate.

Limiting the inference complexity of the models is key to executing SEM efficiently. With this objective, Scanagatta et al. [2018b] adopt the k-MAX algorithm in the M-step of hard SEM. They call the resulting method SEM-kMAX.

A relevant detail regarding the implementation of SEM-kMAX is that it adopts hard SEM rather than soft SEM. This is motivated by the unfeasibility of determining in advance what statistics will be required during the structure search. As k-MAX precomputes the scores beforehand at each iteration, in the majority of cases, it would be difficult to store the required number of sufficient statistics in memory. Scanagatta et al. [2018b] demonstrate that SEM-kMAX obtains promising results in the imputation experiments, yielding similar imputation accuracy to other well-known imputation methods in significantly less time.

2.4 Multidimensional classification with Bayesian networks

Van der Gaag and de Waal [2006] introduced MBCs as an extension of BN classifiers to multidimensional classification. MBCs are a special case of BNs with a restricted structure topology. Bielza et al. [2011] defined MBCs as follows:

Definition 2.11. ***(MBC)*** *An MBC is a BN $\mathcal{B}$ over a set of variables $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, where $\mathcal{X}$ is partitioned into two sets $\mathcal{C} = \{C_1, \ldots, C_d\}$, $d \geq 1$, of class variables and $\mathcal{F} = \{F_1, \ldots, F_l\}$, $l \geq 1$, of feature variables ($d+l = n$). The arcs in the structure $\mathcal{G}$ are partitioned into three subsets, A_C, A_F, A_B, such that:*

- *$A_C \subseteq \mathcal{C} \times \mathcal{C}$ is composed of the arcs between the class variables having a subgraph $\mathcal{G}_C = (\mathcal{C}, A_C)$ –class subgraph– of $\mathcal{G}$ induced by $\mathcal{C}$.*
- *$A_F \subseteq \mathcal{F} \times \mathcal{F}$ is composed of the arcs between the feature variables having a subgraph $\mathcal{G}_F = (\mathcal{F}, A_F)$ –feature subgraph– of $\mathcal{G}$ induced by $\mathcal{F}$.*
- *$A_B \subseteq \mathcal{C} \times \mathcal{F}$ is composed of the arcs from the class variables to the feature variables*

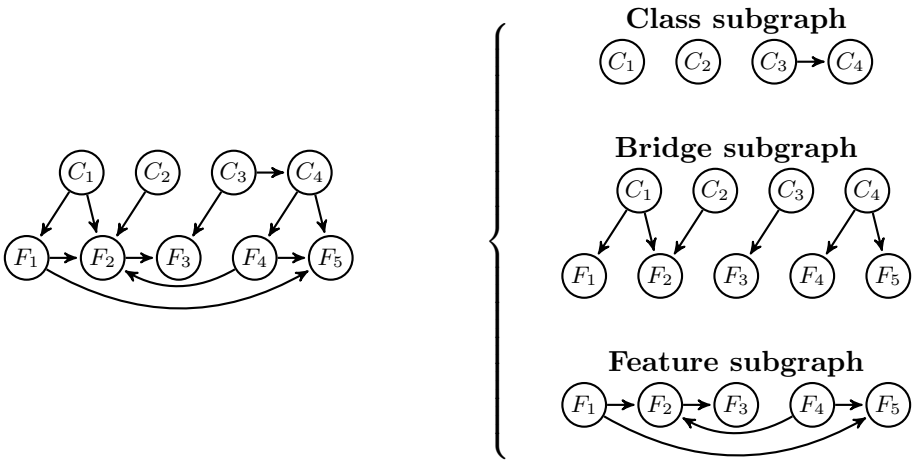


Figure 2.2: MBC structure.

having a subgraph $\mathcal{G}_B = (\mathcal{X}, A_B)$ –bridge subgraph– of $\mathcal{G}$ induced by $\mathcal{X}$ connecting class and feature variables.

Figure 2.2 shows an example of the structure of an MBC and its corresponding subgraphs.

The problem of multidimensional classification in MBCs involves getting the MPE of the class variables given an instantiation of the feature variables, which is given by

$$\mathbf{c}^* = \arg\max_{\mathbf{c}\in\Omega_{\mathcal{C}}} \mathcal{P}(\mathbf{c}|\mathbf{f}) = \arg\max_{\mathbf{c}\in\Omega_{\mathcal{C}}} \mathcal{P}(\mathbf{c},\mathbf{f}), \tag{2.6}$$

where $\mathbf{f}$ is an instantiation of $\mathcal{F}$ and $\Omega_{\mathcal{C}}$ is the set containing all the possible configurations of $\mathcal{C}$.

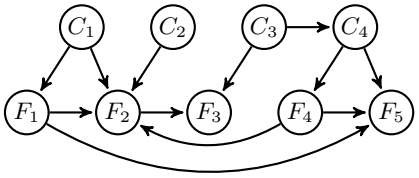
2.4.1 Class-bridge decomposable multidimensional Bayesian network classifiers

An MBC is class-bridge decomposable [Bielza et al., 2011] if it can be decomposed into multiple connected components, where each component is composed of all the nodes that are connected by an undirected path in $\mathcal{G}_C \cup \mathcal{G}_B$. Basically, the components of an MBC are the connected graphs obtained after removing the arcs of the feature subgraph from this MBC.

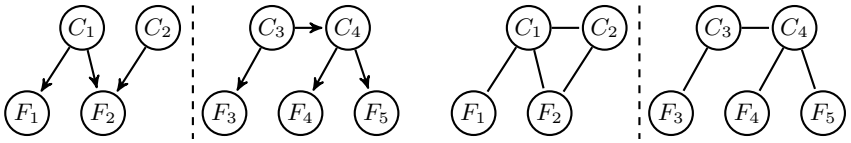
Definition 2.12. ***(CB-decomposable MBC)*** *A CB-decomposable MBC is a BN $\mathcal{B}$ whose class subgraph and bridge subgraph are decomposed into r maximal components such that:*

1. *$\mathcal{G}_C \cup \mathcal{G}_B = \bigcup_{i=1}^{r}(\mathcal{G}_{C_i} \cup \mathcal{G}_{B_i})$, where $\mathcal{G}_{C_i} \cup \mathcal{G}_{B_i}$, $i = 1,\ldots,r$, are its maximal connected components.*

2. *$\boldsymbol{Ch}_{\mathcal{C}_i}^{\mathcal{G}} \cap \boldsymbol{Ch}_{\mathcal{C}_j}^{\mathcal{G}} = \emptyset$, with $i,j = 1,\ldots,r$ and $i \neq j$, where $\boldsymbol{Ch}_{\mathcal{C}_i}^{\mathcal{G}}$ and $\boldsymbol{Ch}_{\mathcal{C}_j}^{\mathcal{G}}$ denote the children of all variables in $\mathcal{C}_i$ and $\mathcal{C}_j$ (the subsets of class variables in $\mathcal{G}_{C_i}$ and $\mathcal{G}_{C_j}$), respectively.*



(a) MBC structure



(b) Connected components

(c) Moralized connected components

Figure 2.3: Connected components of an MBC.

Bielza et al. [2011] proved that exploiting the CB-decomposability of MBCs can reduce the number of computations required to perform multidimensional classification. Specifically, they showed that the MPE can be computed independently in each component, given that

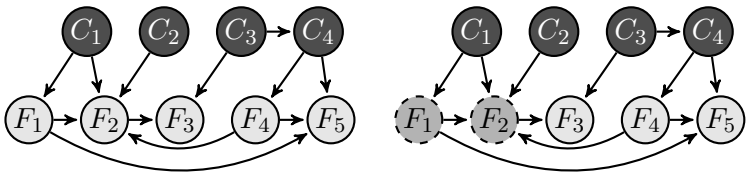
$$\max_{\mathbf{c}\in\Omega_{\mathcal{C}}} \mathcal{P}(\mathbf{c}|\mathbf{f}) \propto \prod_{i=1}^{r} \max_{\mathbf{c}_i\in\Omega_{\mathcal{C}_i}} \prod_{C_{ij}\in\mathcal{C}_i} \mathcal{P}(c_{ij}|\mathbf{Pa}_{C_{ij}}^{\mathcal{G}_C}) \prod_{F_{ijk}\in\mathbf{Ch}_{C_{ij}}^{\mathcal{G}_B}} \mathcal{P}(f_{ijk}|\mathbf{Pa}_{F_{ijk}}^{\mathcal{G}_B}, \mathbf{Pa}_{F_{ijk}}^{\mathcal{G}_F}), \tag{2.7}$$

where $\mathcal{C}_i$ is the set containing the class variables that belong to component i, $\Omega_{\mathcal{C}_i}$ is the set containing all the possible configurations of $\mathcal{C}_i$, c_{ij} is the value in $\mathbf{c}$ of the j-th variable in $\mathcal{C}_i$, and f_{ijk} is the value in $\mathbf{f}$ of the k-th child of C_{ij} in the bridge subgraph. This means that it is possible to maximize over each maximal connected component independently, therefore maximizing over lower dimensional spaces.

Let us consider the MBC shown in Figure 2.3, which can be CB-decomposed in two connected components that contain nodes $\{C_1, C_2, F_1, F_2\}$ and $\{C_3, C_4, F_3, F_4, F_5\}$, respectively. To classify an instance $\mathbf{f} = (f_1, \ldots, f_5)$ we should get the MPE of $(C_1, \ldots, C_4)$ given $\mathbf{f}$. By Equation (2.7) we know that for any $\mathbf{c} = (c_1, \ldots, c_4)$,

$$\begin{aligned}\max_{\mathbf{c}\in\Omega_{\mathcal{C}}} \mathcal{P}(\mathbf{c}|\mathbf{f}) \propto \Big(&\max_{c_1,c_2} \mathcal{P}(c_1)\mathcal{P}(c_2)\mathcal{P}(f_1|c_1)\mathcal{P}(f_2|c_1,c_2,f_1,f_4),\\ &\max_{c_3,c_4} \mathcal{P}(c_3)\mathcal{P}(c_4|c_3)\mathcal{P}(f_3|c_3,f_2)\mathcal{P}(f_4|c_4)\mathcal{P}(f_5|c_4,f_1,f_4)\Big).\end{aligned}$$

Thus, the MPE can be computed maximizing over (C_1, C_2) and (C_3, C_4) independently. In Chapter 5 we refer to the moralized connected components of MBCs (Figure 2.3c) as the prunded graph.



(a) (b)

Figure 2.4: Multidimensional classification with an MBC. In (a) the values $\boldsymbol{f} = (f_1, \ldots, f_5)$ of all the features are given, and multidimensional classification is equivalent to obtaining the MPE (i.e., $\arg\max_{\mathbf{c}\in\Omega_{\mathcal{C}}} \mathcal{P}(\mathbf{c}|\mathbf{f})$). In (b) the values of F_1 and F_2 are missing, and multidimensional classification is equivalent to obtaining the MAP (i.e., $\arg\max_{\mathbf{c}\in\Omega_{\mathcal{C}}} \mathcal{P}(\mathbf{c}|f_3, f_4, f_5)$).

2.4.2 Complexity of most probable explanations in multidimensional Bayesian network classifiers

Assuming that all the feature variables are observed, performing multidimensional classification in an MBC $\mathcal{B}$ with class variables $\mathcal{C} = \{C_1, \ldots, C_d\}$ and feature variables $\mathcal{F} = \{F_1, \ldots F_l\}$ is equivalent to obtaining the MPE of the class variables conditioned on an instance $\mathbf{f}$ of the features. If there are unobserved feature variables, performing multidimensional classification in $\mathcal{B}$ is equivalent to obtaining not the MPE in $(C_1, \ldots, C_d)$ but the MAP. This can be intractable even if the treewidth of $\mathcal{B}$ is bounded [Park, 2002].

Figure 2.4 shows an example of both cases where multidimensional classification in an MBC is equivalent to obtaining the MPE, in (a), and the MAP, in (b), respectively.

Existing research addresses the complexity of multidimensional classification in MBCs as the complexity of computing the MPE. Thus, they implicitly assume that MPE queries will not contain missing values (i.e., the values of all the feature variables will be given). Otherwise, the resulting MPE would provide the most probable instantiation of $(C_1, \ldots, C_d, F_{h_1}, \ldots, F_{h_k})$, where $F_{h_1}, \ldots, F_{h_k}$ are the non-instantiated features. Note that the most probable instantiation of $(C_1, \ldots, C_d)$, that is equivalent to the MAP of the class variables given an instantiation of the observed features in this case, may differ from the projection to $(C_1, \ldots, C_d)$ of the most probable instantiation of $(C_1, \ldots, C_d, F_{h_1}, \ldots, F_{h_k})$. In Chapter 5, we also focus on the case where all the feature variables are observed. Hence, we consider that, to perform multidimensional classification, an MBC obtains $\arg\max_{\mathbf{c}\in\Omega_{\mathcal{C}}} \mathcal{P}(\mathbf{c}|\mathbf{f})$.

MPE is generally NP-hard [Kwisthout, 2011] and known exact methods for MPE computations in a BN $\mathcal{B}$ are exponential in the treewidth of $\mathcal{G}$, where $\mathcal{G}$ is the structure of $\mathcal{B}$. Nevertheless, MPE can be computed in polynomial time in $\mathcal{B}$ if the treewidth of $\mathcal{G}$ is bounded [Sy, 1992]. Given MBC structural constraints, further bounds on their inference complexity have been found. De Waal and van der Gaag [2007] demonstrated that

$$\text{treewidth}(\mathcal{G}) \leq \text{treewidth}(\mathcal{G}_F) + d,$$

where $\mathcal{G}_F$ is the feature subgraph and d is the number of class variables. This means that $\mathcal{B}$ could perform multidimensional classification in polynomial time if the addition of the

treewidth of the feature subgraph and the number of class variables is bounded.

Furthermore, Kwisthout [2011] showed that for any CB-decomposable MBC with structure $\mathcal{G}$

$$\text{treewidth}(\mathcal{G}) \leq \text{treewidth}(\mathcal{G}_F) + |d_{\max}|,$$

where $|d_{\max}|$ is the number of class variables of the component with the maximum number of class variables. Hence, the MPE can be computed in polynomial time if the treewidth of $\mathcal{G}_F$ and the number of class variables of each component of $\mathcal{G}$ are bounded.

Pastink and van der Gaag [2015] focused on MBCs with an empty feature subgraph. To bound the structure, they used

$$\text{treewidth}(\mathcal{G}_{\bar{F}}) < \text{treewidth}(\mathcal{G}'),$$

where $\mathcal{G}_{\bar{F}}$ is the structure of an MBC with empty feature subgraph and $\mathcal{G}'$ is the graph obtained after moralizing $\mathcal{G}_{\bar{F}}$ (see Figure 2.3c) and then removing all its feature nodes from the moralized graph.

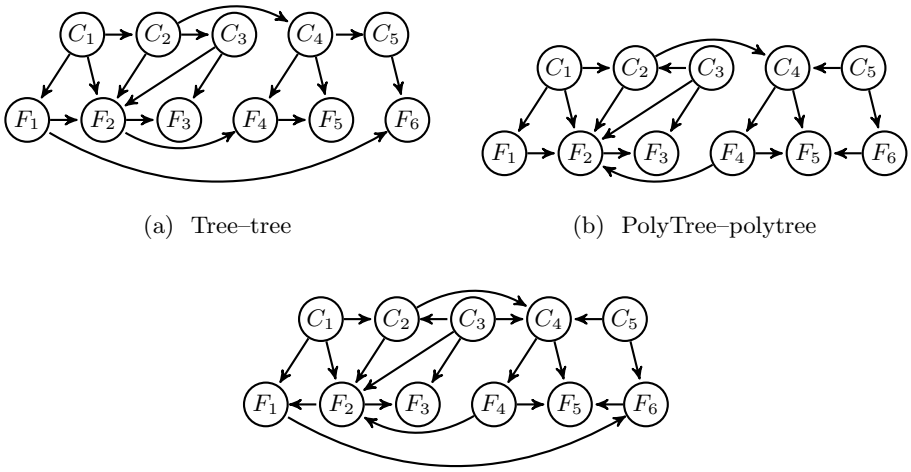
2.4.3 Previous work on learning multidimensional Bayesian network classifiers

The problem of learning MBCs from data has been addressed before. The literature contains methods for learning different families of MBCs, depending on the type of class and feature subgraphs that they can obtain (trees, forests, polytrees or DAGs). Here we denote the family of the MBC using <class subgraph> – <feature subgraph> (e.g., Tree–DAG has a tree as the class subgraph and a DAG as the feature subgraph). Figure 2.5 shows some of the most popular MBC families.

Methods have been proposed for learning Tree–tree [van der Gaag and de Waal, 2006], PolyTree–polytree [de Waal and van der Gaag, 2007] and DAG–DAG [Bielza et al., 2011] MBCs. These approaches do not explicitly consider the inference complexity of the learned models. Hence, they may lead to MBCs where the MPE cannot be solved efficiently, unless the number of class variables is very small.

There are also other approaches in the literature that consider the complexity of the MBCs during the learning process. Corani et al. [2014] proposed a method for learning sparse MBCs with a forest class subgraph and an empty feature subgraph, and Borchani et al. [2010] introduced the first method to learn CB-decomposable MBCs. However, neither provides guarantees regarding the complexity of multidimensional classification in the models. Pastink and van der Gaag [2015] proposed a method for learning Tree–empty MBCs of bounded treewidth, providing an optional step to learn a forest feature subgraph, and guaranteeing the tractability of the resulting models. The method computes the treewidth of each candidate and rejects any that exceeds the treewidth bound.

Markov random fields have also been used for multi-label classification. Ghamrawi and McCallum [2005] proposed two pairwise models, the collective multilabel classifier (CML) and the collective multilabel classifier with features (CMLF). CML learns a factor between



(a) Tree–tree

(b) PolyTree–polytree

(c) DAG–DAG

Figure 2.5: Widely used MBC families ordered from the least general (a) to the most general (c). Note that Tree–tree is a special case of polyTree–polytree, which is likewise a special case of DAG–DAG.

each pair of class variables and between each pairwise combination of a class variable and a feature variable. CMLF also learns the latter factors, but instead of learning the former it learns a factor between each combination of two class variables and a feature variable, increasing the expressiveness of CML. Exact inference in these models requires computing a factor over all the possible configurations of the class variables. Hence, it is intractable when the number of class variables is not small. Arias et al. [2016] proposed a method that learns an undirected graph between the class variables, and learns a base model (e.g., naïve Bayes) for each pair of connected class variables. The base model gives a factor over a pair of class variables given an instance of the feature variables. The drawback of this approach is that the number of base models is huge if the graph between the class variables is not sparsely connected.

In Chapter 5 we bound the complexity of (the most general) DAG–DAG MBCs by bounding the treewidth of a transformation of their structures (similar to the transformation used by Pastink and van der Gaag [2015]). However, we do not bound the treewidth of their complete structures. We use these bounds to learn MBCs where multidimensional classification can be performed in polynomial time. We show that even high treewidth MBCs may be tractable given other structural constraints.

2.4.4 Discriminative learning of Bayesian networks

In classification, generative methods usually result in suboptimal classification performance compared to discriminative methods [Friedman et al., 1997; Grossman and Domingos, 2004]. Analogously, in multidimensional classification, wrapper approaches that optimize a discrim-

inative score (e.g., accuracy) show promising results compared to generative methods [Bielza et al., 2011].

An alternative that has been successfully applied for learning BN classifiers is to directly optimize the conditional log-likelihood (CLL) of the models instead of a generative score (eg., log-likelihood or information gain). CLL is defined as:

$$\mathrm{CLL}(\mathcal{B}|\mathcal{D}) = \sum_{m=1}^{M} \log \mathcal{P}(\boldsymbol{c}[m]|\boldsymbol{f}[m], \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ are the parameters of $\mathcal{B}$. It can also be formulated according to the log-likelihood:

$$\mathrm{CLL}(\mathcal{B}|\mathcal{D}) = \ell(\mathcal{B}|\mathcal{D}) - \ell_F(\mathcal{B}|\mathcal{D}), \tag{2.8}$$

where

$$\ell_F(\mathcal{B}|\mathcal{D}) = \sum_{m=1}^{M} \log \mathcal{P}(\boldsymbol{f}[m]|\boldsymbol{\theta}).$$

Analogously, the log-likelihood can be formulated as:

$$\ell(\mathcal{B}|\mathcal{D}) = \mathrm{CLL}(\mathcal{B}|\mathcal{D}) - \ell_F(\mathcal{B}|\mathcal{D}),$$

Friedman et al. [1997] speculated that one of the problems of optimizing log-likelihood is that often the term $\ell_F(\mathcal{B}|\mathcal{D})$ is much bigger than $\mathrm{CLL}(\mathcal{B}|\mathcal{D})$, and therefore the optimization focuses on maximizing $\ell_F(\mathcal{B}|\mathcal{D})$ rather than $\mathrm{CLL}(\mathcal{B}|\mathcal{D})$. The problems of the discriminative learning of the parameters and the structure of BN classifiers have been studied separately.

Greiner et al. [2005] proposed the extended logistic regression (ELR) algorithm, that uses conjugate gradient [Bishop, 1995] to optimize CLL. Roos et al. [2005] showed that learning the parameters in BN classifiers is equivalent to solving logistic regression when their structure satisfies a certain graph-theoric property. Su et al. [2008] provided a discriminative parameter learning method, called discriminative frequency estimate, that yields similar accuracy than ELR with a significantly lower computational cost. Zaidi et al. [2017] used discriminative learning as an initialization to speed-up the process, and then updated the parameters using a variant of gradient ascent, maximizing CLL.

To learn the structure of BN classifiers in a discriminative manner, Grossman and Domingos [2004] found that the simple heuristic of estimating the parameters via maximum likelihood and optimizing the conditional likelihood of the structure was both accurate and efficient. Their experiments suggested that optimizing the CLL of both the parameters and the structure does not improve the results of the simpler approach, where the maximum likelihood parameters are chosen.

Carvalho et al. [2011] proposed an approximation of the CLL for binary classification called factorized CLL (fCLL) with several desirable properties. Namely, the score is decomposable, and the approximation is unbiased and has minimum variance. Carvalho et al. [2013] extended fCLL to multi-class problems.

To the best of our knowledge, the only proposals that focus on the discriminative learning of MBCs are wrapper methods that guide the structure search according to a measure of accuracy [Borchani et al., 2010; Bielza et al., 2011]. However, these approaches are usually extremely computationally demanding when the number of class variables is not very small.

Chapter 3

Learning tractable Bayesian networks in the space of elimination orders

A common approach for learning BNs from data is to perform a search process optimizing a scoring function that measures the quality of each structure. Well-known scoring functions such as BDe [Heckerman et al., 1995], K2 [Cooper and Herskovits, 1992], AIC [Akaike, 1974], BIC [Schwarz, 1978] or MDL [Bouckaert, 1993; Lam and Bacchus, 1994] implicitly or explicitly penalize the number of network parameters. The representation complexity, which is given by the number of network parameters, does not place an upper bound on the inference complexity of the models, and a model with a low representation complexity can have a high inference complexity. Thus, more precise estimations of the inference complexity are required to ensure the tractability of models during the learning process.

A good indicator of the inference complexity of a BN $\mathcal{B}$ with structure $\mathcal{G}$ is the treewidth of $\mathcal{G}$, given that the most widely used exact inference methods for BNs, like VE or message passing in JTs, can be computed in exponential time in $\text{tw}(\mathcal{G})$. It is NP-hard to determine the treewidth of a graph [Arnborg et al., 1987], and there are no efficient exact methods for solving this problem. Many heuristics have been proposed for treewidth estimation (see Section 2.2.2), but most are very computationally demanding. This is especially important when BNs have to be learned from data, since we have to compute the treewidth of each candidate during the learning process to ensure tractability.

The treewidth of a graph $\mathcal{G}$ can also be expressed as the width of the optimal EO π_{opt} for graph $\mathcal{G}$. This means that obtaining an optimal EO of $\mathcal{G}$ is equivalent to obtaining the treewidth of $\mathcal{G}$, and it is also an NP-hard problem. Hence, one way of getting an accurate estimation of $\text{tw}(\mathcal{G})$ is to find a good EO for $\mathcal{G}$. It would often be intractable to get a good EO from scratch for each candidate network during the structure search. As most structure learning methods perform local changes in $\mathcal{G}$ during the learning process, a more efficient solution to this problem is to incrementally update the EOs for each local change performed

in $\mathcal{G}$.

There are usually multiple equivalent EOs for $\mathcal{G}$. This means that the combined space of DAGs and EOs is highly redundant, and it would be extremely computationally demanding to search for low complexity structures in this space. In our first approach we restricted the search to topological EOs [Benjumeda et al., 2015a] to reduce the search space. We performed experiments that suggested that learning tractable models and using exact inference may produce more precise predictions than using traditional learning methods and relying on approximate inference. Lowd and Domingos [2008] arrived at similar conclusions when they evaluated the performance of method LearnAC. However, the width of an optimal topological EO is sometimes many times higher than the width of the optimal EO. This may lead to gross overestimations of the treewidth.

In this chapter, we define a type of elimination trees (ETs) [Grant and Horsch, 2009], which we call valid ETs, that avoid the redundancy of the combined space of DAGs and EOs. A single valid ET can be used to represent all the EOs that are equivalent (i.e., induce the same factors during VE) for any graph $\mathcal{G}$. We propose methods for efficiently compiling each possible local change that could be applied in $\mathcal{G}$ (i.e., arc additions, removals or reversals), and provide a framework for learning valid ETs from data using the above methods.

The rest of this chapter is organized as follows. Section 3.1 shows the relation between ETs and EOs, and the way the former can be used as an equivalence class of EOs and DAGs. Section 3.2 describes the proposed compilation and optimization methods. It shows how to use ETs to learn tractable BNs in the space of EOs. Section 3.3 reports the experimental results. Section 3.4 outlines the concluding remarks and future research lines.

This chapter is derived from Benjumeda et al. [2019a], and extends the work in Benjumeda et al. [2015a,b] to EOs that are not topological. The software of the proposed method is available at `https://github.com/marcobb8/tr_bn`.

3.1 Elimination trees

This chapter addresses the problem of learning bounded treewidth BNs. We focus on choosing a compact representation of the combined space of DAGs and EOs and a set of operators that allow efficiently moving in this space for the next reasons: First, this search space does not put any restrictions on the structure beyond the treewidth bound. Second, given addition, removal and reversal operators, most score+search BN learning methods can be easily adapted to learn bounded treewidth BNs.

In a BN $\mathcal{B}$ over $\mathcal{X} = \{X_1, \ldots, X_n\}$, there are $n!$ different EOs of $\mathcal{X}$, although many are usually equivalent (Definition 2.9) for the structure $\mathcal{G}$ of $\mathcal{B}$, especially when $\mathcal{G}$ is not densely connected. We need to avoid this redundancy to reduce the size of the search space during the learning process. Next, we define elimination trees (ETs), a representation that is especially well suited for this purpose. ETs are based on the representation proposed by Grant and Horsch [2009] for recursive conditioning, which we adapt to represent a set $\mathcal{S}$ of EOs for $\mathcal{B}$.

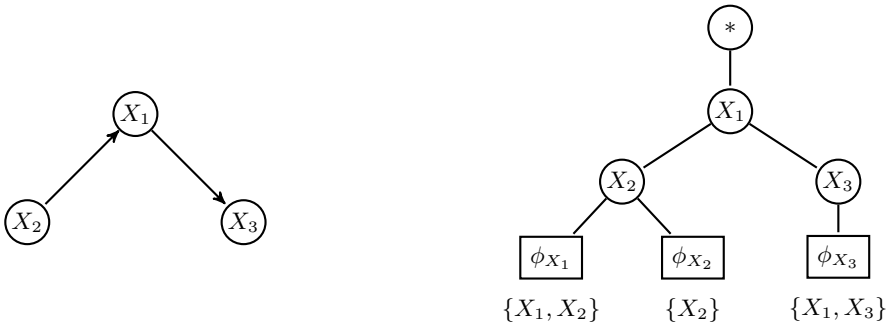


Figure 3.1: Structures of a BN $\mathcal{B}$ (left) and an ET $\mathcal{E}_\mathcal{B}$ (right). In $\mathcal{E}_\mathcal{B}$, $*$ is the root node, X_1, X_2 and X_3 are the inner nodes, and ϕ_{X_1}, ϕ_{X_2} and ϕ_{X_3} are the leaves and potentials. The domain of the potential of each leaf node ϕ_{X_i} is illustrated below the respective node. $\mathcal{E}_\mathcal{B}$ represents the EOs (X_2, X_3, X_1) and (X_3, X_2, X_1) for $\mathcal{B}$.

Definition 3.1. ***(Elimination tree)*** *Let $\mathcal{B}$ be a BN over $\mathcal{X} = \{X_1, \ldots, X_n\}$. An elimination tree $\mathcal{E}_\mathcal{B}$ over $\mathcal{X}$ is composed of:*

- *A set of factors or potentials $\phi_{X_1}, \ldots, \phi_{X_n}$ that represent the parameters of $\mathcal{B}$ of each node $X_1, \ldots, X_n$.*
- *A tree $\mathcal{T}$ composed of a root node, $*$, an inner node (node with parent and children) for each variable $X_i \in \mathcal{X}$, and a leaf node labelled ϕ_{X_i} for each potential ϕ_{X_i}. The nodes are connected by undirected edges.*

Assuming that we use VE over an ET $\mathcal{E}_\mathcal{B}$ to perform inference, the topology of the tree shows the orders in which each variable $X_i \in \mathcal{X}$ should be eliminated from the factors of the model. If an inner node X_i is the predecessor (this precedence must be read from the root node to the leaves) of another inner node X_j, X_i is eliminated after X_j.

Definition 3.2. ***(ET representation of an EO)*** *Let $\mathcal{B}$ be a BN over $\mathcal{X} = \{X_1, \ldots, X_n\}$. An elimination tree $\mathcal{E}_\mathcal{B}$ represents an EO π for $\mathcal{B}$ if, for each $X_i, X_j \in \mathcal{X}$, $(X_i < X_j)_\pi$ implies that $X_j \notin \boldsymbol{Desc}_{X_i}^{\mathcal{E}_\mathcal{B}}$ (the descendants of X_i in $\mathcal{E}_\mathcal{B}$). $\mathcal{E}_\mathcal{B}$ represents a set of EOs $\mathcal{S}$ for $\mathcal{B}$ if it represents each $\pi_i \in \mathcal{S}$ for $\mathcal{B}$.*

Figure 3.1 shows an ET $\mathcal{E}_\mathcal{B}$ that represents the set of EOs $\mathcal{S}$ for the probability distribution $\mathcal{P}(X_1, X_2, X_3) = \phi_{X_1}(X_1, X_2) \cdot \phi_{X_2}(X_2) \cdot \phi_{X_3}(X_1, X_3)$. As X_1 is a predecessor in $\mathcal{E}_\mathcal{B}$ of X_2 and X_3, $\mathcal{E}_\mathcal{B}$ represents each EO π such that $(X_2 < X_1)_\pi$ and $(X_3 < X_1)_\pi$, that is, (X_2, X_3, X_1) and (X_3, X_2, X_1).

Let us again consider the product of marginals. If we have a BN $\mathcal{B}$ over $\mathcal{X} = \{X_1, \ldots, X_n\}$ that represents the probability distribution $\mathcal{P}(X_1, \ldots, X_n) = \phi_{X_1}(X_1) \cdots \phi_{X_n}(X_n)$, all the EOs of $X_1, \ldots, X_n$ are equivalent for $\mathcal{B}$. This can be represented by a single ET, as shown in Figure 3.2.

Inference in ETs is straightforward. Given an ET $\mathcal{E}_\mathcal{B}$ that represents a set of EOs $\mathcal{S}$ for $\mathcal{B}$, we could use any EO $\pi_i \in \mathcal{S}$ to perform VE, or to efficiently compile $\mathcal{B}$ into a JT or an AC.

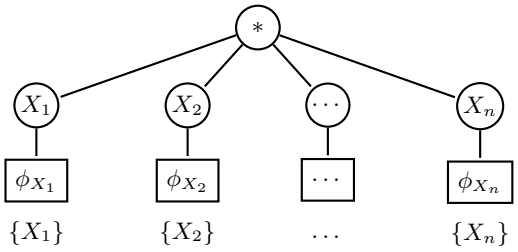


Figure 3.2: Structure of an ET that represents the product of marginals. Below each leaf node ϕ_{X_i}, the domain of its corresponding potential is shown.

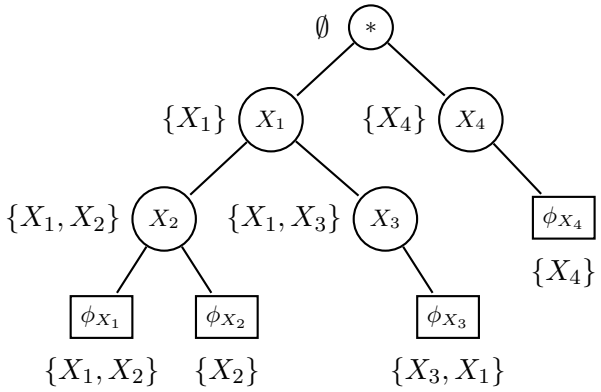


Figure 3.3: Structure of an ET. The clusters of the ET are shown near to their corresponding nodes.

3.1.1 Properties of elimination trees

In this section, we introduce some terms that we use in the rest of the chapter. Let $\mathcal{E}_\mathcal{B}$ be an ET over $\mathcal{X} = \{X_1, \ldots, X_n\}$. We use $\mathrm{Pa}_{X_i}^{\mathcal{E}_\mathcal{B}}$ and $\mathbf{Ch}_{X_i}^{\mathcal{E}_\mathcal{B}}$ to refer to the parent and the children of node X_i in $\mathcal{E}_\mathcal{B}$. $\mathbf{Pred}_{X_i}^{\mathcal{E}_\mathcal{B}}$ is the set of predecessor of X_i in $\mathcal{E}_\mathcal{B}$. For example, $\mathbf{Pred}_{X_1}^{\mathcal{E}_\mathcal{B}} = \{*\}$ and $\mathbf{Desc}_{X_1}^{\mathcal{E}_\mathcal{B}} = \{X_2, X_3, \phi_{X_1}, \phi_{X_2}, \phi_{X_3}\}$ in the ET shown in Figure 3.3.

Given a factor $\phi_{X_i}(X_{i(1)}, \ldots, X_{i(n_i)})$, $\mathbf{Dom}(\phi_{X_i})$ represents its domain, that is, the set of nodes $\{X_{i(1)}, \ldots, X_{i(n_i)}\}$, where $X_{i(1)}, \ldots, X_{i(n_i)} \in \mathcal{X}$ and n_i is the cardinality of $\mathbf{Dom}(\phi_{X_i})$. We use $\mathbf{Leaves}(\mathcal{E}_\mathcal{B})$ to refer to the set of leaf nodes in $\mathcal{E}_\mathcal{B}$.

ETs closely resemble dtrees, a representation used for recursive conditioning [Darwiche, 2001]. Unlike ETs, dtrees are full binary trees (i.e., trees in which any inner node has two children), and their inner nodes are labeled with a set of variables instead of being labeled with a single variable. There follows a definition of clusters in ETs, which is analogous to the definition of clusters given by Darwiche [2009] for dtrees.

Definition 3.3. ***(Clusters of ET nodes)*** *The cluster of an inner node X_i in an ET $\mathcal{E}_\mathcal{B}$ is defined as:*

$$\boldsymbol{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) := \left(\bigcup_{X_j \in \boldsymbol{Ch}_{X_i}^{\mathcal{E}_\mathcal{B}}} \boldsymbol{Cls}_{\mathcal{E}_\mathcal{B}}(X_j) \setminus \{X_j\} \right).$$

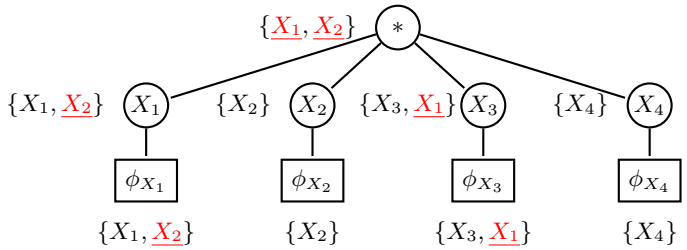


Figure 3.4: Structure of an unsound ET. The clusters of the ET are shown near to their respective nodes, and the variables that compromise the soundness of the ET are underlined.

The cluster of a leaf node ϕ_{X_i} in $\mathcal{E}_\mathcal{B}$ is defined as:

$$\boldsymbol{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_i}) := \boldsymbol{Dom}(\phi_{X_i}).$$

When we perform VE in $\mathcal{E}_\mathcal{B}$, $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i)$ is equivalent to the cluster (domain of the generated factor) induced by eliminating X_i. Figure 3.3 shows an example of the clusters $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i)$ of an ET $\mathcal{E}_\mathcal{B}$. The clusters of ETs and the clusters (or cliques) of JTs are also closely related (see Section 3.1.1.1).

3.1.1.1 Valid elimination trees

The purpose of using ETs to search for structures with a small treewidth is to reduce the combined space of DAGs and EOs, and consequently allow efficient algorithms for learning bounded treewidth BNs. By the above definition, there are many solutions that are incorrect or redundant. To identify and avoid such ETs during the learning process, we define two new properties: soundness and completeness.

We say that an ET $\mathcal{E}_\mathcal{B}$ is sound if all the EOs that it represents are equivalent for $\mathcal{B}$.

Definition 3.4. ***(Sound ETs)*** *Let $\mathcal{E}_\mathcal{B}$ be an ET over $\mathcal{X}$. Node X_i is sound for $\mathcal{E}_\mathcal{B}$ if $\boldsymbol{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \subseteq \boldsymbol{Pred}_{X_i}^{\mathcal{E}_\mathcal{B}} \cup \{X_i\}$. A leaf node $\phi_{X_i} \in \boldsymbol{Leaves}(\mathcal{E}_\mathcal{B})$ is sound for $\mathcal{E}_\mathcal{B}$ if $\boldsymbol{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_i}) \subseteq \boldsymbol{Pred}_{\phi_{X_i}}^{\mathcal{E}_\mathcal{B}}$. $\mathcal{E}_\mathcal{B}$ is sound if every node (inner and leaf nodes) is sound for $\mathcal{E}_\mathcal{B}$.*

Figure 3.4 shows the structure of an unsound ET $\mathcal{E}_\mathcal{B}$. Given that there are no ancestral relationships between the inner nodes in $\mathcal{E}_\mathcal{B}$, it represents all the possible permutations of $\{X_1, X_2, X_3, X_4\}$ as EOs. The clusters of some nodes contain variables (underlined) that are not their predecessors in $\mathcal{E}_\mathcal{B}$. For example, $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_1)$ contains X_1 and X_2, but X_2 is not a predecessor of X_1. As there is no ancestral relationship between X_1 and X_2 in $\mathcal{E}_\mathcal{B}$, it is equivalent whether $\mathcal{E}_\mathcal{B}$ eliminates X_1 before or after X_2. Unfortunately, this is not true, as eliminating X_2 before X_1 would induce cluster $\{X_1, X_2\}$. However, this cluster cannot be induced by any EO π_1 where $(X_1 < X_2)_{\pi_1}$ because X_1 will have been eliminated from all factors before X_2 has been eliminated. Thus, if there is a variable that belongs to the cluster of a node X_i that is not one of the predecessors of X_i in $\mathcal{E}_\mathcal{B}$, then the $\mathcal{E}_\mathcal{B}$ is not sound, and it represents EOs that are not equivalent.

The completeness of ETs is analogous to the completeness of a set of EOs.

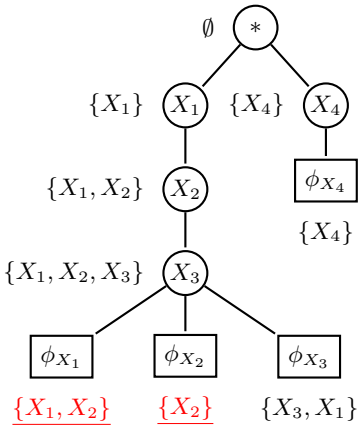


Figure 3.5: Structure of an incomplete ET. The clusters of the ET are shown near to their respective nodes, and the clusters that compromise the completeness of the ET are underlined.

Definition 3.5. ***(Complete ETs)*** *Let $\mathcal{E}_\mathcal{B}$ be an ET over $\mathcal{X}$. Node $X_i \in \mathcal{X} \cup \textbf{\textit{Leaves}}(\mathcal{E}_\mathcal{B})$ (i.e., X_i is either an inner node from $\mathcal{X}$ or a leaf node from $\textbf{\textit{Leaves}}(\mathcal{E}_\mathcal{B})$) is complete for $\mathcal{E}_\mathcal{B}$ if $Pa_{X_i}^{\mathcal{E}_\mathcal{B}} \in \textbf{\textit{Cls}}_{\mathcal{E}_\mathcal{B}}(X_i)$ or $Pa_{X_i}^{\mathcal{E}_\mathcal{B}} = *$. $\mathcal{E}_\mathcal{B}$ is complete if every node (inner and leaf nodes) is complete for $\mathcal{E}_\mathcal{B}$.*

Figure 3.5 shows the structure of an incomplete ET $\mathcal{E}_\mathcal{B}$. It represents the EOs $\mathcal{S} = \{(X_3, X_2, X_1, X_4), (X_3, X_2, X_4, X_1), (X_3, X_4, X_2, X_1), (X_4, X_3, X_2, X_1)\}$, but there are other EOs that are equivalent for $\mathcal{B}$ that are not represented by $\mathcal{E}_\mathcal{B}$. For example, (X_2, X_3, X_1, X_4) is equivalent to (X_3, X_2, X_1, X_4) given that the clusters induced after eliminating X_2 and X_3 are $\{X_1, X_2\}$ and $\{X_1, X_3\}$ in both cases.

Definition 3.6. ***(Valid ETs)*** *An ET $\mathcal{E}_\mathcal{B}$ is valid if it is sound and complete.*

The ET shown in Figure 3.3 is sound (for every node X_i, all the variables in its cluster are either its predecessors or X_i) and complete (for every node X_i with parent X_p, the cluster of X_i contains X_p). This means that it is valid. The space of valid ETs does not contain incorrect or redundant solutions.

The process described by Algorithm 3.1 yields a valid ET $\mathcal{E}_\mathcal{B}$ given a BN $\mathcal{B}$ and an EO π.

Algorithm 3.1 starts with an ET where the parent of every node is the root node $*$ (line 1). First, the variables in $\mathcal{X}$ are visited in the order given by π (line 2). When variable X_i is visited, node X_i is set as the parent of the nodes whose cluster contains X_i and whose parent is the root node $*$ in the ET (lines 3–7). This is analogous to the process of eliminating variable X_i from $\mathcal{B}$. The cluster $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i)$ of X_i in the ET $\mathcal{E}_\mathcal{B}$ is output in the same way as the cluster $\mathbf{Cls}_\pi(X_i)$ induced by eliminating X_i using π in $\mathcal{B}$, and they are equal. The complete process con be computed efficiently.

Proposition 3.1. *Algorithm 3.1 executes in polynomial time in the number of variables n.*

Proof. Line 1 consumes linear time. The loops at lines 2 and 3 iterate over at most n values. Obtaining the cluster at line 4 requires polynomial time (see Lemma A.1), and the set operations at line 3 can be computed in polynomial time. □

Input: BN $\mathcal{B}$ over $\mathcal{X}$, EO π
Output: Valid ET $\mathcal{E}_\mathcal{B}$

let $\mathcal{E}_\mathcal{B}$ be an ET with inner nodes $\mathcal{X}$ and factor nodes $\textbf{Leaves}(\mathcal{E}_\mathcal{B})$ where $\forall X_i \in \mathcal{X} \cup \textbf{Leaves}(\mathcal{E}_\mathcal{B})$, $\mathrm{Pa}_{X_i}^{\mathcal{E}_\mathcal{B}} = *$;
for $X_i \in \pi$ **do**
  for $X_j \in (\mathcal{X} \cup \boldsymbol{Leaves}(\mathcal{E}_\mathcal{B})) \cap \boldsymbol{Ch}_*^{\mathcal{E}_\mathcal{B}}$ **do**
    if $X_i \in \boldsymbol{Cls}_{\mathcal{E}_\mathcal{B}}(X_j)$ **then**
      $\mathrm{Pa}_{X_j}^{\mathcal{E}_\mathcal{B}} \leftarrow X_i$;
    end
  end
end
return $\mathcal{E}_\mathcal{B}$;

Algorithm 3.1: Compile a valid ET from an EO and a BN.

Proposition 3.2 states that given an EO π and a BN $\mathcal{B}$, Algorithm 3.1 returns always valid ETs. Hence, there is at least one valid ET for π and $\mathcal{B}$.

Proposition 3.2. *Let $\mathcal{B}$ be a BN over $\mathcal{X}$ and π an EO of $\mathcal{X}$. Algorithm 3.1 returns a valid ET that represents π for $\mathcal{B}$.*

Proof. Algorithm 3.1 ensures that when any variable X_i is visited, it is set as the parent of every node whose cluster contains X_i and whose parent is the root node $*$. Therefore:

- If the cluster of node X_j contains X_i when X_i is visited, the cluster of each of its predecessors also contains X_i, given that node X_i has no children until X_i has been visited. When X_i is visited, it is set as a predecessor of all the nodes whose cluster contains X_i. After visiting X_i, there are no nodes whose clusters contain X_i that are not their descendants in $\mathcal{E}_\mathcal{B}$. As this applies to each node $X_i \in \mathcal{X}$, all the nodes in $\mathcal{E}_\mathcal{B}$ must be sound, making $\mathcal{E}_\mathcal{B}$ sound.
- The cluster of every node X_j that is a child of X_i contains X_i. Each X_j is complete for $\mathcal{E}_\mathcal{B}$, making $\mathcal{E}_\mathcal{B}$ complete.
- A node X_j can only be a descendant of a node X_i in $\mathcal{E}_\mathcal{B}$ if $(X_j < X_i)_\pi$. Hence, $\mathcal{E}_\mathcal{B}$ represents π.

As $\mathcal{E}_\mathcal{B}$ is valid (sound and complete) and represents π, there is at least one valid $\mathcal{E}_\mathcal{B}$ for $\mathcal{B}$ and π. □

Proposition 3.3 ensures that there is a single valid ET $\mathcal{E}_\mathcal{B}$ that represents an EO π for a BN $\mathcal{B}$.

Proposition 3.3. *Let $\mathcal{B}$ be a BN over $\mathcal{X}$ and π an EO of $\mathcal{X}$. There is exactly one valid ET $\mathcal{E}_\mathcal{B}$ that represents π for $\mathcal{B}$.*

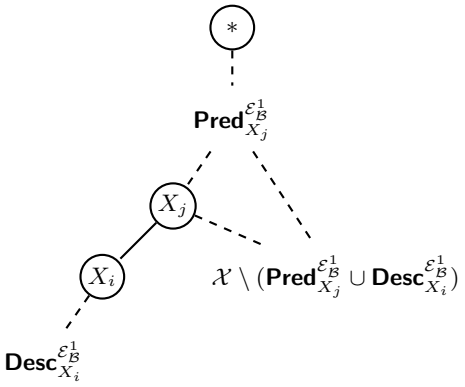


Figure 3.6: State of the ET $\mathcal{E}^1_{\mathcal{B}}$ during the inductive step of the proof of Propositions 3.3 and 3.4.

Proof. From Proposition 3.2, we know that there is at least one valid ET for $\mathcal{B}$ and π. We prove that there is exactly one by structural induction. We consider two ETs $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ for $\mathcal{B}$ and π. We show that if $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are valid, then, for each node $X_i \in \mathcal{X}$, $\mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}} = \mathrm{Pa}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}}$, starting from the leaves (base case). This means that $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are the same, which implies that there is a single valid ET for $\mathcal{B}$ and π.

Base case:

The subtrees that have $\phi_{X_i} \in \mathbf{Leaves}(\mathcal{E}_{\mathcal{B}})$ as its root in $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are only composed of node ϕ_{X_i}. Hence, they are equal.

Inductive step:

Assume that the subtrees hanging from node X_i in $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are equal. Let $X_j = \mathrm{Pa}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}}$ (Figure 3.6). As $\mathcal{E}^1_{\mathcal{B}}$ is valid, if $X_j \neq \mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}}$, then $\mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}}$ is a node X_k where either:

a) $X_k \in \mathbf{Pred}_{X_j}^{\mathcal{E}^1_{\mathcal{B}}}$: then $\mathcal{E}^2_{\mathcal{B}}$ would not represent π because $(X_j < X_k)_\pi$, or

b) $X_k \in \mathbf{Desc}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}}$: then $\mathcal{E}^2_{\mathcal{B}}$ would not represent π because $(X_k < X_i)_\pi$, or

c) $X_k \in \mathcal{X} \setminus (\mathbf{Pred}_{X_j}^{\mathcal{E}^1_{\mathcal{B}}} \cup \mathbf{Desc}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}})$: then $\mathcal{E}^2_{\mathcal{B}}$ is not complete because $\mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}} = X_j \notin \mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}}}(X_i)$.

This means that if $\mathcal{E}^2_{\mathcal{B}}$ is valid, then the following condition $\mathrm{Pa}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}} = \mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}}$ holds. □

Valid ETs avoid the redundancy between EOs and DAGs. We demonstrate that a valid ET $\mathcal{E}_{\mathcal{B}}$ represents a complete set of equivalent EOs for $\mathcal{B}$.

Proposition 3.4. *Let $\mathcal{B}$ be a BN over $\mathcal{X}$, π an elimination EO of $\mathcal{X}$, and $\mathcal{E}^1_{\mathcal{B}}$ a valid ET that represents π_1 for $\mathcal{B}$. For each EO π_2 equivalent to π_1 for $\mathcal{B}$ (Definition 2.9), $\mathcal{E}^1_{\mathcal{B}}$ also represents π_2.*

Proof. Let $\mathbf{Cl}_i^{\pi_1}$ and $\mathbf{Cl}_i^{\pi_2}$ be the cluster induced by VE after eliminating X_i from $\mathcal{B}$ using the EO π_1 and π_2, respectively. As π_1 and π_2 are equivalent for $\mathcal{B}$, $\mathbf{Cl}_i^{\pi_1} = \mathbf{Cl}_i^{\pi_2}$.

From Proposition 3.3 we know that there is a single valid ET $\mathcal{E}^1_{\mathcal{B}}$ that represents π_1 for $\mathcal{B}$, and also a single valid ET $\mathcal{E}^2_{\mathcal{B}}$ that represents π_2 for $\mathcal{B}$. We prove that $\mathcal{E}^1_{\mathcal{B}}$ also represents π_2 by structural induction. We show that if $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are valid, then, for each of node $X_i \in \mathcal{X}$, starting from the leaves (base case), $\mathrm{Pa}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}} = \mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}}$. This means that $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are the same, which implies that $\mathcal{E}^1_{\mathcal{B}}$ also represents π_2 for $\mathcal{B}$.

Base case:

The subtrees whose root is $\phi_{X_i} \in \mathbf{Leaves}(\mathcal{E}^1_{\mathcal{B}})$ in $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are composed of node ϕ_{X_i} only. Hence, they are equal.

Inductive step:

Assume that the subtrees hanging from node X_i in $\mathcal{E}^1_{\mathcal{B}}$ and $\mathcal{E}^2_{\mathcal{B}}$ are equal. Let $X_j = \mathrm{Pa}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}}$ (Figure 3.6). As $\mathcal{E}^1_{\mathcal{B}}$ is valid, if $X_j \neq \mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}}$, then $\mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}}$ is a node X_k where either:

a) $X_k \in \mathbf{Pred}_{X_j}^{\mathcal{E}^1_{\mathcal{B}}}$: then $X_j \notin \mathbf{Cls}_{\pi_1}(X_k)$ given that $X_j \in \mathbf{Desc}_{X_k}^{\mathcal{E}^1_{\mathcal{B}}}$. Assuming that $\mathbf{Cls}_{\pi_1}(X_i) = \mathbf{Cls}_{\pi_2}(X_i)$, $X_j \in \mathbf{Cls}_{\pi_2}(X_k)$ given that $X_j \in \mathbf{Cls}_{\pi_2}(X_i) = \mathbf{Cls}_{\pi_1}(X_i)$ (X_j is the parent of X_i in $\mathcal{E}^2_{\mathcal{B}}$) and that X_k is the parent of X_i in $\mathcal{E}^2_{\mathcal{B}}$. Hence, $\mathbf{Cls}_{\pi_1}(X_k) \neq \mathbf{Cls}_{\pi_2}(X_k)$.

b) $X_k \notin \mathbf{Pred}_{X_j}^{\mathcal{E}^1_{\mathcal{B}}}$: then $X_k \notin \mathbf{Cls}_{\pi_1}(X_i)$ given that $X_i \notin \mathbf{Pred}_{X_k}^{\mathcal{E}^1_{\mathcal{B}}}$, and $X_k \in \mathbf{Cls}_{\pi_2}(X_i)$ given that $\mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}} = X_k$. Thus, $\mathbf{Cls}_{\pi_1}(X_i) \neq \mathbf{Cls}_{\pi_2}(X_i)$.

This means that if $\mathcal{E}^1_{\mathcal{B}}$ is valid, then the following condition $\mathrm{Pa}_{X_i}^{\mathcal{E}^1_{\mathcal{B}}} = \mathrm{Pa}_{X_i}^{\mathcal{E}^2_{\mathcal{B}}}$ holds. □

Additionally, valid ETs can be easily transformed into JTs. Let $C_1, \ldots, C_n$ be the cluster sequence induced by VE in a BN $\mathcal{B}$ with an EO π. The maximal clusters (clusters that are not contained in other clusters) in $C_1, \ldots, C_n$ can be connected to form a JT with the same width as π [Darwiche, 2009]. $C_1, \ldots, C_n$ are also the clusters of the inner nodes of the valid ET$\mathcal{E}_{\mathcal{B}}$ that represents π for $\mathcal{B}$. This means that the maximal clusters of an ET can be connected to create a JT of the same width.

3.1.2 Inference complexity in elimination trees

Given the above definitions it is simple to analyze inference complexity in ETs, which we will later use to define the proposed algorithm for learning bounded treewidth BNs. Inference by VE is exponential in the width of the chosen EO. Analogously, inference in an ET $\mathcal{E}_{\mathcal{B}}$ is exponential in the width of $\mathcal{E}_{\mathcal{B}}$.

Definition 3.7. ***(ET width)*** *The width of an ET $\mathcal{E}_{\mathcal{B}}$ is the length of its largest cluster minus one.*

Given a BN $\mathcal{B}$, the width of the ET $\mathcal{E}_{\mathcal{B}}$ with lowest width that is valid for $\mathcal{B}$ is the treewidth of $\mathcal{B}$. Therefore, if $\mathcal{E}_{\mathcal{B}}$ is good enough (near-minimum width), its width is an indicator of the inference complexity of the model as it should be close to the treewidth of $\mathcal{B}$.

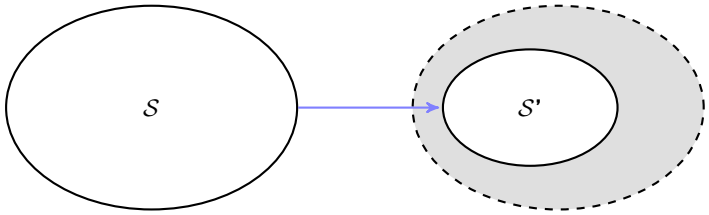


Figure 3.7: A set of EOs $\mathcal{S}$ that are equivalent for a BN $\mathcal{B}$, and a subset $\mathcal{S}'$ of $\mathcal{S}$ that are equivalent for the BN $\mathcal{B}'$, output after adding an arc in $\mathcal{B}$.

3.2 Learning elimination trees

3.2.1 Compiling changes

A naive solution to learn bounded treewidth BNs would be to use a known heuristic to output a good EO (see Subsection 2.2.2). The width of the chosen EO is an estimate of the treewidth of the BN candidates. However, it is computationally demanding to search for good EOs from scratch, and it can be intractable if we have to perform this process for each candidate during the structure search. The results shown in Section 3.1 can be applied to learn tractable BNs in the combined space of DAGs and EOs. Our proposal is to limit the treewidth of the BN by bounding the width of the ET (Definition 3.7). This strategy requires obtaining a valid ET for each network candidate during the learning process.

In this section, we propose methods to compile incrementally in ETs the arc additions and removals made to a BN during the learning process, and show that the proposed algorithms always output valid ETs. As the reversal of arc $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{B}$ can be seen as the removal of arc $X_{\text{out}} \to X_{\text{in}}$ followed by the addition of the reversed arc $X_{\text{in}} \to X_{\text{out}}$, we assume that both changes are compiled each time a reversal is made to a BN.

3.2.1.1 Arc addition

The addition of an arc $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{B}$ may compromise the soundness of an ET $\mathcal{E}_{\mathcal{B}}$. If $\mathcal{E}_{\mathcal{B}}$ is valid, it represents a complete set of equivalent EOs $\mathcal{S}$ (Proposition 3.3). The addition of $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{B}$ places a new restriction on the equivalence of the EOs in $\mathcal{S}$. After applying this local change, there is at least one factor over both X_{out} and X_{in}. Therefore, an ET $\mathcal{E}'_{\mathcal{B}'}$ can only be valid for the new BN $\mathcal{B}'$ if it encodes an ancestral relationship between X_{out} and X_{in}. Algorithm 3.2 modifies the structure of $\mathcal{E}_{\mathcal{B}}$ to meet the new restrictions. The resulting ET $\mathcal{E}'_{\mathcal{B}'}$ represents a complete subset of EOs $\mathcal{S}' \subseteq \mathcal{S}$ (see Figure 3.7) that are also equivalent in $\mathcal{B}'$.

Algorithm 3.2 receives a valid ET $\mathcal{E}_{\mathcal{B}}$, and arc addition $X_{\text{out}} \to X_{\text{in}}$. In $\mathcal{B}'$, variable X_{out} is added to the domain of $\phi_{X_{\text{in}}}$ (line 2). The clusters of the nodes that are predecessors of $\phi_{X_{\text{in}}}$ and descendants of X_{out} contain X_{out} after applying this change. There are three different scenarios that require performing different changes in the ET to ensure its validity.

If X_{out} is a predecessor of $\phi_{X_{\text{in}}}$ in $\mathcal{E}'_{\mathcal{B}'}$, it is not necessary to make any changes in $\mathcal{E}'_{\mathcal{B}'}$. Otherwise, some nodes contain X_{out} in their clusters but not in their predecessors, and $\mathcal{E}'_{\mathcal{B}'}$

Input: Valid ET $\mathcal{E}_{\mathcal{B}}$, output node X_{out}, input node X_{in}
Output: Valid ET $\mathcal{E}'_{\mathcal{B}'}$

let $\mathcal{E}'_{\mathcal{B}'}$ be a copy of $\mathcal{E}_{\mathcal{B}}$;
$\mathbf{Dom}(\phi_{X_{\text{in}}}) \leftarrow \mathbf{Dom}(\phi_{X_{\text{in}}}) \cup \{X_{\text{out}}\}$;
$X_f \leftarrow \text{Pa}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}$;
if $X_f \in \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$ **then**
  $\text{Pa}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \leftarrow X_{\text{out}}$;
else if $X_{\text{out}} \notin \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}$ **then**
  let X_m be the deepest node in $\mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{X_{\text{out}}} \cap \mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}}$;
  let $\mathcal{E}^1_{\mathcal{B}'}$ and $\mathcal{E}^2_{\mathcal{B}'}$ be two copies of $\mathcal{E}'_{\mathcal{B}'}$;
  $X_k \leftarrow \mathbf{Ch}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_m} \cap \mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}}$;
  $\text{Pa}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_k} \leftarrow X_{\text{out}}$;
  $X_h \leftarrow \mathbf{Ch}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_m} \cap \mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_{\text{out}}}$;
  $\text{Pa}^{\mathcal{E}^2_{\mathcal{B}}}_{X_h} \leftarrow X_f$;
  $\text{Pa}^{\mathcal{E}^2_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \leftarrow X_{\text{out}}$;
  if $\text{width}(\mathcal{E}^1_{\mathcal{B}'}) < \text{width}(\mathcal{E}^2_{\mathcal{B}'})$ **then**
    $\mathcal{E}'_{\mathcal{B}'} \leftarrow \mathcal{E}^1_{\mathcal{B}'}$;
  else
    $\mathcal{E}'_{\mathcal{B}'} \leftarrow \mathcal{E}^2_{\mathcal{B}'}$;
  end
end
return $\mathcal{E}'_{\mathcal{B}'}$;

Algorithm 3.2: Compilation of the addition of arc $X_{\text{out}} \rightarrow X_{\text{in}}$ ($\text{add}(\mathcal{E}_{\mathcal{B}}, X_{\text{out}}, X_{\text{in}})$).

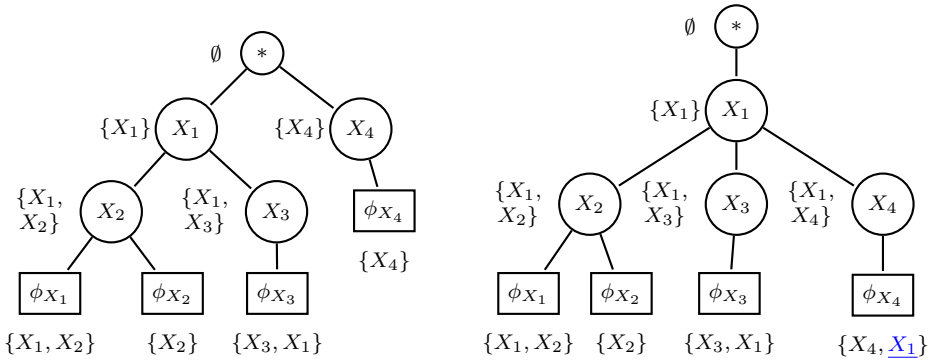


Figure 3.8: (a) an ET $\mathcal{E}_{\mathcal{B}}$, and (b) the ET $\mathcal{E}'_{\mathcal{B}'}$ output after incrementally compiling the arc addition $X_1 \rightarrow X_4$ (i.e., addition of X_1 to $\mathbf{Dom}(\phi_{X_4})$) in $\mathcal{E}_{\mathcal{B}}$.

is not sound. If $X_f = \mathrm{Pa}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}$ is a predecessor of X_{out} in $\mathcal{E}_{\mathcal{B}}$ (line 4), X_{out} is set as the new parent of $\phi_{X_{\text{in}}}$ in $\mathcal{E}'_{\mathcal{B}'}$ (line 5). Thus, X_{out} is a predecessor of $\phi_{X_{\text{in}}}$ in $\mathcal{E}'_{\mathcal{B}'}$.

If X_{out} is not a predecessor of $\phi_{X_{\text{in}}}$ in $\mathcal{E}_{\mathcal{B}}$ and X_f is not a predecessor of X_{out} in $\mathcal{E}_{\mathcal{B}}$ (line 6), the cluster of the node in $\{X_{\text{out}}\} \cup \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$ contains X_{out} but the clusters of their predecessors in $\mathcal{E}_{\mathcal{B}}$ do not. Algorithm 3.2 creates two candidate ETs ($\mathcal{E}^1_{\mathcal{B}'}$ and $\mathcal{E}^2_{\mathcal{B}'}$). The first is output by setting X_{out} as the parent of X_k in $\mathcal{E}^1_{\mathcal{B}'}$ (line 10), that is, the shallowest predecessor of $\phi_{X_{\text{in}}}$ in $\mathcal{E}_{\mathcal{B}}$ that does not belong to $\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$ (line 9). The second is output by setting $\mathrm{Pa}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}$ as the parent of X_h in $\mathcal{E}^2_{\mathcal{B}'}$ (line 12), that is, the shallowest predecessor of X_{out} in $\mathcal{E}_{\mathcal{B}}$ that does not belong to $\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}$ (line 11), and setting X_{out} as the new parent of $\phi_{X_{\text{in}}}$ in $\mathcal{E}^2_{\mathcal{B}'}$ (line 13). $\mathcal{E}'_{\mathcal{B}'}$ is selected as the ET of smaller width between $\mathcal{E}^1_{\mathcal{B}'}$ and $\mathcal{E}^2_{\mathcal{B}'}$ (lines 14–18). Either way, X_{out} is a predecessor in $\mathcal{E}'_{\mathcal{B}'}$ of the nodes in $\{X_{\text{out}}\} \cup \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$, and the returned ET $\mathcal{E}'_{\mathcal{B}'}$ is valid (Lemma 3.1).

Lemma 3.1. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET that represents $\mathcal{B}$ over $\mathcal{X}$. Given $X_{out}, X_{in} \in \mathcal{X}$, the ET $\mathcal{E}'_{\mathcal{B}'}$ yielded after applying add$(\mathcal{E}_{\mathcal{B}}, X_{out}, X_{in})$ in Algorithm 3.2, is also a valid ET representing $\mathcal{B}'$ over $\mathcal{X}$.*

Proof. See Appendix A.1. □

This process can be performed efficiently in ETs of bounded width (see Theorem 3.2).

Figure 3.8 shows an example of the incremental compilation of the addition of arc $X_1 \rightarrow X_4$ in $\mathcal{X}_2$. $\mathcal{E}_{\mathcal{B}}$ represents all the permutations of $\{X_1, X_2, X_3, X_4\}$ where X_2 and X_3 are eliminated before X_1. After adding $X_1 \rightarrow X_4$ in $\mathcal{B}$, there are EOs represented by $\mathcal{E}_{\mathcal{B}}$ that are not equivalent for $\mathcal{B}'$ (e.g., $\{X_2, X_3, X_1, X_4\}$ and $\{X_2, X_3, X_4, X_1\}$). Using Algorithm 3.2, $X_{\text{out}} = X_1$ and $X_{\text{in}} = X_4$. As $X_f = X_4$ is not a predecessor of X_1 in $\mathcal{E}_{\mathcal{B}}$ (line 4), and X_1 is not a predecessor of ϕ_{X_4}(line 6), two ETs $\mathcal{E}^1_{\mathcal{B}'}$ and $\mathcal{E}^2_{\mathcal{B}'}$ are created. Assuming that $\mathcal{E}^1_{\mathcal{B}'}$ is smaller than $\mathcal{E}^2_{\mathcal{B}'}$, X_1 is the new parent of X_4 in $\mathcal{E}'_{\mathcal{B}'}$.

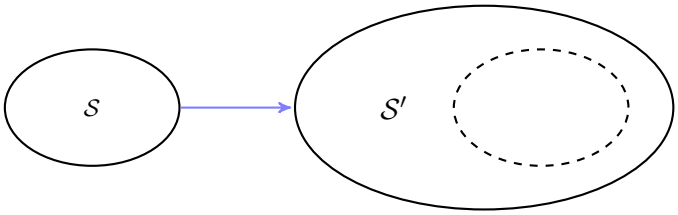


Figure 3.9: A set of EOs $\mathcal{S}$ that are equivalent for a BN $\mathcal{B}$, and a superset $\mathcal{S}'$ of $\mathcal{S}$ that are equivalent for the BN $\mathcal{B}'$, yielded after removing an arc in $\mathcal{B}$.

3.2.1.2 Arc removal

The removal of an arc $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{B}$ may compromise the completeness of an ET $\mathcal{E}_{\mathcal{B}}$. Let $\mathcal{S}$ be the set of EOs represented by $\mathcal{E}_{\mathcal{B}}$. The removal of $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{B}$ leads to a reduction in the restrictions on EO equivalence in $\mathcal{S}$. This means that $\mathcal{E}_{\mathcal{B}}$ may not represent all the EOs that are equivalent to EOs in $\mathcal{S}$. Algorithm 3.3 yields an ET that represents a complete superset of EOs $\mathcal{S}' \supseteq \mathcal{S}$ (see Figure 3.9) containing all the EOs that are equivalent to EOs in $\mathcal{S}$ for $\mathcal{B}'$, which is the resulting BN after removing arc $X_{\text{out}} \to X_{\text{in}}$ from $\mathcal{B}$.

Input: Valid ET $\mathcal{E}_{\mathcal{B}}$, output node X_{out}, input node X_{in}
Output: Valid ET $\mathcal{E}'_{\mathcal{B}'}$
1 let $\mathcal{E}'_{\mathcal{B}'}$ be a copy of $\mathcal{E}_{\mathcal{B}}$;
2 $\mathbf{Dom}(\phi(X_{\text{in}})) \leftarrow \mathbf{Dom}(\phi(X_{\text{in}})) \setminus \{X_{\text{out}}\}$;
3 $X_j \leftarrow X_{\text{out}}$;
4 let X_i be the shallowest node in $(\mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cup \{\phi_{X_{\text{in}}}\}) \cap \mathbf{Desc}^{\mathcal{E}'_{\mathcal{B}'}}_{X_{\text{out}}}$;
5 **while** $X_j \notin \mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i)$ **and** $X_j \neq *$ **do**
6 let X'_j be the deepest node in $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i) \setminus \{X_i\}$ if $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i) \setminus \{X_i\} \neq \emptyset$ and $*$ otherwise ;
7 $\text{Pa}^{\mathcal{E}'_{\mathcal{B}'}}_{X_i} \leftarrow X'_j$;
8 set X_i as the shallowest node in $\{X_j\} \cup \mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{X_j} \cap \mathbf{Desc}^{\mathcal{E}'_{\mathcal{B}'}}_{X'_j}$;
9 $X_j \leftarrow X'_j$;
10 **end**
11 **return** $\mathcal{E}'_{\mathcal{B}'}$;

Algorithm 3.3: Compilation of the removal of arc $X_{\text{out}} \to X_{\text{in}}$ (remove$(\mathcal{E}_{\mathcal{B}}, X_{\text{out}}, X_{\text{in}})$).

After removing $X_{\text{out}} \to X_{\text{in}}$ from $\mathcal{B}$, the shallowest node in $(\mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cup \{\phi_{X_{\text{in}}}\}) \cap \mathbf{Desc}^{\mathcal{E}'_{\mathcal{B}'}}_{X_{\text{out}}}$, which we refer to as X_i (line 4), may not be complete. If X_i is not complete (line 5), Algorithm 3.3 sets the deepest node in $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i)$, which we refer to as X'_j, as its new parent (lines 6–7). Note that the idea behind this change is that the new parent of X_i is in its cluster, making X_i complete. After this change, the shallowest node in $\{X_j\} \cup \mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{X_j} \cap \mathbf{Desc}^{\mathcal{E}'_{\mathcal{B}'}}_{X'_j}$ may not be complete. Thus, Algorithm 3.3 repeats the same process until every node is complete, guaranteeing the validity of every node in $\mathcal{E}'_{\mathcal{B}'}$ (Lemma 3.2).

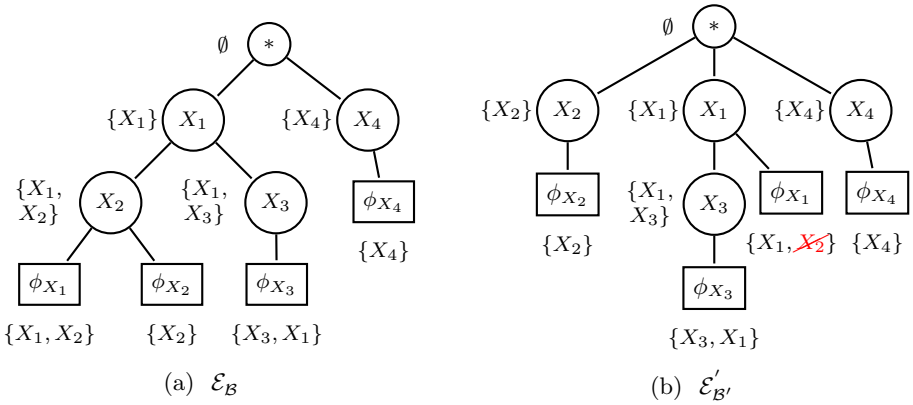


Figure 3.10: (a) an ET $\mathcal{E}_{\mathcal{B}}$, and (b) an ET $\mathcal{E}'_{\mathcal{B}'}$ yielded after incrementally compiling the removal of arc $X_2 \to X_1$ (i.e., removal of X_2 from $\mathbf{Dom}(\phi_{X_1})$) in $\mathcal{E}_{\mathcal{B}}$.

Lemma 3.2. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET that represents $\mathcal{B}$ over $\mathcal{X}$. Given $X_{out}, X_{in} \in \mathcal{X}$, the ET $\mathcal{E}'_{\mathcal{B}'}$ that represents $\mathcal{B}'$ output after applying remove$(\mathcal{E}_{\mathcal{B}}, X_{out}, X_{in})$ in Algorithm 3.3, is also valid.*

Proof. See Appendix A.1. □

Figure 3.10 shows an example of how Algorithm 3.3 compiles, in an ET $\mathcal{E}_{\mathcal{B}}$, the removal of arc $X_2 \to X_1$ in $\mathcal{B}$. Let $\mathcal{E}'_{\mathcal{B}'}$ be a copy of $\mathcal{E}_{\mathcal{B}}$ where X_2 has been removed from the domain of ϕ_{X_1}. First, X_j is set to X_{out} (line 3). As $X_{\text{out}} = X_2$ and $X_{\text{in}} = X_1$, X_i is the shallowest node in $\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_1}} \cup \{X_1\}$ that is a descendant of X_2 (line 4), namely ϕ_{X_1}.

X_{out} is not in the cluster of ϕ_{X_1} in $\mathcal{E}'_{\mathcal{B}'}$ (line 5). Hence, X'_j is set to X_1, that is, the deepest node in the cluster of ϕ_{X_1} (line 6). X_1 is now set as the parent of ϕ_{X_1} in $\mathcal{E}'_{\mathcal{B}'}$, which makes node ϕ_{X_1} complete. The new X_i is set to X_2 (line 8), and the new X_j is X_1 (line 9).

As X_1 is not in the cluster of X_2 in $\mathcal{E}_{\mathcal{B}}$, the same process is applied again (lines 6–9). In this case, the root node $*$ is set as the parent of X_2 in $\mathcal{E}'_{\mathcal{B}'}$ and the new X_j is $*$, ending the loop (line 5) and returning $\mathcal{E}'_{\mathcal{B}'}$.

The complexity of this process is polynomial in the number of variables and the width of the ET (see Theorem 3.2).

3.2.2 Optimization

The methods described above adapt an ET $\mathcal{E}_{\mathcal{B}}$ (resulting in $\mathcal{E}'_{\mathcal{B}'}$) to a local change in a BN $\mathcal{B}$ (resulting in $\mathcal{B}'$). The objective of these methods is to make $\mathcal{E}_{\mathcal{B}}$ valid for the new BN. In addition to the incremental compilation of ETs, we also propose a strategy to search in the space of EOs given a BN $\mathcal{B}$. The purpose of this procedure is to reduce the width of an ET without modifying $\mathcal{B}$. We use a simple and efficient heuristic to address this problem. In this paper, we use the optimization process to refine (reduce the width) of the ETs returned by Algorithms 3.2 and 3.3.

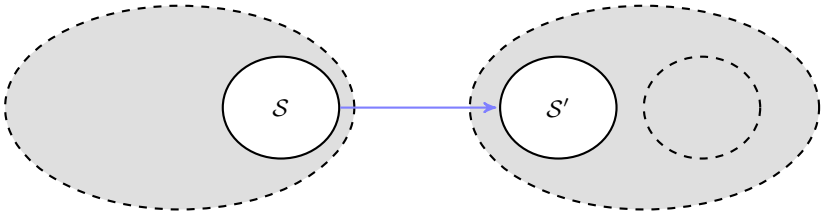


Figure 3.11: The EOs in $\mathcal{S}$ are equivalent for a BN $\mathcal{B}$, and a swap produces other set of EOs $\mathcal{S}'$, with $\mathcal{S} \cap \mathcal{S}' = \emptyset$, where the EOs in $\mathcal{S}'$ are also equivalent for $\mathcal{B}$.

Algorithm 3.4 swaps the position in $\mathcal{E}_\mathcal{B}$ of node X_i with the position of its parent X_p, also changing the parents of any children of X_i in $\mathcal{E}_\mathcal{B}$ whose validity is compromised by the swap. Algorithm 3.4 guarantees the validity of the resulting ETs. Note that after each swap only the clusters of X_i and X_p may change.

The sets of EOs represented by $\mathcal{E}_\mathcal{B}$ and the new ET $\mathcal{E}'_\mathcal{B}$, which we refer to as $\mathcal{S}$ and $\mathcal{S}'$ respectively, are disjoint (see Figure 3.11). In $\mathcal{E}_\mathcal{B}$, $(X_p < X_i)_\pi$ for any EO $\pi \in \mathcal{S}$, whereas $(X_i < X_p)_{\pi'}$ in $\mathcal{E}'_\mathcal{B}$ for any EO $\pi' \in \mathcal{S}'$.

Algorithm 3.4 proceeds as follows: First, a copy $\mathcal{E}'_\mathcal{B}$ of $\mathcal{E}_\mathcal{B}$ is created (line 1), and X_j is set to the parent of X_i in $\mathcal{E}_\mathcal{B}$ (line 2). Then, the positions of X_i and X_j are swapped in $\mathcal{E}'_\mathcal{B}$ (lines 3 and 4). After that, the children of X_i whose cluster contained X_j set their parent in $\mathcal{E}'_\mathcal{B}$ to X_j (lines 5–9), because otherwise these nodes would not be sound.

Input: Valid ET $\mathcal{E}_\mathcal{B}$, node X_i
Output: Valid ET $\mathcal{E}'_\mathcal{B}$
1 Let $\mathcal{E}'_\mathcal{B}$ be a copy of $\mathcal{E}_\mathcal{B}$;
2 $X_j \leftarrow \mathrm{Pa}_{X_i}^{\mathcal{E}'_\mathcal{B}}$;
3 $\mathrm{Pa}_{X_i}^{\mathcal{E}'_\mathcal{B}} \leftarrow \mathrm{Pa}_{X_j}^{\mathcal{E}'_\mathcal{B}}$;
4 $\mathrm{Pa}_{X_j}^{\mathcal{E}'_\mathcal{B}} \leftarrow X_i$;
5 **for** $X_k \in \mathbf{Ch}_{X_i}^{\mathcal{E}'_\mathcal{B}}$ **do**
6 **if** $X_j \in \mathbf{Cls}_{\mathcal{E}'_\mathcal{B}}(X_k)$ **then**
7 $\mathrm{Pa}_{X_k}^{\mathcal{E}'_\mathcal{B}} \leftarrow X_j$;
8 **end**
9 **end**
10 **return** $\mathcal{E}'_\mathcal{B}$;

Algorithm 3.4: Swap of X_i and $\mathrm{Pa}_{X_i}^{\mathcal{E}_\mathcal{B}}$ in ET $\mathcal{E}_\mathcal{B}$ (swap$(\mathcal{E}_\mathcal{B}, X_i)$).

Lemma 3.3. *Let $\mathcal{E}_\mathcal{B}$ be a valid ET that represents $\mathcal{B}$ over $\mathcal{X}$. Given $X_i \in \mathcal{X}$, the ET $\mathcal{E}'_\mathcal{B}$ representing $\mathcal{B}$ yielded after applying swap$(\mathcal{E}_\mathcal{B}, X_i)$ in Algorithm 3.4, is also valid.*

Proof. See Appendix A.1. □

Figure 3.12 shows the result of applying Algorithm 3.4 to an ET $\mathcal{E}_\mathcal{B}$. In this example,

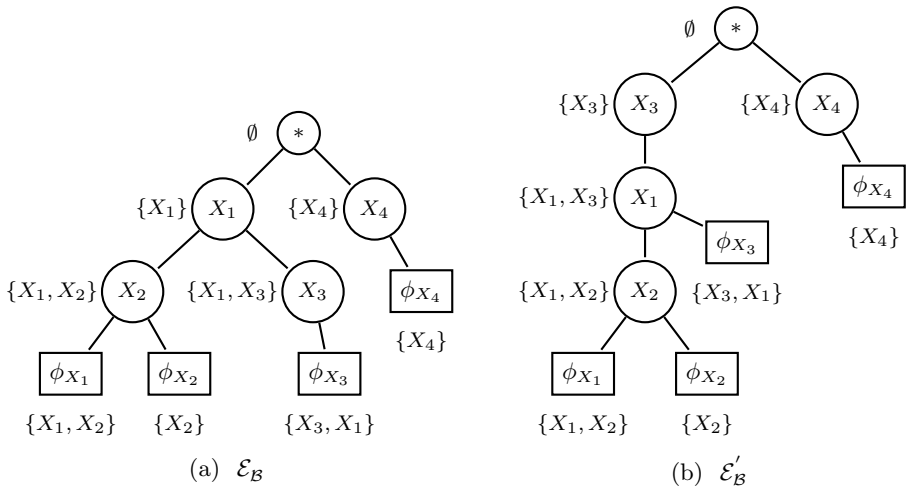


Figure 3.12: (a) an ET $\mathcal{E}_{\mathcal{B}}$, and (b) an ET $\mathcal{E}'_{\mathcal{B}}$ yielded after swapping the positions of X_3 and X_1 in $\mathcal{E}_{\mathcal{B}}$.

the positions of X_3 and X_1 (parent of X_3 in $\mathcal{E}_{\mathcal{B}}$) in the resulting ET $\mathcal{E}_{\mathcal{B}'}$ are swapped. $\mathcal{E}_{\mathcal{B}}$ represents the EOs π of $\{X_1, X_2, X_3, X_4\}$ where $(X_2 < X_1)_\pi$ and $(X_3 < X_1)_\pi$ (e.g., $(X_2, X_3, X_1, X_4), (X_4, X_3, X_2, X_1), \dots$), while $\mathcal{E}_{\mathcal{B}'}$ represents the EOs π' of $\{X_1, X_2, X_3, X_4\}$ where $(X_2 < X_1)_{\pi'}$ and $(X_1 < X_3)_{\pi'}$ (e.g., $(X_4, X_2, X_1, X_3), (X_2, X_4, X_1, X_3), \dots$).

We use a greedy heuristic, which, given an ET $\mathcal{E}_{\mathcal{B}}$ and a set of nodes for optimization ($\mathcal{X}_{\text{opt}}$), visits each node $X_i \in \mathcal{X}_{\text{opt}}$ from the shallowest to the deepest, checking at each step whether swapping the position of X_i and its parent reduces the width of the ET, see Algorithm 3.5.

Orderings that are good for one BN $\mathcal{B}^0$ (i.e., their width is close to the treewidth of $\mathcal{B}^0$) may not be good for the BN $\mathcal{B}$ yielded after applying a local change in $\mathcal{B}^0$. We perform the optimization process after the compilation of each local change. For efficiency, we select the nodes that may have a different cluster after compiling the local change to initialize $\mathcal{X}_{\text{opt}}$, given that they are more likely to produce relevant changes in the width of the ET. Next, we show the set of nodes selected for optimization ($\mathcal{X}_{\text{opt}}$) after compiling arc additions and removals. Each bullet point describes the assignment to $\mathcal{X}_{\text{opt}}$ at a possible scenario. We also explain the reason why the cluster of any of the nodes in $\mathcal{X}_{\text{opt}}$ may have changed.

- Addition of arc $X_{\text{out}} \rightarrow X_{\text{in}}$:
  - If $X_{\text{out}} \in \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}$, $\mathcal{X}_{\text{opt}} = \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$:
    The width of the clusters in $\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$ grows, given that they now contain X_{out}.
  - Else, if $\text{Pa}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \in \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$, $\mathcal{X}_{\text{opt}} = (\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}} \cup \{X_{\text{out}}\}) \cap \mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{\text{Pa}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}}$:
    The width of the clusters in $(\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}} \cup \{X_{\text{out}}\}) \cap \mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{\text{Pa}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}}$ may grow, given

Input: Valid ET $\mathcal{E}_\mathcal{B}$, set of nodes $\mathcal{X}_{\text{opt}}$
Output: Valid ET $\mathcal{E}'_\mathcal{B}$
Let $\mathcal{E}'_\mathcal{B}$ be a copy of $\mathcal{E}_\mathcal{B}$;
Let $\boldsymbol{X}_{\text{opt}}$ be a list that contains the nodes in $\mathcal{X}_{\text{opt}}$ ordered from the shallowest to the deepest;
for $X_i \in \boldsymbol{X}_{\text{opt}}$ **do**
  flag $\leftarrow$ **true**;
  while flag = **true do**
    $\mathcal{E}^1_\mathcal{B} \leftarrow \text{swap}(\mathcal{E}'_\mathcal{B}, X_i)$;
    if $width(\mathcal{E}^1_\mathcal{B}) \leq width(\mathcal{E}'_\mathcal{B})$ **then**
      $\mathcal{E}'_\mathcal{B} \leftarrow \mathcal{E}^1_\mathcal{B}$;
    else
      flag $\leftarrow$ **false**;
    end
  end
end
return $\mathcal{E}'_\mathcal{B}$;

Algorithm 3.5: Optimization of an ET (optimize($\mathcal{E}_\mathcal{B}, \mathcal{X}_{\text{opt}}$)).

that they now contain $\mathbf{Dom}(\phi_{X_{\text{in}}})$.

– Otherwise, $\mathcal{X}_{\text{opt}} = (\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \cup \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}} \cup \{X_{\text{out}}\}) \setminus (\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}})$:
Node $\phi_{X_{\text{in}}}$ is set as a descendant of X_{out}, and the nodes in $\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \setminus (\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}})$ are either predecessors or descendants of X_{out} and $\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}} \setminus (\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}})$ in the new ET.

- Removal of arc $X_{\text{out}} \rightarrow X_{\text{in}}$, $\mathcal{X}_{\text{opt}} = \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}_\mathcal{B}}_{X_j}$:
  Let X_j be the last node that had a new child X_i assigned by Algorithm 3.3. The nodes in $\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}_\mathcal{B}}_{X_j}$ may have smaller clusters in the new ET.

The optimization of an ET takes polynomial time in the number of variables and in the width of the ET (see Theorem 3.2).

3.2.3 Learning elimination trees from data

Using the incremental compilation and optimization methods described above, it is rather straightforward to learn ETs from a dataset $\mathcal{D}$ in combination with any score+search BN learning method that applies local changes during the search. Learning low inference complexity BNs with this approach is also easily derived. It can be achieved by bounding the width of each ET during the learning process (which bounds the treewidth of their corresponding BNs).

Theorem 3.1 ensures that any algorithm that uses the above strategy will always produce valid ETs.

Theorem 3.1. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET over $\mathcal{X}$, and $\mathcal{E}'_{\mathcal{B}'}$ the result of incrementally compiling on $\mathcal{E}_{\mathcal{B}}$ any local change in $\mathcal{B}$ using Algorithms 3.2 and 3.3 and optimizing the resulting ET using Algorithm 3.5. Then $\mathcal{E}'_{\mathcal{B}'}$ is a valid ET.*

Proof. See Appendix A.1. □

As we apply the incremental compilation and optimization methods to each candidate during the learning process, efficiency is a critical issue. Theorem 3.2 bounds the computational time complexity of the incremental compilation and optimization methods proposed above.

Theorem 3.2. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET over a set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. The process described in Theorem 3.1 to output $\mathcal{E}'_{\mathcal{B}'}$ can be performed in time $O(n^2 \cdot width(\mathcal{E}_{\mathcal{B}}))$.*

Proof. See Appendix A.2. □

Let $\mathcal{A}$ be any algorithm that learns the structure of a BN using only local changes in the structure of the network during the learning process. $\mathcal{A}$ can be adapted to learn low inference complexity BNs compiling and optimizing (Algorithms 3.2–3.5) all the local changes that are applied to the BN $\mathcal{B}$ to its respective ET $\mathcal{E}_{\mathcal{B}}$. Thus, $\mathcal{E}_{\mathcal{B}}$ can be used to bound the treewidth of $\mathcal{B}$ using Definition 3.7. Note that the input for the adaptation of $\mathcal{A}$ should be an ET $\mathcal{E}^0_{\mathcal{B}0}$ valid for the initial BN $\mathcal{B}^0$.

3.3 Experimental results

In this section, we empirically analyzed the performance of the proposed framework in terms of fitting and computational complexity. Although our approach could be used with most score+search BN learning methods, in the experiments we combine the incremental compilation and optimization methods proposed in Section 3.2 with a greedy hill-climbing for the structure search. We call the resulting method hc-ET. We compared hc-ET with k-greedy and k-MAX to highlight the advantages and drawbacks of using our approach to learn bounded treewidth BNs. We also tested a polynomial version of hc-ET that only considers arc additions during the structure search. We call this method hc-ET-poly.

To perform the experiments, we used 22 real-world datasets. These datasets were previously used in several papers [Lowd and Davis, 2010; Van Haaren and Davis, 2012; Bekker et al., 2015; Scanagatta et al., 2018b], and can be found at `https://github.com/UCLA-StarAI/Density-Estimation-Datasets`. Additionally, we generated synthetic data from 12 real-world BNs. These BNs were obtained from the bnlearn BN repository `http://www.bnlearn.com/bnrepository/`, and are cited therein. Table 3.1 briefly describes the basic properties of each dataset.

For each dataset we learned three BNs with each of the compared methods, using different treewidth bounds (3, 5 and 7). In all cases, the score function to maximize was BIC. k-greedy and k-MAX require to fix a maximum execution time, which we set to n seconds (i.e., a

Table 3.1: Basic properties of the datasets: Number of variables (N. vars), training instances (N. train inst.) and test instances (N. test inst.). Although N. test inst. does not apply to these experiments, the test data from the real-world datasets is used in Chapter 4.

(a) Real-world datasets

Dataset	N. vars	N. train inst.	N. test inst.
NLTCS	16	16,181	3,236
MSNBC1	17	291,326	58,265
KDDCup	65	180,092	34,955
Plants	69	17,412	3,482
Audio	100	15,000	3,000
Jester	100	9,000	4,116
Netflix	100	15,000	3,000
Accidents	111	12,758	2,551
Retail	135	22,041	4,408
Pumsb-star	163	12,262	2,452
DNA	180	1,600	1,186
Kosarek	190	33,375	6,675
MSWeb	294	29,441	5,000
Book	500	8,700	1,739
EachMovie	500	4,525	591
WebKB	839	2,803	838
Reuters-521	889	6,532	1,540
20 NewsGroup	910	11,293	3,764
Movie reviews	1,001	1,600	250
BBC	1,058	1,670	330
Voting	1,359	1,214	350
Ad	1,556	2,461	491

(b) Synthetic datasets

Dataset	N. vars	N. train inst.
Hailfinder	56	5,000
Hepar II	70	5,000
Win95pts	76	5,000
Pathfinder	135	5,000
Munin1	186	5,000
Andes	223	5,000
Diabetes	413	5,000
Pigs	413	5,000
Link	724	5,000
Munin2	1,003	5,000
Munin3	1,041	5,000
Munin4	1,038	5,000

Table 3.2: Comparison of the methods in all the datasets, using a treewidth bound of 3. The optimal results are denoted in boldface.

	hc-ET	hc-ET-poly	k-greedy	k-MAX
mean rank BIC	**1.32±0.53**	1.79±0.54	3.79±0.41	3.09±0.62
mean rank LL	**1.38±0.55**	1.68±0.47	3.79±0.41	3.15±0.56
mean rank time	1.88±0.33	**1.12±0.33**	3.56±0.5	3.44±0.5
mean treewidth	3±0	3±0	2.85±0.66	2.68±0.77

Table 3.3: Comparison of the methods in all the datasets, using a treewidth bound of 5. The optimal results are denoted in boldface.

	hc-ET	hc-ET-poly	k-greedy	k-MAX
mean rank BIC	**1.35±0.69**	1.91±0.57	3.76±0.61	2.97±0.63
mean rank LL	**1.47±0.71**	1.71±0.52	3.71±0.63	3.12±0.59
mean rank time	1.88±0.33	**1.12±0.33**	3.56±0.5	3.44±0.5
mean treewidth	4.88±0.48	4.91±0.38	3.56±1.13	3.41±1.08

second for each variable) to compute the cache of best parent sets and $n/10$ seconds for the structure search. These values were used by Scanagatta et al. [2018b] in their experiments. To compare the results we used the following performance measures: the BIC score and the log-likelihood (LL) of the models in the training dataset, the learning time, and the treewidth of the returned models.

We analyzed the significance of the differences found for each performance measure in all the datasets and for all the treewidth bounds using the Friedman test with $\alpha = 0.05$ and Holm's [Holm, 1979] and Shaffer's [Shaffer, 1986] post-hoc procedures. Both Holm's and Shaffer's procedures associate pairwise comparisons with a set of hypotheses and perform a step-down process with the corresponding set of ordered p-values to adjust the value of α [Garcia and Herrera, 2008].

Experiments were performed on a computer with an Intel Core i7-6700K CPU at 4.00GHz with 16GB main memory, running Ubuntu 16.04 LTS. hc-ET and hc-ET-poly were written in Python 2.7.12 and C++11 (version 5.4.0), while k-greedy and k-MAX were downloaded from http://ipg.idsia.ch/software/blip and are written in Java.

3.3.1 Comparison with k-MAX

Tables 3.2–3.4 give an overview of the results obtained using the treewidth bounds 3, 5 and 7, respectively. For each performance measure, the mean rank ± the standard deviation of each method over all the datasets is shown. The ranking of the methods is given by their average performance (BIC, LL and time) compared to the rest (i.e., the best is ranked the first and the worst is ranked the fourth). The mean treewidth ± the standard deviation is also shown. The detailed results are supplied at https://github.com/marcobb8/tr_bn/blob/master/Supplementary%20Material/supplemantary-material_ets.pdf.

Figures 3.13–3.15 graphically present the results obtained with Holm's and Shaffer's procedure for each performance measure in all datasets. In the figures, groups of methods that

Table 3.4: Comparison of the methods in all the datasets, using a treewidth bound of 7. The optimal results are denoted in boldface.

	hc-ET	hc-ET-poly	k-greedy	k-MAX
mean rank BIC	**1.35±0.69**	1.97±0.58	3.79±0.59	2.88±0.69
mean rank LL	**1.29±0.46**	1.82±0.52	3.74±0.62	3.15±0.56
mean rank time	1.91±0.29	**1.09±0.29**	3.41±0.5	3.59±0.5
mean treewidth	6.74±0.96	6.74±0.86	3.91±1.31	4±1.44

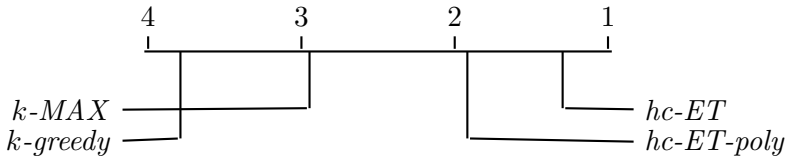


Figure 3.13: Comparison of BIC scores with Holm's and Shaffer's tests.

are not significantly different are connected with a thick horizontal line. We used the graphical representation proposed by Demšar [2006]. Each figure represents both procedures, given that the significant differences observed by Shaffer's procedure are identical to those observed by Holm's procedure.

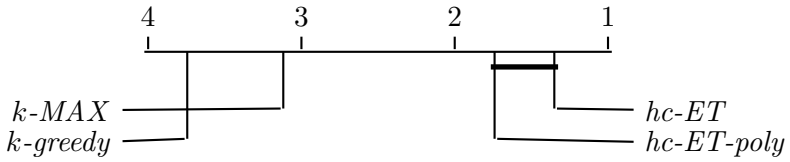


Figure 3.14: Comparison of log-likelihood with Holm's and Shaffer's tests.

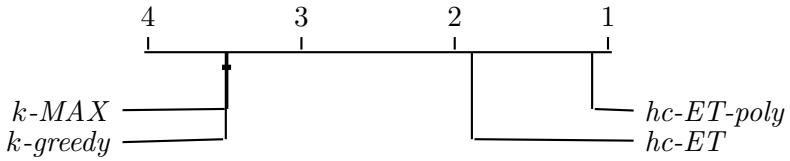


Figure 3.15: Comparison of learning time with Holm's and Shaffer's tests.

Figure 3.13 shows significant differences between the BIC score achieved by all the methods; hc-ET performs the best overall, followed by hc-ET-poly, k-MAX, and k-greedy. Moreover, Tables 3.2–3.4 show that similar results can be found for all the tested treewidth bounds. The detailed results also show that hc-ET performs better than k-MAX and k-greedy in over 94% of the experiments, and performs better than hc-ET-poly in around 80% of the experiments. The treewidth of the models output by each method suggests that one of the reasons why our proposal manages to optimize better the BIC score is that it allows a tighter fitting to the treewidth bound.

The comparison of the log-likelihood in Tables 3.2–3.4 and Figure 3.14 leads us to conclusions that are similar to those drawn for the BIC score. Nevertheless, in this case no

significant differences were found between hc-ET and hc-ET-poly. This suggests that hc-ET-poly requires a higher number of parameters to obtain a similar fitting to hc-ET in terms of log-likelihood.

As shown in Figure 3.15, we observed significant differences between the learning time of all the methods with the exception of k-MAX and k-greedy. The latter result was expected, given that the imposed limit in execution time of both approaches is the same. hc-ET-poly is the fastest method in all cases, followed by hc-ET. However, we must be cautious when interpreting these results. First, these methods are implemented in different programming languages. Second, the bound in execution time set for k-greedy and k-MAX compels their learning time to scale linearly. Therefore, the difference in learning time between our method and k-MAX and k-greedy is clearly higher in the smaller datasets. Finally, although hc-ET takes slightly more time than hc-ET-poly in all the experiments, we think that the improvement in BIC score is worthwhile in most situations.

3.4 Conclusions

Traditional methods for learning BNs usually output models where exact inference is intractable. In this chapter, we provide a novel framework for learning tractable BNs. We defined valid ETs, and proposed compilation methods for adapting valid ETs to any local change that may be applied to a BN (i.e., arc addition, removal, and reversal). We proved that the proposed methods always return valid ETs in polynomial time (Theorems 3.1 and 3.2). Our approach can be easily combined with any score+search BN learning method that uses only local changes in the network during the structure search. Valid ETs can be used to search in the combined space of DAGs and EOs, avoiding redundant solutions (i.e., all the EOs that are equivalent for a BN are represented by the same valid ET). Hence, we used this representation to efficiently bound the inference complexity of each BN during the learning process.

Experimental results showed that our approach places a tight upper bound on the inference complexity of the networks. The models learned with the proposed methods were competitive with other state of the art methods, performing better in terms of BIC score and log-likelihood in most cases.

In the future, we aim to study the relationship between the density of DAGs and the number of equivalent EOs. This would clarify the situations in which it is better to use ETs during the learning process. Another appealing future research line is to study how to learn tractable probabilistic models with large treewidth by taking advange of the local structures or the exchangebility between the variables.

Chapter 4

Tractable learning of Bayesian networks from partially observed data

One of the main advantages of BNs is that, as generative models, they can answer all conditional probability queries involving the variables of the network. In supervised classification, if the value of a certain set of feature variables is missing, a BN can exactly obtain the a posteriori most likely class label given the observed predictors. However, learning BNs from incomplete data continues to be a challenging problem. EM [Dempster et al., 1977; McLachlan and Krishnan, 2008; Liao and Ji, 2009] is the most widely used algorithm for learning a model in the presence of missing values. Friedman's SEM [Friedman, 1997] extends the EM algorithm to simultaneously learn the structure and parameters of a BN from incomplete data. This method has been successfully applied in semi-supervised classification [Hernández-González et al., 2013; Wang et al., 2014] and clustering [Peña et al., 2000; Luengo-Sanchez et al., 2016] problems. One of its most celebrated features is that, by optimizing an expectation of the score, the algorithm guarantees convergence to a local optimum of the objective function with respect to the observed data. Note that under the missing at random assumption, optimizing the scoring function with respect to the observed data is equivalent to optimizing this function in the incomplete dataset. Because of its iterative nature, SEM is known to be computationally a highly demanding algorithm. Moreover, as inference in BNs is NP-hard [Cooper, 1990], the computational cost of the E-step is likely to be prohibitive when the network candidates exhibit high inference complexity. Thus, bounding the inference complexity is critical to ensure the tractability of SEM.

The literature contains several approaches that address the problem of learning BNs with low inference complexity (see Section 2.3.1). Nevertheless, these methods are not capable of learning from incomplete datasets (see Section 2.3.2). More recently, Scanagatta et al. [2018b] proposed SEM-kMAX algorithm for learning BNs with bounded treewidth from incomplete datasets, introducing the k-MAX algorithm in the M-step of SEM. The main drawback of

this approach is that unlike Friedman's SEM, it does not provide convergence guarantees on the objective function with respect to the observed data.

In this chapter, we propose an efficient method (i.e., with polynomial cost in the number of variables and number of instances of the dataset) for learning BNs with low inference complexity in the presence of missing values. For this purpose, we provide a rapid heuristic to estimate the upper bounds in the complexity of the models. Furthermore, we develop an efficient strategy to directly optimize the score with respect to the observed data and discuss its benefits compared to optimizing an expectation of the score (e.g., Friedman's SEM). In the experiments, the proposed approach demonstrates promising results in terms of model fitting and imputation accuracy.

The remainder of this chapter is organized as follows: Section 4.1 presents our proposal and provides theoretical results on the complexity of the algorithm. Section 4.2 provides the experimental results, comparing our proposal with Friedman's SEM and SEM-kMAX. Section 4.3 draws conclusions and recommends future research lines.

This chapter contains the work of Benjumeda et al. [2019c]. The software of the proposed method is available at `https://github.com/marcobb8/tr_bn`.

4.1 Tractable structural expectation-maximization

This section introduces our proposal for learning BNs with low inference complexity from incomplete datasets. Algorithm 4.1 is based on the SEM algorithm; however, it implements several changes toward guaranteeing efficient learning complexity and improving the model fitting with respect to the observed data. To ensure the tractability of Algorithm 4.1, it must limit the inference complexity of the BN candidates during the structure search. This is achieved by setting an upper bound t_b on the treewidth of the BN candidates. We use the strategy proposed in Section 3.2 to this end. With the objective of improving the performance of soft SEM, Algorithm 4.1 computes the score directly with respect to the observed data rather than the expected score to compare BN candidates. In Section 4.1.1, we analyze in more detail the advantages and difficulties of this approach.

Algorithm 4.1 performs as follows. In line 1, the BN structure $\mathcal{G}_0$, parameters $\boldsymbol{\theta}_0$, and the EO π_0 are initialized. In line 2, a valid ET is obtained from the BN $(\mathcal{G}_0, \boldsymbol{\theta}_0)$ and the EO π_0. Line 3 is the main loop of the algorithm, which iterates until convergence. Lines 4 (E-step) and 5 (M-step) perform a single iteration of the EM algorithm with the purpose of updating the parameters of the current BN candidate. The E-step and M-step are computed according to Equations (2.2) and (2.3), respectively. In lines 7–17, Algorithm 4.1 searches for the local change that improves the score the most with respect to the observed data and that satisfies the treewidth bound t_b. For each local change s_j, a new ET $\mathcal{E}'_{\mathcal{B}'}$ is incrementally compiled and optimized (line 9). The width of $\mathcal{E}'_{\mathcal{B}'}$ is used as an estimate of the treewidth of $\mathcal{G}'$. In line 11, $\mathcal{G}'$ is rejected if this estimate is greater than t_b. Otherwise, the parameters of the new structure $\mathcal{G}'$ are updated with the completed dataset D^+_{j+1} (line 12) using MLE (Equation (2.3)). Finally, the resultant BN $(\mathcal{G}', \boldsymbol{\theta}')$ is compared with the current best candidate (line

Input: Dataset $\mathcal{D}$, treewidth bound t_b
Output: Best BN structure $\mathcal{G}^*$ and parameters $\boldsymbol{\theta}^*$

select $\mathcal{G}_0$, $\boldsymbol{\theta}_0$, and EO π_0;
let $\mathcal{E}^0_{\mathcal{B}0}$ be a valid ET that represents π_0 for BN $(\mathcal{G}_0, \boldsymbol{\theta}_0)$ (Algorithm 3.1);
loop for $j = 0, 1, \ldots$ until convergence
    let $\mathcal{D}^+_{j+1}$ be the completed dataset given $\mathcal{D}$ and $\boldsymbol{\theta}_j$;
    let $\boldsymbol{\theta}_{j+1}$ be $\arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\mathcal{D}^+_{j+1})$;
    $\mathcal{G}_{j+1}, \mathcal{E}^{j+1}_{\mathcal{B}j+1} \leftarrow \mathcal{G}_j, \mathcal{E}^j_{\mathcal{B}j}$;
    let $\mathrm{s}_1, \ldots, \mathrm{s}_l$ be the local changes (i.e., arc additions, removals, or reversals) that can be applied to $\mathcal{G}_j$;
    for $d \in 1, \ldots, l$ **do**
        let $\mathcal{E}'_{\mathcal{B}'}$ be the result of compiling local change s_d in $\mathcal{E}^j_{\mathcal{B}j}$ (Algorithms 3.2 and 3.3), and optimizing the resultant ET (Algorithm 3.5);
        let $\mathcal{G}'$ be the structure of $\mathcal{B}'$;
        if $\mathrm{width}(\mathcal{E}'_{\mathcal{B}'}) \leq t_b$ **then**
            let $\boldsymbol{\theta}'$ be $\arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}|\mathcal{D}^+_{j+1})$;
            if $\mathrm{score}(\mathcal{D}_{j+1}, (\mathcal{G}', \boldsymbol{\theta}')) > \mathrm{score}(\mathcal{D}_{j+1}, (\mathcal{G}_{j+1}, \boldsymbol{\theta}_{j+1}))$ **then**
                $\mathcal{G}_{j+1}, \boldsymbol{\theta}_{j+1}, \mathcal{E}^{j+1}_{\mathcal{B}j+1} \leftarrow \mathcal{G}', \boldsymbol{\theta}', \mathcal{E}'_{\mathcal{B}'}$;
            end
        end
    end
$\mathcal{G}^*, \boldsymbol{\theta}^* \leftarrow \mathcal{G}_{j+1}, \boldsymbol{\theta}_{j+1}$;

Algorithm 4.1: Pseudocode of Tractable SEM.

13), and the model with the greater score for $\mathcal{D}$ is selected (line 14).

Theorem 4.1. *If the treewidth bound t_b is a constant (which is assumed to be small), then each iteration of Algorithm 4.1 consumes polynomial time in the number of variables (n) and number of instances in dataset $\mathcal{D}$.*

Proof. Algorithm 3.1 (line 2) requires polynomial time in n (Proposition 3.1). Maintaining the probabilistic completion $\mathcal{D}^+$ of the data requires exponential time and space to the number of missing values for each data case. Alternatively, the ESS of $\mathcal{D}^+$ can be computed prior to the parameter estimations at lines 5 and 12. Computing the ESS of each BN candidate requires performing M inference queries, one per each data instance. Computing the score for $\mathcal{D}$ (line 13) also demands performing M inference queries. Note that, although inference is NP-hard in the general case, when the treewidth of the models is bounded by a constant, it can be performed efficiently. In one iteration of Algorithm 4.1, there are a maximum of $\mathcal{O}(n^2)$ network candidates (i.e., one for each possible local change). Therefore, the total number of inference queries required is upper-bounded by $\mathcal{O}(n^2 M)$.

The maximum likelihood parameters for a structure $\mathcal{G}$ and the completed dataset $\mathcal{D}^+$ can also be computed in polynomial time (lines 5 and 12). The number of possible local changes in a graph $\mathcal{G}$ is quadratic in the number of nodes. Thus, the loop at line 8 uses a number of iterations l that is also quadratic in n. The width of an ET $\mathcal{E}'_{\mathcal{B}'}$ (line 11) is the length of its largest cluster munus one (Definition 3.7), which can be computed in time $O(n \cdot \text{width}(\mathcal{E}'_{\mathcal{B}'}))$ (see Lemma A.2). Finally, Algorithms 3.2, 3.3 and 3.5 (line 9) take polynomial time in n (Theorem 3.2). Therefore, each iteration can be computed efficiently. □

Theorem 4.1 ensures that each iteration of tractable SEM is computed efficiently under the described constraints. This implies that if Algorithm 4.1 loops for a polynomial number of iterations, the complete process is performed efficiently. Corollary 4.1 provides guarantees on the computational complexity of Algorithm 4.1.

Corollary 4.1. *If Algorithm 4.1 satisfies the conditions described in Theorem 4.1, the stopping condition for the loop at line 3 is $\mathcal{G}_{j+1} = \mathcal{G}_j$ and the local changes considered at line 7 are limited to arc additions, then, the complexity of Algorithm 4.1 is polynomial in the number of variables and the number of instances in the dataset.*

Proof. From Theorem 4.1, each iteration of Algorithm 4.1 must consume polynomial time. As the stopping criterion is $\mathcal{G}_{j+1} = \mathcal{G}_j$, the maximum number of iterations of the loop at line 3 is the number of local changes that can be applied to $\mathcal{G}$. Given that the local changes are limited to arc additions, and a complete graph of n nodes has $\frac{n(n-1)}{2}$ arcs, the maximum number of iterations that this algorithm could perform is upper-bounded by n^2. Thus, if the above conditions are fulfilled, Algorithm 4.1 performs a polynomial number of iterations, each in polynomial time. Hence, its complexity is polynomial in n and M. □

4.1.1 Optimizing the score with respect to the observed data

One of the most celebrated properties of soft SEM is that it ensures that the scoring function does not decrease with respect to the observed data $\mathcal{D}$ after each iteration. According to Equation (2.4), given a model $\mathcal{B}_0$ and a dataset $\mathcal{D}^+$, where $\mathcal{D}^+$ is the probabilistic completion of $\mathcal{D}$ by $\mathcal{B}_0$, any improvement in the expected score ($score(\mathcal{D}^+, \mathcal{B}) \geq score(\mathcal{D}^+, \mathcal{B}_0)$) results in an improvement in the score with respect to the observed data ($score(\mathcal{D}, \mathcal{B}) \geq score(\mathcal{D}, \mathcal{B}_0)$). Thus, soft SEM ensures the convergence of the score with respect to the observed data to a local optimum. The advantage of using the expected score rather than the score with respect to the observed data is that once the data is completed during the E-step, the expected score decomposes and therefore the local changes in the network locally affect the nodes, facilitating more efficient learning (e.g., by using cache).

Despite the desirable properties of soft SEM, Equation (2.4) offers no guarantee that the model selected at the end of the search is near to the optimum. For example, if a greedy search is used, only the first local change guarantees an improvement in the score for the observed data. Therefore, if we aim to ensure improvements on the score with respect to the observed data after applying every local change, the data must be repeatedly completed, preventing soft SEM from exploiting the decomposition of the score. Moreover, a local change that improves the score with respect to the observed data may not improve the expected score. If many local changes are incorrectly rejected, the learning process could terminate early.

These problems can be overcome by directly using the score with respect to the observed data. The bottleneck of Algorithm 4.1 is the inference required for the computation of the ESS and score for each network candidate (line 7). Although the cost of this process is polynomial in the number of variables n and the number of instances of the dataset M when the treewidth of the BN candidates is small, the computational time required to answer all the inference queries can be high when n or M are large.

Next, we propose a heuristic with the objective of reducing the number of inference queries during the learning process while ensuring that each local change applied improves the score with respect to the observed data. We modify Algorithm 4.1 as follows:

- At line 7, the data is completed with hard assignments and the local changes $\mathrm{s}_1, \ldots, \mathrm{s}_l$ are ordered according to their score for the completed dataset.

- The loop at line 8 terminates when the first local change s_d that improves the score with respect to the observed data is identified (i.e., we follow a best-first strategy).

Ordering the local changes as suggested above requires imputing the data once. Testing all the local changes until an improvement is identified prevents Algorithm 4.1 from falling into local optima in the early stages of the search. In Section 4.2, we evaluate this strategy. In the experiments, the proposed approach outperformed soft SEM and SEM-kMAX in terms of fitting the data and imputation accuracy.

4.2 Experimental results

In this section, we empirically evaluate the performance of tractable SEM in terms of data fitting, imputation accuracy, and computational complexity. First, in Section 4.2.1, we compare the proposed approach with soft SEM to highlight the advantages of directly computing the score with respect to the observed data and the reduction in the computational cost provided by the strategy proposed in Section 4.1.1. Then, in Section 4.2.2, we compare tractable SEM with SEM-kMAX for analysing the performance of the proposed approach in real world datasets of varied dimensionalities.

The experiments below include two versions of tractable SEM: TSEM implements Algorithm 4.1 and uses the heuristic described in Section 4.1.1 to accelerate the learning process. TSEM-poly also fulfills the conditions required by Corollary 4.1 to ensure polynomial computational complexity.

For all the methods, the score function to maximize is BIC. We analyzed the significance of the differences found for each performance measure in all experiments using the Friedman test with $\alpha = 0.05$ and Holm's [Holm, 1979] and Shaffer's [Shaffer, 1986] post-hoc procedures. Both methods are explained in Section 3.3.1.

Experiments were performed on a computer with an Intel Core i7-6700K CPU at 4.00 GHz with 16 GB main memory, running Ubuntu 16.04 LTS. TSEM, TSEM-poly, and soft SEM were written in Python 2.7.12, and integrate specific functions developed in C++11 (version 5.4.0). SEM-kMAX was downloaded from `http://ipg.idsia.ch/software/blip` and is written in Java.

4.2.1 Comparison with soft SEM

This first experimental study highlights the differences between using TSEM, TSEM-poly, and soft SEM. The goal is to compare the models output using the three methods in diverse scenarios according to a set of performance measures. To compare these methods, we generated synthetic data from 11 real-world BNs. These BNs were obtained from the bnlearn BN repository (`http://www.bnlearn.com/bnrepository/`), and are cited therein. Table 4.1 lists the number of variables (N. vars), arcs (N. arcs), and parameters (N. parameters) of each BN. To include a wide variety of scenarios, we generated training and testing datasets of different sample sizes (500, 2000, and 5000) and different percentages of missing values (30%, 50%, and 70%) from the above networks. In TSEM and TSEM-poly, we set the treewidth bound t_b to 5, which in our experience provides an acceptable trade-off between efficiency and expressiveness. In Section 4.2.2, we evaluate the performance of TSEM and TSEM-poly with different treewidth bounds.

Tables 4.2–4.4 display the experimental results of comparing the above approaches. We use the following performance measures: BIC is the BIC score of the models with respect to the observed values in the training dataset, LL is the log-likelihood of the models in the test dataset, acc is the imputation accuracy in the training dataset, and time is the learning time (in seconds). For each performance measure, the mean rank $\pm$ the standard deviation of

Table 4.1: Basic properties of BNs used to generate synthetic datasets.

Dataset	N. vars	N. arcs	N. parameters
SACHS	11	17	178
ALARM	37	46	509
BARLEY	48	84	114,005
CHILD	20	25	230
INSURANCE	27	52	984
MILDEW	35	46	540,150
WATER	32	66	10,083
HAILFINDER	56	66	2,656
HEPAR II	70	123	1,453
WIN95PTS	76	112	574
PATHFINDER	135	200	77,155

Table 4.2: Comparison of methods in all datasets with 500 instances. Best results are denoted in boldface.

	TSEM	TSEM-poly	soft SEM
mean rank BIC	**1.42±0.5**	2.36±0.55	2.21±0.99
mean rank LL	**1.3±0.47**	2.27±0.52	2.42±0.9
mean rank acc	**1.15±0.36**	2.09±0.38	2.76±0.66
mean rank time	1.82±0.39	**1.18±0.39**	3±0

each method over all the datasets is displayed. The ranking of the methods is given by their average performance (BIC, LL, acc, and time) compared to the others (i.e., the best is ranked first and the worst is ranked third). The detailed results are supplied at https://github.com/marcobb8/tr_bn/blob/master/Supplementary%20Material/supplemantary-material_SEM.pdf.

Figures 4.1–4.4 graphically present the results obtained with Holm's and Shaffer's procedures for each performance measure in all datasets. Each figure represents, in fact, both procedures, given that the significant differences observed by Shaffer's procedure are identical to those observed by Holm's procedure.

In the case of BIC (see Figure 4.1), significant differences were found among all the methods. TSEM performed the best, followed by TSEM-poly and soft SEM. The results suggest that these differences are caused by the inability of the expected score to recognise many of the changes that improve BIC with respect to the observed data. For example, of all the local changes performed by TSEM, on average, only 19% improved the expected

Table 4.3: Comparison of methods in all datasets with 2000 instances. Best results are denoted in boldface.

	TSEM	TSEM-poly	soft SEM
mean rank BIC	**1.15±0.36**	2.18±0.39	2.67±0.74
mean rank LL	**1.33±0.48**	2.06±0.61	2.61±0.79
mean rank acc	**1.21±0.42**	1.97±0.47	2.82±0.58
mean rank time	1.91±0.38	**1.15±0.36**	2.94±0.35

Table 4.4: Comparison of methods in all datasets with 5000 instances. Best results are denoted in boldface.

	TSEM	TSEM-poly	soft SEM
mean rank BIC	**1.06±0.24**	2.06±0.24	2.88±0.48
mean rank LL	**1.12±0.33**	1.94±0.35	2.94±0.35
mean rank acc	**1.12±0.33**	1.94±0.35	2.94±0.35
mean rank time	1.97±0.39	**1.12±0.33**	2.91±0.38

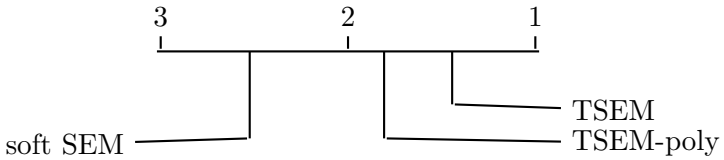


Figure 4.1: Comparison of mean rank BIC scores in training dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

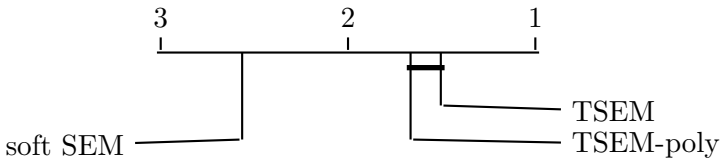


Figure 4.2: Comparison of mean rank log-likelihood (LL) in test dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

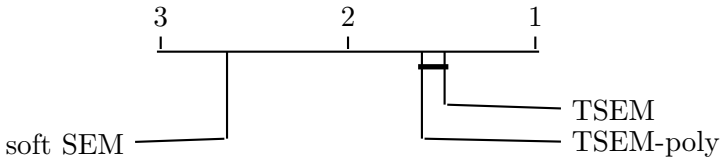


Figure 4.3: Comparison of mean rank imputation accuracy (acc) in training dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

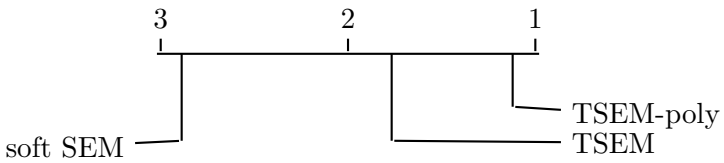


Figure 4.4: Comparison of mean rank learning time in training dataset with Holm's and Shaffer's post-hoc procedures in synthetic datasets.

score. This leads TSEM to apply, on average, 33% more local changes than soft SEM. The rare situations where soft SEM achieved a greater BIC score than TSEM and TSEM-poly correspond to small datasets with a high percentage of missing values. In these situations, the BN structures output by all the methods were extremely sparse or even completely unconnected. As TSEM and TSEM-poly terminate when they do not find a local change that improves the BIC score, both methods clearly require less time than soft SEM to optimise the parameters. A method of improving the performance of the proposed approach in these situations would be to perform the EM algorithm until convergence in the output models.

Figures 4.2 and 4.3 compare the log-likelihood and imputation accuracy of the models, respectively. TSEM and TSEM-poly performed significantly better than soft SEM for both measures. However, no significant differences were found between TSEM and TSEM-poly.

Figure 4.4 indicates significant differences among the learning times of all the methods; TSEM-poly was the fastest method overall, followed by TSEM and soft SEM. As these experiments were only performed in datasets generated from medium and small size BNs, the treewidth of the models learnt by soft SEM was never high (seven in the worst case). Therefore, the differences in the computational time can be explained by the number of times that each approach computed the parameters. For example, soft SEM required computing the ESS an average 61 more times than TSEM. Note that as soft SEM does not bound the treewidth of the models during the learning process, larger datasets typically lead to models with greater treewidth, where exact inference, and therefore computing the ESS is unfeasible.

4.2.2 Comparison with SEM-kMAX

In this section, the proposed approach is compared with SEM-kMAX for learning bounded treewidth BNs from incomplete datasets. To perform the experiments, we used 22 real-world datasets of varied dimensionalities and sample sizes. The details of these datasets are given in Section 3.3. Table 3.1 provides the number of variables and number of training and testing instances in each dataset.

We set several scenarios to evaluate both methods according to the performance measures used in Section 4.2.1. For each real-world dataset, we input, at random, different percentages of missing values (30%, 50%, and 70%), and tested the methods in all the situations. For each method, we learned a model using four different treewidth bounds (2, 3, 4, and 5). We set the parameters of SEM-kMAX to the values recommended by Scanagatta et al. [2018b]. Concretely, they set an execution time of n s (i.e., a second for each variable) to compute the cache of the most optimum parent sets and $n/10$ s for the structure search. Tables 4.5–4.8 display the mean rank $\pm$ the standard deviation (over all the datasets) of each method for every performance measure given a treewidth bound t_b. The detailed results are supplied at https://github.com/marcobb8/tr_bn/blob/master/Supplementary%20Material/supplemantary-material_SEM.pdf.

The significance results obtained with Holm's and Shaffer's post-hoc procedures over all datasets and treewidth bounds are displayed in Figures 4.5–4.8.

TSEM and TSEM-poly performed better than SEM-kMAX in all experiments in terms

Table 4.5: Comparison of methods in all datasets, using treewidth bound of 2. Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
mean rank BIC	**1.17±0.38**	1.83±0.38	3±0
mean rank LL	**1.35±0.48**	1.65±0.48	3±0
mean rank time	1.94±0.24	**1.06±0.24**	3±0
mean rank acc	**1.42±0.58**	1.79±0.54	2.79±0.62

Table 4.6: Comparison of methods in all datasets, using treewidth bound of 3. Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
mean rank BIC	**1.17±0.38**	1.83±0.38	3±0
mean rank LL	**1.48±0.5**	1.52±0.5	3±0
mean rank time	1.95±0.21	**1.05±0.21**	3±0
mean rank acc	**1.55±0.59**	1.68±0.61	2.77±0.63

Table 4.7: Comparison of methods in all datasets, using treewidth bound of 4. Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
mean rank BIC	**1.17±0.38**	1.83±0.38	3±0
mean rank LL	**1.35±0.48**	1.65±0.48	3±0
mean rank time	1.85±0.36	**1.15±0.36**	3±0
mean rank acc	1.67±0.64	**1.64±0.6**	2.7±0.72

Table 4.8: Comparison of methods in all datasets, using treewidth bound of 5. Optimal results are denoted in boldface.

	TSEM	TSEM-poly	SEM-kMAX
mean rank BIC	**1.21±0.41**	1.79±0.41	3±0
mean rank LL	**1.44±0.5**	1.56±0.5	3±0
mean rank time	1.85±0.36	**1.15±0.36**	3±0
mean rank acc	1.68±0.68	**1.62±0.55**	2.7±0.72

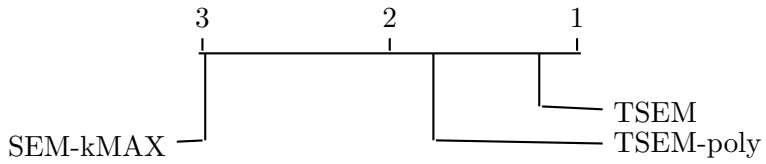


Figure 4.5: Comparison of mean rank BIC scores in training dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

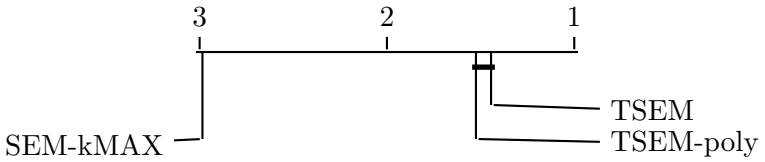


Figure 4.6: Comparison of mean rank log-likelihood (LL) in test dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

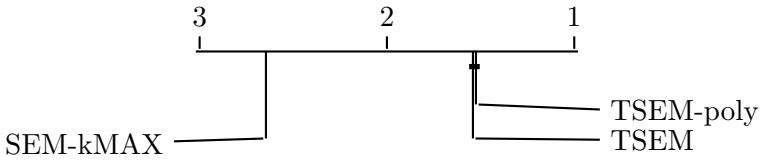


Figure 4.7: Comparison of mean rank imputation accuracy (acc) in the training dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

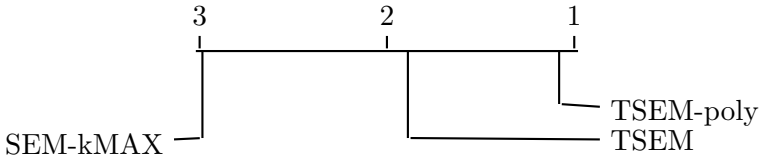


Figure 4.8: Comparison of mean rank learning time in training dataset with Holm's and Shaffer's post-hoc procedures in real-world datasets.

of BIC and LL, and achieved a significantly greater imputation accuracy. The treewidth of the models output by each method suggests that the proposed approach leads to a tighter treewidth bound fitting than SEM-kMAX (details are provided in the Supplementary Material). Although TSEM led to significantly greater BIC scores than TSEM-poly, the difference in the majority of cases was small. This explains why the results of both methods in terms of LL and imputation accuracy were similar. Finally, TSEM and TSEM-poly executed faster than SEM-kMAX in all experiments. However, we must be cautious when interpreting these results given that these methods were implemented in different programming languages. Although TSEM-poly executed faster than TSEM in the majority of cases, the differences in learning time in each particular case were small.

4.3 Conclusions

In this study, we addressed the problem of learning BNs in the case of missing values and hidden variables in tractable time. We proposed an adaptation of SEM that ensures the efficiency of the E-step by bounding the inference complexity of the BN candidates. To limit their inference complexity, we proposed the use of an efficiency-focused heuristic for searching for low-width EOs. We demonstrated that the resulting algorithm consumes polynomial time under certain conditions. Further, we analyzed the advantages of using the score with respect

to the observed data directly, rather than the expected score, and provided a heuristic to reduce the number of inference queries performed during the learning process.

As demonstrated by the experimental results, the proposed approach outperformed soft SEM and SEM-kMAX based on all the tested evaluation metrics. Apparently, these differences were a consequence of the advantages of directly optimising the score with respect to the observed data. Moreover, the proposed heuristics lead to a significant reduction in the computational cost of the learning process.

Friedman adapted soft SEM to the Bayesian learning of BNs [Friedman, 1998], which entails several advantages. For example, it provides a method to incorporate prior knowledge and a superior evaluation of the generalization properties of a model given the data. Adapting our proposal to use Bayesian scoring functions would be relatively straightforward, and we intend to do this in the future.

At present, there is an increasing interest in learning with hidden variables from high dimensional spaces. Examples are multidimensional clustering [Poon et al., 2013; Keivani and Peña, 2016] and learning deep probabilistic graphical models [Zhang, 2004; Poon and Domingos, 2011]. We consider that our proposal is effective for these tasks, and we intend to study its application to these problems.

Chapter 5

Tractability of most probable explanations in multidimensional Bayesian network classifiers

The problem of multidimensional classification is common in several domains like text categorization (a text can be assigned to multiple topics), medicine (a patient may suffer from several diseases) or system monitoring (a system may break down from multiple failures). MBCs are BNs with a restricted topology, where no arcs from feature variables to class variables are allowed. Inference in MBCs may have a high computational cost for some structures, even when the class and feature subgraphs are restricted to trees or polytrees.

Although there is work in the literature addressing the problem of computational complexity in MBCs, the focus has not been on taking advantage of the most common type of queries of such models. In this chapter, we study the computational complexity of MPEs and marginals of class variables in MBCs when an instantiation of the feature variables is given. We also provide upper bounds on the complexity of these models given additional restrictions on their structure that limit the treewidth of a transformation of it that we call the pruned graph. We propose a learning method that uses these properties to search for tractable MBCs in the space of topological orderings.

This chapter is organized as follows. Section 5.1 presents the new theoretical results with respect to the complexity of computations of MPEs and marginals in MBCs. Section 5.2 describes the method proposed for learning tractable MBCs. Section 5.3 reports the experimental results. Section 5.4 draws some conclusions and suggests future research lines.

This chapter 3 adapts the work of Benjumeda et al. [2018a], which is an extension of Benjumeda et al. [2016]. The software of the proposed method is available at https://github.com/marcobb8/tr_bn.

5.1 Theoretical results on most probable explanations and marginal computations

In BNs with bounded treewidth, both evidence propagation and MPEs can be computed in polynomial time. In the case of MBCs (which are, in fact, also BNs), this is also true, but it is possible to exploit the restrictions on the network structure and the information about the queries sent to the MBCs. From the structure of MBCs, we know that there are no arcs from the feature to the class nodes. Also, the values of all the features should appear in the evidence.

As multidimensional classification in MBCs involves obtaining the MPE of the class variables given an instantiation of all the feature variables, we focus on finding bounds for this problem. Nevertheless, the results are extended to marginal computations because it is sometimes worth calculating the probability of a configuration of certain class variables given the value of all the features, and the extension is straightforward.

The complexity of inference in BNs is query dependent, given that the parameters of a network can be updated with the value of the evidence variables before performing inference.

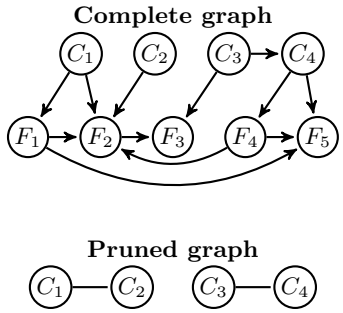
Definition 5.1. *Let $\mathcal{G} = (\mathcal{C} \cup \mathcal{F}, \mathcal{A}_C \cup \mathcal{A}_B \cup \mathcal{A}_F)$ be the structure of an MBC $\mathcal{B}$. The pruned graph $\mathcal{G}'$ of $\mathcal{G}$ is the result of moralizing $\mathcal{G}$ and then removing the feature nodes from the resulting graph.*

When computing the MPE in a BN given an evidence $\mathbf{f}$, we can simplify the structure of the network by pruning every arc $X_i \to X_j$ such that X_i appears in $\mathbf{f}$. Pruning arc $X_i \to X_j$ for evidence $\mathbf{f}$ from a BN means removing arc $X_i \to X_j$ and the parameters of X_j that are not compatible with $\mathbf{f}$. Previous research uses the treewidth of $\mathcal{G}$ to bound the inference complexity, exploiting the restrictions on the topology of $\mathcal{G}$, but without considering the known query-dependent information, that is, that all the feature variables are instantiated when we compute the MPE in $\mathcal{B}$.

Theorem 5.1 states that MPE and marginal computations in an MBC are tractable if the treewidth of its pruned graph is bounded. This transformation was used by Pastink and van der Gaag [2015] to bound the treewidth of Tree–empty MBCs. Here, we extend it to bound the complexity of (the more general) DAG–DAG MBCs.

Theorem 5.1. *Let $\mathcal{G} = (\mathcal{C} \cup \mathcal{F}, \mathcal{A}_C \cup \mathcal{A}_B \cup \mathcal{A}_F)$ be the structure of an MBC $\mathcal{B}$, and $\boldsymbol{f}$ be an instantiation of $\mathcal{F}$. If the treewidth of its pruned graph $\mathcal{G}'$ and the number of parents of each node that belongs to $\mathcal{F}$ are bounded, $\mathcal{B}$ can compute MPEs and marginals in polynomial time given $\boldsymbol{f}$.*

Proof. Suppose that the CPD of each node $X_i \in \mathcal{C} \cup \mathcal{F}$ is represented by a potential ϕ_i. ϕ_i is updated with $\mathbf{f}$ by removing the entries that are not compatible with $\mathbf{f}$. This can be done in linear time in the size of ϕ_i, that is exponential in the number of parents of X_i in $\mathcal{G}$. Hence, the nodes in $\mathcal{F}$ can be updated with $\mathbf{f}$ in polynomial time if the number of parents of each node in $\mathcal{F}$ is bounded.



Potential	Before	After
ϕ_{C_1}	$\{C_1\}$	$\{C_1\}$
ϕ_{C_2}	$\{C_2\}$	$\{C_2\}$
ϕ_{C_3}	$\{C_3\}$	$\{C_3\}$
ϕ_{C_4}	$\{C_3, C_4\}$	$\{C_3, C_4\}$
ϕ_{F_1}	$\{C_1, F_1\}$	$\{C_1\}$
ϕ_{F_2}	$\{C_1, C_2, F_1, F_2, F_4\}$	$\{C_1, C_2\}$
ϕ_{F_3}	$\{C_3, F_2, F_3\}$	$\{C_3\}$
ϕ_{F_4}	$\{C_4, F_4\}$	$\{C_4\}$
ϕ_{F_5}	$\{C_4, F_1, F_4\}$	$\{C_4\}$

Figure 5.1: MBC structure and pruned graph (left), and domain of the potential of each node before and after they are updated with evidence $\mathbf{f} = (f_1, \ldots, f_5)$ (right). Note that the treewidth of the pruned graph is smaller than the number of class variables.

After updating $\mathcal{G}$ with $\mathbf{f}$, the domain of each potential ϕ_f of $X_f \in \mathcal{F}$ is $\mathbf{Pa}^{\mathcal{G}}_{X_f} \cap \mathcal{C}$. There is an undirected link in $\mathcal{G}'$ between each node in $\mathbf{Pa}^{\mathcal{G}}_{X_f} \cap \mathcal{C}$. It is evident that the width of the best elimination order for the resulting potentials is equal to the treewidth of $\mathcal{G}'$. As the width of the best elimination order bounds the complexity of MPE and marginal computations, if the treewidth of $\mathcal{G}'$ is bounded, $\mathcal{B}$ can compute MPEs and marginals in polynomial time given $\mathbf{f}$. □

Figure 5.1 shows an example of the structure of an MBC and its pruned graph. It also illustrates that all the variables belonging to the domain of the same potential $\phi_i \in \{\phi_{C_1}, \ldots, \phi_{C_4}, \phi_{F_1}, \ldots, \phi_{F_5}\}$ updated with an instance $\mathbf{f} = (f_1, \ldots, f_5)$ of the features are connected by a link in the pruned graph (and vice versa). This means that the treewidth of the pruned graph is equal to the width of the best elimination order in the updated potentials.

Although the computational cost of calculating the treewidth of the pruned graph $\mathcal{G}'$ is less than calculating the treewidth of the complete structure $\mathcal{G}$, the exact computation of the treewidth of a graph is an NP-complete problem [Arnborg et al., 1987].

In Section 2.2.2 several treewidth estimation techniques are reviewed, but using them for each candidate during the structure search can be very computationally demanding. Alternatively, the incremental compilation of ETs (see Chapter 3) can be easily adapted to bound the treewidth of the pruned graph. In Section 5.2 we propose a method that uses the above approach to learn tractable MBCs.

However, there are special cases where it is not necessary to compute the treewidth of the models to ensure their tractability. Collorary 5.1 shows that if the number of class variables of an MBC $\mathcal{B}$ is bounded, then we can perform inference in $\mathcal{B}$ in polynomial time.

Corollary 5.1. *Let $\mathcal{G} = (\mathcal{C} \cup \mathcal{F}, \mathcal{A}_C \cup \mathcal{A}_B \cup \mathcal{A}_F)$ be the structure of an MBC $\mathcal{B}$, and $\boldsymbol{f}$ be an instantiation of $\mathcal{F}$. If the number of class variables d and the number of parents of each node in $\mathcal{F}$ are bounded, $\mathcal{B}$ can compute MPEs and marginals in polynomial time given $\boldsymbol{f}$.*

Proof. Let $\mathcal{G}'$ be the pruned graph of $\mathcal{G}$. As each node in $\mathcal{G}'$ belongs to $\mathcal{C}$, treewidth$(\mathcal{G}') \leq d$. Hence, from Theorem 5.1 we know that if the number of parents of each feature and d are

bounded, $\mathcal{B}$ can compute MPEs and marginals in polynomial time given $\mathbf{f}$. □

As the pruned graph only contains class nodes, it is patent that its treewidth is always smaller than the number of class variables in the classifier. Let us consider the MBC shown in Figure 5.1 and its pruned graph. The nodes in the pruned graph are $\{C_1, \ldots, C_4\}$, so its treewidth can never be greater than 3.

Additionally, if the classifier is CB-decomposable, the maximum number of class nodes per component is an upper bound in the inference complexity of the MBCs, as shown in Collorary 5.2.

Corollary 5.2. *Let $\mathcal{G} = (\mathcal{C} \cup \mathcal{F}, \mathcal{A}_C \cup \mathcal{A}_B \cup \mathcal{A}_B)$ be the structure of a CB-decomposable MBC $\mathcal{B}$, and $\boldsymbol{f}$ be an instantiation of $\mathcal{F}$. If the number of class variables in each component of $\mathcal{G}$ and the number of parents of each node in $\mathcal{F}$ are bounded, $\mathcal{B}$ can compute MPEs and marginals in polynomial time given $\boldsymbol{f}$.*

Proof. Let $\mathcal{G}'$ be the pruned graph of $\mathcal{G}$. If $\mathcal{G}$ is CB-decomposable into r components $\mathcal{G}_1, \ldots, \mathcal{G}_r$, then $\mathcal{G}'$ is composed of r unconnected subgraphs $\mathcal{G}'_1, \ldots, \mathcal{G}'_r$, such that $\mathcal{X}'_i = \mathcal{X}_i \cap \mathcal{C}$, $i = 1, \ldots, r$, where $\mathcal{X}_i$ and $\mathcal{X}'_i$ are the nodes in $\mathcal{G}_i$ and $\mathcal{G}'_i$, respectively. As $\mathrm{tw}(\mathcal{G}') = \max_i \mathrm{tw}(\mathcal{G}'_i) < \max_i |\mathcal{X}'_i| = \max_i |\mathcal{X}_i \cap \mathcal{C}|$, we know from Theorem 5.1 that if the number of parents of each feature and the number of class variables in each component of $\mathcal{G}$ are bounded, $\mathcal{B}$ can compute MPEs and marginals in polynomial time given $\mathbf{f}$. □

Figure 5.1 shows that the treewidth of the pruned graph is bounded by the maximum number of class variables per component (two in this case), given that there is no path from C_i to C_j in the pruned graph if two class nodes C_i and C_j are in two different connected components.

5.2 Learning tractable multidimensional Bayesian network classifiers

Given that inference in a BN is tractable if the treewidth of its structure is bounded, most existing algorithms for learning BNs with low inference complexity bound the treewidth of the networks during the learning process, rejecting any candidate that exceeds the treewidth bound.

In the case of an MBC, instead of bounding the treewidth of its complete structure, we focus on bounding the treewidth of its pruned graph. The simplest solution would be to force the CB-decomposability of the MBC by setting a maximum number of class variables per component. However, this restriction may be too hard in practice when the data contains more complex relationships among the variables. Algorithm 5.1 consists of two parts. First (lines 3-15), the class and bridge subgraphs are simultaneously learned, and the incremental compilation of ETs is used to bound the treewidth of the MBC structure. Second (lines 16-25), the feature subgraph is learned in a greedy manner. As the arcs between features do

not affect the treewidth of the pruned graph, it is not necessary to check the tractability of the models during this phase.

Input: Dataset $\mathcal{D}$, treewidth bound t_b
Output: Best MBC $\mathcal{B}^*$
1 initialize MBC $\mathcal{B}^*$ and EO π^*;
2 let $\mathcal{E}^*_{\mathcal{B}^*}$ be a valid ET that represents π^* for MBC $\mathcal{B}^*$ (Algorithm 3.1);
3 **loop** for $j = 0, 1, \ldots$ until convergence
4 let $\mathrm{s}_1, \ldots, \mathrm{s}_l$ be the local changes (i.e., arc additions, removals, or reversals) that can be applied to $\mathcal{G}^*$ (the structure of $\mathcal{B}^*$) such that:
5 **a)** If local change s_d is arc addition $X_{\text{out}} \to X_{\text{in}}$, then $X_{\text{out}} \in \mathcal{C}$
6 **b)** If local change s_d is arc reversal $X_{\text{out}} \to X_{\text{in}}$, then $X_{\text{out}} \in \mathcal{C}$ and $X_{\text{in}} \in \mathcal{C}$
7 **for** $d \in 1, \ldots, l$ **do**
8 let $\mathcal{B}'$ be the result of applying local change s_d in $\mathcal{B}^*$;
9 let $\mathcal{E}'_{\mathcal{B}'}$ be the result of compiling local change s_d in $\mathcal{E}^*_{\mathcal{B}^*}$ (Algorithms 3.2 and 3.3), and optimizing the resulting ET (Algorithm 3.5);
10 **if** $\text{width}(\mathcal{E}'_{\mathcal{B}'}) \leq t_b$ **then**
11 **if** $\text{score}(\mathcal{D}, \mathcal{B}') > \text{score}(\mathcal{D}, \mathcal{B}^*)$ **then**
12 $\mathcal{B}^*, \mathcal{E}^*_{\mathcal{B}^*} \leftarrow \mathcal{B}', \mathcal{E}'_{\mathcal{B}'}$;
13 **end**
14 **end**
15 **end**
16 **loop** for $j = 0, 1, \ldots$ until convergence
17 let $\mathrm{s}_1, \ldots, \mathrm{s}_l$ be the local changes (i.e., arc additions, removals, or reversals) that can be applied to $\mathcal{G}^*$ (the structure of $\mathcal{B}^*$) such that:
18 **a)** If local change s_d is arc addition $X_{\text{out}} \to X_{\text{in}}$, then $X_{\text{out}} \in \mathcal{F}$, $X_{\text{in}} \in \mathcal{F}$, $\mathbf{Pa}^{\mathcal{G}^*}_{X_{\text{out}}} \cap \mathcal{C} \neq \emptyset$, and $\mathbf{Pa}^{\mathcal{G}^*}_{X_{\text{in}}} \cap \mathcal{C} \neq \emptyset$
19 **b)** If local change s_d is arc reversal $X_{\text{out}} \to X_{\text{in}}$, then $X_{\text{out}} \in \mathcal{F}$, $X_{\text{in}} \in \mathcal{F}$
20 **for** $d \in 1, \ldots, l$ **do**
21 let $\mathcal{B}'$ be the result of applying local change s_d in $\mathcal{B}^*$;
22 **if** $\text{score}(\mathcal{D}, \mathcal{B}') > \text{score}(\mathcal{D}, \mathcal{B}^*)$ **then**
23 $\mathcal{B}^* \leftarrow \mathcal{B}'$;
24 **end**
25 **end**

Algorithm 5.1: Pseudocode of greedy search of tractable MBCs (GS–pruned).

Algorithm 5.1 proceeds as follows. In line 1, an initial BN $\mathcal{B}^*$ and EO π^* are selected. A valid ET that represents π^* for MBC $\mathcal{B}^*$, which is later used to bound the complexity of the models, is obtained at line 2. The loop at line 3 represents a greedy search that simultaneously learns the class and bridge subgraphs of the MBC. Lines 5 and 6 make sure that each candidate is an MBC with empty feature subgraph. Lines 9 and 10 check if the new candidate violates the treewidth bound. Otherwise, the chosen network is updated if the new candidate improves the objective score (lines 11 and 12). We do not specify a definite scoring function because any score (see Section 2.3) used to evaluate BNs can be applied. We assume that the score is decomposable and must be maximized. When none of the l

	Family	Addresses complexity	Theoretical guarantees	Bound
[van der Gaag and de Waal, 2006]	Tree–tree			
[de Waal and van der Gaag, 2007]	PolyTree–polytree			
[Bielza et al., 2011]	DAG–DAG			
[Pastink and van der Gaag, 2015]	Tree–empty	x	x	Complete graph
[Corani et al., 2014]	Forest–empty	x		
[Borchani et al., 2010]	DAG–DAG	x		
This work	DAG–DAG	x	x	Pruned graph

Table 5.1: Comparison of the properties of different MBC learning methods. For each approach, the table shows the family of MBC returned by the method, whether or not it addresses the problem of computational complexity of the learned models, whether or not it provides theoretical guarantees on the tractability of the models, and which part of the structure is bounded to ensure tractability. Note that Pastink and van der Gaag [2015] provide an optional step to augment the empty feature subgraph to a forest (to give a Tree–forest MBC).

local changes improves the score, the method exits from the loop. At this point, the class and bridge subgraphs of $\mathcal{B}^*$ have been learned, and its treewidth is at most t_b. As the arcs between the features do not affect the treewidth of the pruned graph we can learn the feature subgraph without checking the tractability of the MBC candidates. The loop at line 16 learns the feature subgraph in a greedy manner. Line 18 allows only arc additions between feature variables that are children of at least one class variable. Finally, the local change that optimizes the objective score is applied (lines 20-25).

Table 5.1 compares the properties of our method with other popular MBC learning methods in the state of the art. Note that our approach is one of the few that provides theoretical guarantees with respect to the complexity of the models. Also, it allows highly expressive structures to be learned since it does not bound the treewidth of the complete graph.

5.3 Experimental results

To test the performance of our proposal, we compared it with the next MBC learning methods. First, we compare our approach with methods that learn a more restricted network topology, including Tree–tree [van der Gaag and de Waal, 2006] and a version of the method proposed by Pastink and van der Gaag [2015], which we refer to as Tree–tw. Instead of the branch and bound approach that they proposed to search the bridge subgraph, we used a greedy search process that bounds the treewidth of the candidates using Algorithms 3.2–3.5, given that the computational cost of the former is too high for this experimental framework.

To test the effects of bounding the treewidth of the pruned graph, we compare GS–pruned with two adaptations of Algorithm 5.1. GS–MBC does not bound the inference complexity of the models, while GS–tw bounds the treewidth of the complete graph. Note that GS–MBC is an adaptation of the pure filter algorithm [Bielza et al., 2011] that uses hill-climbing instead of the K2 algorithm. Unlike the latter, GS–MBC does not require predefining a topological ordering and it allows the use of a cache. In all cases, we use BIC as the scoring function to

Table 5.2: Basic properties of the multilabel datasets: Number of class variables (N. class), number of feature variables (N. feat.) and number of instances (N. inst.). The datasets are sorted in ascending order according to N. class × N. inst.

Dataset	N. class	N. feat.	N. inst.
emotions	6	72	593
foodtruck	12	21	407
birds	19	260	645
scene	6	294	2407
genbase	27	1186	662
yeast	14	103	2417
medical	45	1186	978
enron	53	1001	1702
ohsumed	23	1002	13929
reutersk500	103	500	6000
tmc2007_500	22	500	28596
bibtex	159	1836	7395
corel5k	374	499	5000
imdb	28	1001	120919
mediamill	101	120	43907

be optimized.

Additionally, we performed the experiments with two transformation methods. CDL2 [Guo and Gu, 2011] uses ridge logistic regression to train the base classifiers and a conditional dependency network (CDN), i.e., a full cyclic directed graphical model, encodes the relationships among the outcomes of the base classifiers and the class variables. SVM–struct implements the cutting-plane training of structural support vector machines (SVMs) [Joachims et al., 2009]. The class variables are connected by a pairwise Markov random field with the tree structure retrieved with the Chow-Liu algorithm [Chow and Liu, 1968].

We applied 5-fold cross-validation in 15 real-world multilabel datasets extracted from the MULAN data repository http://mulan.sourceforge.net/datasets-mlc.html and the mldr.datasets R package https://github.com/fcharte/mldr.datasets. The articles that correspond to each dataset are cited therein. Table 5.2 shows the basic properties of each dataset. The continuous variables were discretized using five equal frequency intervals.

To test the performance of the methods, we computed the following four performance measures. The CLL of the model in the test dataset indicates how well each MBC fits the conditional distribution in the data. Note that this measure is not available for CDL2 and SVM–struct.

The mean accuracy of the classifiers averages the accuracy values of all the class variables individually, as described below for M samples and d classes:

$$\text{acc}_M = \frac{1}{d \cdot M} \sum_{i=1}^{d} \sum_{j=1}^{M} \delta(c'_{ij}, c_{ij}) \ , \tag{5.1}$$

where c'_{ij} represents the predicted class label for variable C_j in instance i, c_{ij} is its true value, and $\delta(c'_{ij}, c_{ij}) = 1$ if $c'_{ij} = c_{ij}$, and 0 otherwise.

The global accuracy measures the fraction of instances in which the labels of all the classes were correctly assigned, and is given by:

$$\text{acc}_G = \frac{1}{M} \sum_{i=1}^{M} \delta(\mathbf{c}'_i, \mathbf{c}_i) \ , \tag{5.2}$$

where $\delta(\mathbf{c}'_i, \mathbf{c}_i) = 1$ if $\mathbf{c}'_i = \mathbf{c}_i$, and 0 otherwise.

Finally, the learning times (time) required by each method are also compared.

We analyzed the significance of the differences found for each performance measure in all experiments using the Friedman test with $\alpha = 0.05$ and Holm's [Holm, 1979] and Shaffer's [Shaffer, 1986] post-hoc procedures. Both methods are explained in Section 3.3.1.

Experiments were performed on a computer with an Intel Core i7-6700K CPU at 4.00 GHz with 16 GB main memory, running Ubuntu 16.04 LTS. The MBC learning methods were written in Python 2.7.12, and integrate specific functions developed in C++11 (version 5.4.0). We used the implementation of CDL2 provided by MEKA [Read et al., 2016] with the scikit-multilearn [Szymański and Kajdanowicz, 2017] Python wrapper. We used PyStruct library [Müller and Behnke, 2014] to perform SVM–struct. We set the input parameters of methods CDL2 and SVM–struct to their default values.

5.3.1 Comparison with unbounded methods

Some of the compared methods were either very slow or returned intractable models when applied in large datasets. Therefore, in this subsection we only show the experiments performed in the first nine datasets datasets in Table 5.2. The rest of the datasets are used in Section 5.3.2. Additionally, we filtered the 200 features with higher information gain among all the class variables in the datasets with more than 200 features to reduce their dimensionality. We set the treewidth bound to five for methods GS–pruned, GS–tw and Tree–tw. In our experience, this bound provides a good trade-off between expressiveness and tractability. Note that in GS–pruned, we bound the treewidth of the pruned graph rather than the treewidth of the complete graph. Table 5.3 displays the mean rank ± the standard deviation of each method over all the datasets for each performance measure. Note that the fastest method is ranked the first in learning time. The mean treewidth (tw) and the mean treewidth of the pruned graph (tw–pr) are also given. The detailed results can be found at https://github.com/marcobb8/tr_bn/blob/master/Supplementary%20Material/supplemantary-material_MBCs_unb.pdf.

Figures 5.2–5.5 graphically present the results obtained with Holm's and Shaffer's post-hoc procedures for each performance measure in all datasets. Note that both procedures produced identical results for CLL and acc_G. In Figures 5.4 and 5.5, the differences between the results output by both methods were explained in the corresponding caption.

GS–MBC and GS–pruned performed the best according to the first three evaluation metrics. They are the less restrictive methods, and the models returned by both approaches

Table 5.3: Comparison of all the methods in the first nine datasets datasets in Table 5.2. The optimal results are denoted in boldface.

	GS-pruned	Tree-tree	Tree-tw	GS–MBC	GS-tw	CDL2	SVM-struct
CLL	2.06±0.73	3.76±1.05	4.56±1.16	**2.02±0.83**	2.61±0.71	6.5±0	6.5±0
acc_G	2.49±1.11	3.61±1.69	4.86±1.49	**2.47±1.07**	2.99±0.94	6.86±0.36	4.73±1.52
acc_M	**2.83±1.11**	3.04±1.90	5.22±1.14	2.83±1.02	3.18±0.86	6.93±0.33	3.96±2.09
time	3.24±0.93	5.82±0.39	**1.42±0.97**	2.67±1.02	3.78±1.17	4.07±1.56	7±0
tw (mean)	14.36±18.19	8.91±5.23	3.22±1.04	13.96±17.49	4.33±1	–	–
tw-pr (mean)	3.71±1.20	3.47±1.78	3.16±1.09	3.98±1.79	3.64±1.23	–	–

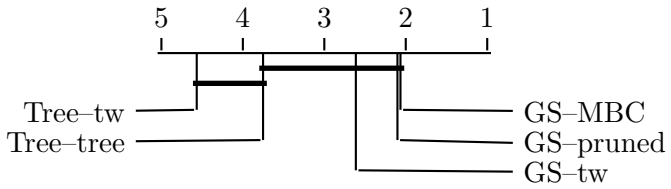


Figure 5.2: Comparison of CLL for all methods against each other with the Holm's and Shaffer's post-hoc test for the experimental results shown in Table 5.3.

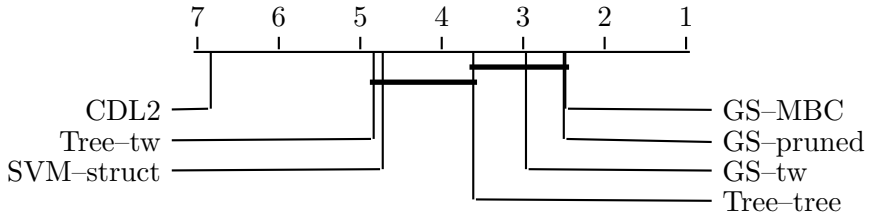


Figure 5.3: Comparison of acc_G for all methods against each other with the Holm's and Shaffer's post-hoc test for the experimental results shown in Table 5.3.

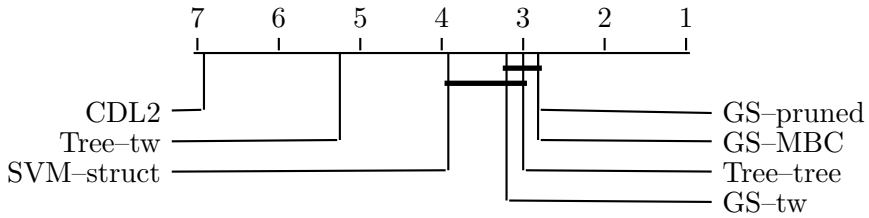


Figure 5.4: Comparison of acc_M for all methods against each other with the Holm's post-hoc test for the experimental results shown in Table 5.3. Shaffer's procedure did not find significant differences between Tree–tree and SVM–struct.

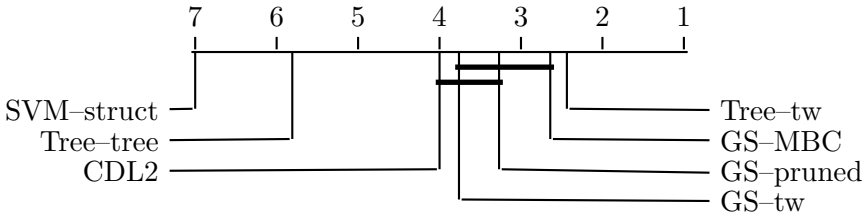


Figure 5.5: Comparison of learning time for all methods against each other with the Holm's and Shaffer's post-hoc tests for the experimental results shown in Table 5.3. Shaffer's procedure did not find significant differences between Tree–tw and GS-MBC and between Tree–tree and SVM–struct.

are very similar in most of the experiments, given that GS–MBC only surpasses treewidth five (of the pruned graph) in a few cases. They are followed by GS–tw, that is ranked the third for CLL and acc_G, and Tree–tree, that performs better than GS–tw for acc_M. Given that the restrictions of Tree-tree directly apply to the class subgraph, this method was expected to perform better for acc_M than for CLL and acc_G. No significant differences were found between the above methods.

Tree–tw was the worst performing MBC learning method, and there were significant differences between this method and GS–MBC, GS–pruned and GS–tw for CLL, acc_G and acc_M. SVM–struct obtained slightly better results than Tree-tw for acc_G and acc_M. As Tree–tw, SVM–struct represents the relationships among the class variables using a tree, and does not explicitly represent the relationships among the features, but the use of discriminative models (SVMs) as base classifiers seems to improve the performance, specially for acc_M. Finally, CDL2 performs significantly worse than the rest of the methods for acc_G and acc_M.

Tree–tw was the fastest method in most of the experiments. GS–MBC, GS–pruned and GS–tw required a similar amount of computations. Although Tree–tree required more learning time than the latter methods, this is mainly caused by the Python implementation of the conditional information gain. Most of the times, the computational cost of Tree–tree should be lower than the cost of GS–MBC, GS–pruned and GS–tw. SVM–struct was by far the slowest method in all the experiments.

Overall, the experiments suggest that the least restrictive methods tend to perform better than the most restrictive ones.

5.3.2 Comparison of treewidth bounded methods

In this subsection we compare the methods GS–pruned, GS–tw and Tree–tw in all the datasets (see Table 5.2). These methods ensure the tractability of the output models and performed efficiently in all the datasets. Therefore, the feature variables of the large datasets were not filtered beforehand. The experiments were performed using varied treewidth bounds, from two to five. Table 5.3 compares the results obtained with the above approaches in all datasets. The detailed results are available at https://github.com/marcobb8/tr_bn/blob/master/Supplementary%20Material/supplemantary-material_MBCs_b.pdf.

Table 5.4: Comparison of methods for learning tractable MBCs for the datasets of Table 5.2. The optimal results are denoted in boldface.

	GS-pruned	Tree-tw	GS-tw	tw bound
CLL	**1.19±0.49**	2.87±0.5	1.94±0.28	2
acc_G	**1.52±0.76**	2.49±0.73	1.99±0.52	
acc_M	**1.29±0.58**	2.61±0.61	2.09±0.52	
time	2.61±0.54	**1.57±0.89**	1.81±0.59	
tw (mean)	65.79±85.89	2.09±0.47	2±0	
tw-pr (mean)	1.99±0.12	2.09±0.47	2±0	
CLL	**1.23±0.46**	2.89±0.45	1.88±0.38	3
acc_G	**1.6±0.67**	2.61±0.69	1.79±0.59	
acc_M	**1.41±0.65**	2.72±0.58	1.87±0.45	
time	2.43±0.6	**1.53±0.89**	2.04±0.69	
tw (mean)	65.24±85.04	2.97±0.61	2.95±0.23	
tw-pr (mean)	2.91±0.29	2.93±0.64	2.89±0.31	
CLL	**1.35±0.49**	2.89±0.45	1.75±0.45	4
acc_G	**1.57±0.56**	2.72±0.64	1.71±0.55	
acc_M	**1.42±0.59**	2.77±0.54	1.81±0.43	
time	2.41±0.55	**1.53±0.89**	2.05±0.73	
tw (mean)	65.68±85.01	3.63±0.91	3.83±0.5	
tw-pr (mean)	3.63±0.65	3.59±0.96	3.6±0.7	
CLL	**1.47±0.48**	2.89±0.45	1.63±0.46	5
acc_G	**1.54±0.53**	2.73±0.63	1.73±0.52	
acc_M	**1.47±0.53**	2.79±0.56	1.74±0.44	
time	2.43±0.57	**1.47±0.81**	2.11±0.75	
tw (mean)	66.24±86.17	4.15±1.28	4.6±0.82	
tw-pr (mean)	4.25±1.07	4.08±1.32	4.25±1.13	

The significant differences found by Holm's and Shaffer's procedures are shown in Figures 5.6–5.9. Both procedures produced identical results for all the evaluation metrics.

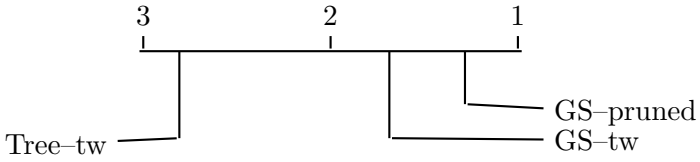


Figure 5.6: Comparison of CLL in test dataset with Holm's and Shaffer's post-hoc procedures for the experimental results shown in Table 5.4.

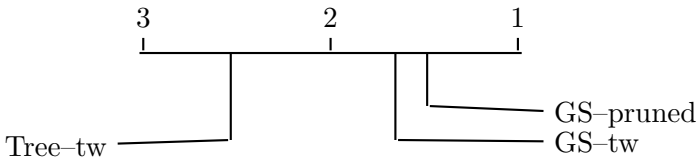


Figure 5.7: Comparison of acc_G with Holm's and Shaffer's post-hoc procedures for the experimental results shown in Table 5.4.

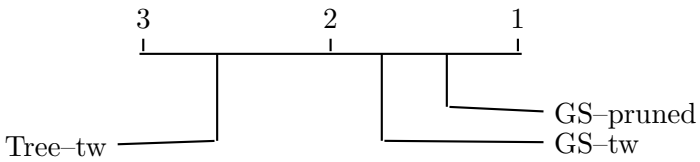


Figure 5.8: Comparison of acc_M with Holm's and Shaffer's post-hoc procedures for the experimental results shown in Table 5.4.

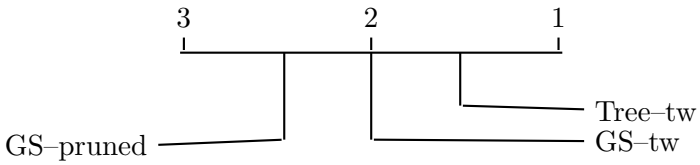


Figure 5.9: Comparison of learning time with Holm's and Shaffer's post-hoc procedures for the experimental results shown in Table 5.4.

GS–pruned performed the best according to all the evaluation metrics followed by GS–tw, and Tree–tw. Holm's and Shaffer's procedures found significant differences between all the methods. The treewidth of the output models was clearly higher for GS–pruned than for GS–tw, allowing the former to fit better the data. As expected, the difference between the results obtained with both methods was greater when the treewidth bound was smaller. This suggests that bounding the pruned graph (as GS–pruned does) instead of the complete graph may help improving the performance of the models when the relationships allowed by an MBC of bounded treewidth are not expressive enough.

Finally, Tree–tw was the fastest in most cases. Although GS–tw required less learning time than GS–pruned in most of the experiments, the differences in learning time for each particular experiment were small.

5.4 Conclusions

In this chapter we addressed the problem of the complexity of multidimensional classification in MBCs. We provided theoretical upper bounds for the complexity of these models, and we proved that the complexity of the queries that are usually performed in MBCs is bounded by the treewidth of the pruned graph. The treewidth of the pruned graph may be small even if the treewidth of the complete structure is high. We proposed a learning method that uses the above properties to ensure such tractability.

Experimental results showed that the proposed method is competitive with other state-of-the-art methods in terms of fitting to the conditional distribution of the data and accuracy. We also observed that some models remain tractable even with a large treewidth. By setting a bound on the pruned graph instead of on the complete graph we can learn more expressive models with the same inference complexity. The experiments show that this is specially useful for lower treewidth bounds.

In the future we are interested in addressing the problem of learning MBCs from incomplete datasets.

Chapter 6

Discriminative learning of multidimensional Bayesian network classifiers

Discriminative methods usually outperform generative methods in the task of classification [Friedman et al., 1997; Grossman and Domingos, 2004]. One of the reasons may be that, while the latter try to fit the joint probability distribution of the data, the former focus only on fitting the conditional distribution of the class variables given the features.

In multidimensional classification, correctly identifying the relationships among the class variables is essential to provide accurate predictions. MBCs are specially well suited for this task, given that they explicitly represent these relationships. Moreover, they give access to the posterior distribution of the class variables conditional on the features instead of providing only point estimates.

To the best of our knowledge, there is little work in the literature addressing the problem of learning MBCs from data in a discriminative way. Although adapting discriminative parameter learning to MBCs is straightforward, there is no evidence of whether this leads to better performance or not. There are some wrapper structure learning methods that optimize the cross-validated accuracy of MBCs [Bielza et al., 2011; Borchani et al., 2010], but they are extremely computationally demanding in practice.

In this chapter, we propose strategies for learning the parameters and the structure of MBCs by optimizing the CLL. As computing CLL requires performing inference, we use the methods described in Chapter 3 to bound the treewidth of the models during the learning process. The rest of this chapter is organized as follows. Section 6.1 describes in detail how to optimize the CLL of the parameters. Section 6.2 contains our proposal for learning the structure of MBCs. Section 6.3 discusses the experimental results. Finally, Section 6.4 gives our conclusions and suggests future research lines.

This chapter contains work in progress, that will be submitted as Benjumeda et al. [2019b].

6.1 Parameter learning

A common way to optimize the CLL of the parameters of BN classifiers is to perform gradient ascent [Greiner et al., 2005; Roos et al., 2005; Zaidi et al., 2017]. This idea can be directly applied to learn the parameters of MBCs. The main problem of gradient ascent is knowing how to compute the gradient of the objective function. From Equation (2.8), given the initial BN $\mathcal{B} = (\mathcal{G}, \boldsymbol{\theta})$ and a dataset $\mathcal{D}$, the gradient of the CLL is given by:

$$\frac{\partial \mathrm{CLL}(\mathcal{B}|\mathcal{D})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} = \frac{\partial \ell(\mathcal{B}|\mathcal{D})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} - \frac{\partial \ell_F(\mathcal{B}|\mathcal{D})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}}, \tag{6.1}$$

where $\boldsymbol{\theta}_{x|\boldsymbol{u}}$ is the parameter that represents $\mathcal{P}(X = x|\mathbf{Pa}_X^G = \boldsymbol{u})$.

The first term of this equation is the gradient of the log-likelihood function, which is given by:

$$\begin{aligned}\frac{\partial \ell(\mathcal{B}|\mathcal{D})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} &= \sum_{m=1}^{M} \frac{\partial \log \mathcal{P}(\boldsymbol{c}[m], \boldsymbol{f}[m]|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} = \sum_{m=1}^{M} \left(\frac{1}{\mathcal{P}(\boldsymbol{c}[m], \boldsymbol{f}[m]|\boldsymbol{\theta})}\right) \frac{\partial \mathcal{P}(\boldsymbol{c}[m], \boldsymbol{f}[m]|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} \\ &= \frac{M_{x,\boldsymbol{u}}}{\boldsymbol{\theta}_{x|\boldsymbol{u}}}, \text{ when } \boldsymbol{\theta}_{x|\boldsymbol{u}} \neq 0,\end{aligned} \tag{6.2}$$

where $M_{x,\boldsymbol{u}}$ are the number of instances in $\mathcal{D}$ where X takes value x and $\mathbf{Pa}_x^{\mathcal{B}}$ take values $\boldsymbol{u}$. Note that $\frac{\partial \mathcal{P}(\boldsymbol{c}[m], \boldsymbol{f}[m]|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} = 0$ when X and $\mathbf{Pa}_x^{\mathcal{B}}$ do not take values x and $\boldsymbol{u}$, respectively, and $\frac{\partial \mathcal{P}(\boldsymbol{c}[m], \boldsymbol{f}[m]|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} = \frac{\mathcal{P}(\boldsymbol{c}[m], \boldsymbol{f}[m]|\boldsymbol{\theta})}{\boldsymbol{\theta}_{x|\boldsymbol{u}}}$ when X and $\mathbf{Pa}_x^{\mathcal{B}}$ take values x and $\boldsymbol{u}$, respectively, and $\boldsymbol{\theta}_{x|\boldsymbol{u}} \neq 0$.

The second term of Equation (6.1) can be computed as follows:

$$\begin{aligned}\frac{\partial \ell_F(\mathcal{B}|\mathcal{D})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} &= \sum_{m=1}^{M} \frac{\partial \log \mathcal{P}(\boldsymbol{f}[m]|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} = \sum_{m=1}^{M} \left(\frac{1}{\mathcal{P}(\boldsymbol{f}[m]|\boldsymbol{\theta})}\right) \frac{\partial \mathcal{P}(\boldsymbol{f}[m]|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} \\ &= \frac{1}{\boldsymbol{\theta}_{x|\boldsymbol{u}}} \sum_{m=1}^{M} \mathcal{P}(x, \boldsymbol{u}|\boldsymbol{f}[m], \boldsymbol{\theta}), \text{ when } \boldsymbol{\theta}_{x|\boldsymbol{u}} \neq 0.\end{aligned} \tag{6.3}$$

Although Equation (6.2) has closed form, Equation (6.3) requires performing inference for each instance in the dataset. Well-known inference algorithms as JT belief propagation allow computing the partial derivatives with respect to all the network parameters at the same time. In these cases, the cost of evaluating Equation (6.1) is linear in M and exponential in the treewidth of $\mathcal{G}$. Therefore, if the treewidth of $\mathcal{G}$ is small, the gradient of the CLL can be computed efficiently.

Gradient ascent cannot be directly applied, given that the parameter estimation is a constrained optimization problem, where $\boldsymbol{\theta}_{x|\boldsymbol{u}} \in [0, 1]$ and $\sum_{x_i \in \Omega_X} \boldsymbol{\theta}_{x_i|\boldsymbol{u}} = 1$. The same issue has been studied for applying gradient ascent to optimize the LL in incomplete datasets [Darwiche, 2009; Koller and Friedman, 2009]. We adopt a popular strategy that consists of

performing the search in the soft-max space [Bishop, 1995]. In this case, the parameters are defined with respect to the soft-max variables:

$$\boldsymbol{\theta}_{x|\boldsymbol{u}} = \frac{e^{s_{x|\boldsymbol{u}}}}{\sum_{x_i \in \Omega_X} e^{s_{x_i|\boldsymbol{u}}}},$$

where $s_{x|\boldsymbol{u}}$ is the soft-max variable associated with $\boldsymbol{\theta}_{x|\boldsymbol{u}}$. For any choice of the soft-max variables, the parameters fulfill the constraints mentioned above. The gradient of the CLL with respect to $s_{x|\boldsymbol{u}}$ is

$$\frac{\partial \mathrm{CLL}(\mathcal{D}|\boldsymbol{\theta})}{\partial s_{x|\boldsymbol{u}}} = \frac{\partial \mathrm{CLL}(\mathcal{D}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} \frac{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}}{\partial s_{x|\boldsymbol{u}}} = \frac{\partial \mathrm{CLL}(\mathcal{D}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_{x|\boldsymbol{u}}} \frac{e^{s_{x|\boldsymbol{u}}} \sum_{x_i \in (\Omega_X \setminus \{x\})} e^{s_{x_i|\boldsymbol{u}}}}{(\sum_{x_i \in \Omega_X} e^{s_{x_i|\boldsymbol{u}}})^2}. \tag{6.4}$$

Equation (6.4) can be combined with multiple optimization methods. A classic approach is to use conjugate gradient [Bishop, 1995]. Instead, we use the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L–BFGS) algorithm [Nocedal, 1980; Byrd et al., 1994]. L–BFGS is a quasi-Newton optimization method that, to guide the search, estimates an approximation of the inverse Hessian matrix using a limited amount of memory. While the computational overhead of L–BFGS in a single iteration is higher than in conjugate gradient, the former requires fewer steps until convergence and fewer evaluations of the objective function than the latter [Malouf, 2002]. This is specially useful when learning the parameters of MBCs, where the evaluation of the CLL is usually the bottleneck.

6.2 Structure learning

This section describes the proposed MBC structure learning algorithm. Our approach is inspired by the BNC algorithm [Grossman and Domingos, 2004], which learns BN classifiers in a discriminative manner. Like BNC, we estimate the parameters via ML during the structure search, given that directly optimizing the CLL would be intractable in most cases. The scoring function used by BNC to guide the structure search is the CLL of each model (both structure and parameters), which requires computing the joint probability of the features for each instance in the dataset. When the dataset is complete, this can be done by marginalizing out a single variable (the class variable).

In multidimensional classification, it is necessary to marginalize over multiple class variables to obtain the joint probability of the features. These operations can be performed efficiently when the treewidth of the MBC structure $\mathcal{G}$ is bounded. In this case, the complexity of computing the CLL is exponential in the treewidth of $\mathcal{G}$ and is linear in M. We use the methods described in Chapter 3 to bound the inference complexity of the models. As directly optimizing the CLL of the MBCs would result in very dense structures, we penalize the complexity of the models to avoid overfitting. Specifically, we assume that the scoring function is of the form:

$$\mathrm{score}_{\mathrm{CLL}}(\mathcal{D}, \mathcal{B}) = \mathrm{CLL}(B|\mathcal{D}) - \mathrm{pen}(\mathcal{B}, \mathcal{D}), \tag{6.5}$$

where $\text{pen}(\mathcal{B}, \mathcal{D})$ is a penalty function that can depend on $\mathcal{B}$ and $\mathcal{D}$.

Even for bounded treewidth models, the amount of computations required to compute the CLL of each MBC candidate during the learning process may be too high. Instead, Algorithm 6.1, called discriminative greedy search (DGS), computes the CLL of the most promising candidates to speed-up the process. At each iteration, the MBC candidates are ordered from the best to the worst according to a generative score, and the first model that produces an increment in the discriminative score (Equation (6.5)) is selected. This process is repeated until no change improves the discriminative score. The objective of this procedure is to take advantage of the efficiency of generative methods, avoiding some of their drawbacks. Specifically, a drawback is that many local changes that improve the generative scoring function may not improve, or even worsen, the model from a discriminative perspective. For example, it is common to add arcs from class variables to redundant features. The proposed procedure ensures that each local change applied during the structure search improves the discriminative score. Note that, after learning the structure of the MBC, we may optimize the CLL of the parameters by using the method described in Section 6.1.

Algorithm 6.1 performs as follows: line 2 compiles a valid ET for the given BN. Line 3 loops until no local changes improves the current MBC candidate. In line 4, the local changes that fulfill the restrictions at lines 5, 6 and 7 are obtained. These restrictions avoid arcs that have no effect on the CLL of the model and arcs from the features to the class variables. In line 8, the local changes are sorted in a descending order according to a generative score. The loop at line 12 searches for the first MBC of bounded treewidth (line 15) that impoves the discriminative score (line 16). Theorem 6.1 shows that each iteration of Algorithm 6.1 can be computed efficiently.

Theorem 6.1. *If the treewidth bound t_b is a constant (which is assumed to be small), then each iteration of Algorithm 6.1 consumes polynomial time in the number of variables (n) and number of instances (M) in dataset $\mathcal{D}$.*

Proof. Algorithm 3.1 (line 2) requires polynomial time in n (Proposition 3.1). At line 4, there are a maximum of $\mathcal{O}(n^2)$ local changes. Therefore, the scores of all the candidates (line 8) can be computed in polynomial time in n and M. The loop at line 12 performs at worst $\mathcal{O}(n^2)$ steps (i.e., one for each possible local change). Algorithms 3.2, 3.3 and 3.5 (line 14) take polynomial time in n (Theorem 3.2), and the width of an EO (line 15) can be obtained efficiently [Darwiche, 2009]. Finally, the CLL of an MBC of bounded treewidth (line 16) can be computed in polynomial time in n and M. □

6.3 Experimental results

In this section we empirically analyze the strategies proposed in this chapter for learning MBCs. We compare DGS (Algorithm 6.1) with the methods proposed in Chapter 5, GS–pruned and GS–tw. A comparison of the latter approaches with other methods in the state of the art is provided in Section 5.3. DGS–clp and GS–clp use the structures output by DGS

Input: Dataset $\mathcal{D}$, treewidth bound t_b
Output: Best MBC $\mathcal{B}^*$

initialize MBC $\mathcal{B}^*$ and EO π^*;
let $\mathcal{E}^*_{\mathcal{B}^*}$ be a valid ET that represents π^* for MBC $\mathcal{B}^*$ (Algorithm 3.1);
loop for $j = 0, 1, \ldots$ until convergence
  let $\mathrm{s}_1, \ldots, \mathrm{s}_l$ be the local changes (i.e., arc additions, removals, or reversals) that can be applied to $\mathcal{G}^*$ (the structure of $\mathcal{B}^*$) such that:
  a) local change s_d is arc addition $X_{\text{out}} \rightarrow X_{\text{in}}$ and $X_{\text{out}} \in \mathcal{C}$
  b) If local change s_d is arc addition $X_{\text{out}} \rightarrow X_{\text{in}}$ and $X_{\text{out}} \in \mathcal{F}$, then $X_{\text{in}} \in \mathcal{F}$ and $\mathbf{Pa}^{\mathcal{G}^*}_{X_{\text{in}}} \cap \mathcal{C} \neq \emptyset$
  c) If local change s_d is arc reversal $X_{\text{out}} \rightarrow X_{\text{in}}$, then $X_{\text{out}} \in \mathcal{F}$, $X_{\text{in}} \in \mathcal{F}$, and $\mathbf{Pa}^{\mathcal{G}^*}_{X_{\text{out}}} \cap \mathcal{C} \neq \emptyset$
  let $\mathrm{s}_{o(1)}, \ldots, \mathrm{s}_{o(l)}$ be the result of sorting in descending order $\mathrm{s}_1, \ldots, \mathrm{s}_l$ according to $\text{score}(\mathcal{D}, \mathcal{B}^d)$, where $\mathcal{B}^d$ is the result of applying local change s_d to $\mathcal{B}^*$;
  improve $\leftarrow$ FALSE;
  $d \leftarrow 1$;
  let $\mathcal{B}^j$ and $\mathcal{E}^j_{\mathcal{B}j}$ be a copy of $\mathcal{B}^*$ and $\mathcal{E}^*_{\mathcal{B}^*}$, respectively;
  while $d \leq l$ **and not** improve **do**
    let $\mathcal{B}'$ be the result of applying local change $\mathrm{s}_{o(d)}$ in $\mathcal{B}^*$;
    let $\mathcal{E}'_{\mathcal{B}'}$ be the result of compiling local change $\mathrm{s}_{o(d)}$ in $\mathcal{E}^*_{\mathcal{B}^*}$ (Algorithms 3.2 and 3.3), and optimizing the resulting ET (Algorithm 3.5);
    if $\text{width}(\mathcal{E}'_{\mathcal{B}'}) \leq t_b$ **then**
      if $\text{score}_{\text{CLL}}(\mathcal{D}, \mathcal{B}') > \text{score}_{\text{CLL}}(\mathcal{D}, \mathcal{B}^j)$ **then**
        $\mathcal{B}^j, \mathcal{E}^j_{\mathcal{B}j} \leftarrow \mathcal{B}', \mathcal{E}'_{\mathcal{B}'}$;
        improve $\leftarrow$ TRUE;
      end
    end
    increment d;
  end
  $\mathcal{B}^*, \mathcal{E}^*_{\mathcal{B}^*} \leftarrow \mathcal{B}^j, \mathcal{E}^j_{\mathcal{B}j}$;

Algorithm 6.1: Pseudocode of discriminative greedy search of MBCs (DGS).

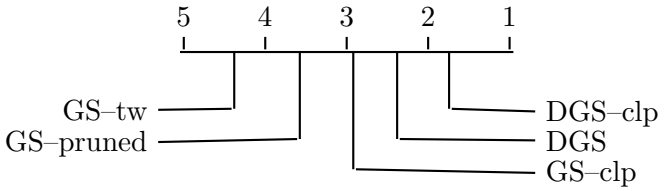


Figure 6.1: Comparison of CLL for all methods against each other with the Holm's and Shaffer's post-hoc tests for the experimental results shown in Table 6.1.

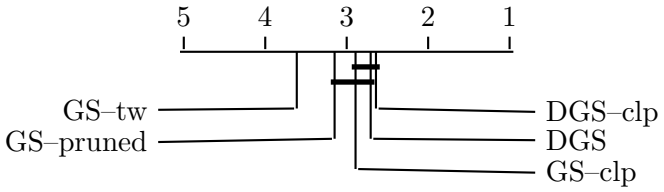


Figure 6.2: Comparison of acc_G for all methods against each other with the Holm's and Shaffer's post-hoc tests for the experimental results shown in Table 6.1.

and GS–tw, respectively, but they optimize the CLL, as described in Section 6.1, instead of the LL to learn the parameters of the MBC. In all the cases, BIC was the generative score to optimize. We also used BIC's penalization on the number of parameters as the complexity penalization of the discriminative score (Equation (6.5)).

We learned models from the 15 multilabel datasets described in Table 5.2, using 5-fold cross-validation. We applied all methods using different treewidth bounds (2, 3, 4 and 5). The continuous variables were discretized using five equal frequency intervals. To evaluate the models, we used the CLL in the test dataset, acc_G and acc_M (described in Section 5.3).

We used the Friedman test with $\alpha = 0.05$ and Holm's [Holm, 1979] and Shaffer's [Shaffer, 1986] post-hoc procedures to analyze the significance of the differences found for each performance measure. Both methods are described in Section 3.3.1.

Experiments were performed on a computer with an Intel Core i7-6700K CPU at 4.00 GHz with 16 GB main memory, running Ubuntu 16.04 LTS. All the compared methods were written in Python 2.7.12, and integrate specific functions developed in C++11 (version 5.4.0).

6.3.1 Comparison with generative methods

Table 6.1 shows the mean rank (± the standard deviation) of each MBC learning method in the datasets described in Table 5.2 for the different treewidth bounds, according to the CLL, acc_G, acc_M, and the learning time. The mean treewidth, and the mean treewidth of the pruned graph (see Section 5.1) are also shown. The detailed experimental results are available online at https://github.com/marcobb8/tr_bn/blob/master/Supplementary%20Material/supplemantary-material_MBCs_cll.pdf.

Figures 6.1–6.4 show the significant differences among the MBC learning methods for each evaluation metric.

Table 6.1: Comparison of discriminative and generative MBC learning methods. The optimal results are denoted in boldface.

	DGS	DGS-clp	GS-tw	GS-clp	GS-pruned	tw bound
CLL	2.13±0.84	**1.55±1.07**	4.61±0.96	3.31±0.94	3.41±0.81	2
acc_G	**2.58±1.05**	2.58±1.21	3.95±1.23	2.9±1.19	2.99±1.42	
acc_M	2.62±0.94	**2.04±1.09**	4.34±1.03	3.01±1.39	2.99±1.21	
time	3.94±1.03	3.94±1.03	**1.89±0.88**	1.89±0.88	3.35±1.01	
tw (mean)	1.89±0.31	1.89±0.31	2±0	2±0	65.79±85.89	
tw-pr (mean)	1.8±0.43	1.8±0.43	2±0	2±0	1.99±0.12	
CLL	2.4±0.97	**1.72±1.13**	4.49±1.02	2.95±1.17	3.44±0.99	3
acc_G	2.9±1.29	**2.77±1.25**	3.47±1.39	2.91±1.2	2.96±1.43	
acc_M	2.67±1.13	**2.28±1.23**	4.13±1.21	2.81±1.33	3.11±1.27	
time	3.59±1.35	3.59±1.35	**2.45±1.14**	2.45±1.14	2.92±1.25	
tw (mean)	2.65±0.67	2.65±0.67	2.95±0.23	2.95±0.23	65.24±85.04	
tw-pr (mean)	2.52±0.79	2.52±0.79	2.89±0.31	2.89±0.31	2.91±0.29	
CLL	2.48±1.07	**1.79±1.15**	4.3±0.98	2.76±1.17	3.67±1.11	4
acc_G	**2.71±1.3**	2.71±1.33	3.43±1.35	3.04±1.35	3.1±1.23	
acc_M	2.76±1.13	**2.29±1.28**	4.09±1.19	2.72±1.36	3.15±1.23	
time	3.65±1.32	3.65±1.32	**2.41±1.13**	2.41±1.13	2.89±1.23	
tw (mean)	3.29±1.06	3.29±1.06	3.83±0.5	3.83±0.5	65.68±85.01	
tw-pr (mean)	3.16±1.22	3.16±1.22	3.6±0.7	3.6±0.7	3.63±0.65	
CLL	2.48±1.12	**1.83±1.19**	4.11±1.15	2.71±1.15	3.87±0.94	5
acc_G	2.85±1.34	2.79±1.37	3.42±1.3	**2.77±1.27**	3.17±1.27	
acc_M	2.76±1.18	**2.13±1.25**	3.96±1.18	2.91±1.34	3.25±1.22	
time	3.55±1.32	3.55±1.32	**2.54±1.29**	2.54±1.29	2.81±1.1	
tw (mean)	3.88±1.49	3.88±1.49	4.6±0.82	4.6±0.82	66.24±86.17	
tw-pr (mean)	3.67±1.61	3.67±1.61	4.25±1.13	4.25±1.13	4.25±1.07	

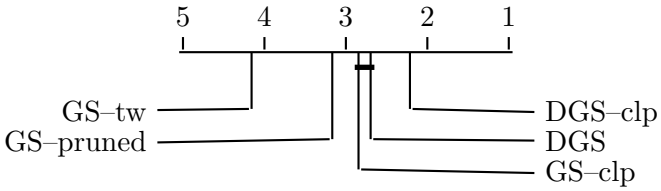


Figure 6.3: Comparison of acc_M for all methods against each other with the Holm's post-hoc test for the experimental results shown in Table 6.1. Shaffer's procedure did not find significant differences between GS–clp and GS–pruned.

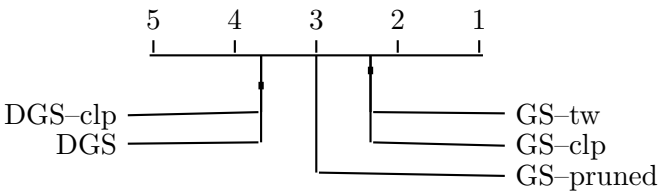


Figure 6.4: Comparison of learning time for all methods against each other with the Holm's and Shaffer's post-hoc tests for the experimental results shown in Table 6.1.

The proposed strategies for the discriminative learning of the structure and parameters of MBCs led to higher CLL and accuracy. Both DGS and GS–clp performed better than GS–tw and GS–pruned, and DGS–clp was the best performing method. Overall, DGS outperformed GS–clp, which suggests that optimizing the CLL of the structure is more relevant than optimizing only the CLL of the parameters.

DGS–clp performed significantly better than GS–tw and GS–pruned according to the first three evaluation metrics, and better than DGS and GS–clp for the CLL and acc_M. All the methods performed similarly in terms of general accuracy, with the exception of GS–tw, that obtained the worst results.

GS–clp performed clearly better for higher treewidth bounds. For example, GS–clp and GS–pruned obtained similar results for treewidth bound 2, while GS–clp was closer to DGS for bound 5. This suggests that dense structures favor GS–clp. Moreover, computing L–BFGS to estimate the parameters took less than a minute in the worst case, while DGS took a mean of 6.28 times the learning time required by GS–tw. As the difference in learning time scales with the size of the dataset, performing DGS in even larger datasets may be prohibitive. In these cases, GS–clp can be a good alternative.

6.4 Conclusions

In this chapter we proposed a new method for learning MBCs. To the best of our knowledge, this is the first MBC learning method where the learning process is guided by the CLL of the models. Additionally, we demonstrated that each iteration of the new method takes polynomial time in the number of variables and the size of the dataset. The keys to allow the method to scale up to large domains were: first, to bound the treewidth of all the MBC candidates, and second, to use a generative scoring function to order the candidates from the most to the least promising. We also provided a strategy to optimize the CLL of the MBC parameters.

We performed extensive experiments to test the performance of our proposal in a wide variety of multilabel datasets. Both the discriminative structure and parameter learning methods supposed an improvement in the performance compared to the generative methods from Chapter 5. Nevertheless, the proposed structure learning method was significantly slower than its generative counterpart, which could be a problem if we aim to learn MBCs from even larger domains. In these cases, the discriminative learning of the parameters may suffice, given that its combination with a generative structure learning method proved competitive with the rest of the methods, requiring clearly less computations.

When data is complete, the CLL of the MBCs can be computed efficiently even when their treewidth is not small if the treewidth of their pruned graph is bounded. Therefore, using the latter to limit the complexity of the MBCs may lead to learning more expressive models. Additionally, we are interested in combining this work with our proposal in Chapter 4 to learn MBCs from incomplete datasets. Finally, we intend to study if it is possible to extend the work of Carvalho et al. [2011] to approximate the CLL of MBCs.

Chapter 7

Multidimensional Bayesian network classifiers to predict individual temporal lobe epilepsy patient surgical outcomes

Temporal lobe epilepsy (TLE) is the most common form of focal epilepsy in adults, with drug-resistance developing in approximately a third of patients [Semah et al., 1998]. Drug-resistant TLE is potentially curable via surgery. Overall, rates of seizure-freedom after one year of post-surgical follow-up varies from 53 to 84% [Spencer and Huh, 2008]. Even the so called "best surgical candidates" can develop rather suboptimal post-surgical outcomes, in particular during long-term follow-up, with seizure recurrence reaching 40-50% ten years after surgery [Spencer, 1996; McIntosh et al., 2004]. Meaningful improvement in the quality of life is primarily observed only in patients who achieve seizure-freedom after surgery [Markand et al., 2000]. A tool for predicting both short and medium-term surgical success would hence be immensely useful for both physicians and patients.

Although several predictors of seizure freedom after TLE surgery have been identified, combining these predictors via multivariate logistic regression modeling has only achieved a modest discriminative ability [Uijl et al., 2008]. A nomogram and seizure freedom score have independently been developed to predict success of epilepsy surgery in general [Jehi et al., 2015; Garcia-Gracia et al., 2015], but there is no clinical prediction model for patients undergoing TLE surgery in particular.

We therefore aimed to use a supervised machine learning approach to improve surgical outcome predictions based on clinical, neurophysiological and imaging features retrospectively collected from a TLE surgical dataset at the University of California, San Francisco (UCSF) over a fifteen year period. Specifically, we learned an MBC from the UCSF dataset. We further tested this model on a separate surgical TLE cohort treated at the Montreal Neurological Institute (MNI) to validate the learning model, and compared its performance

with established nomograms.

This chapter includes the content of Benjumeda et al. [2019d].

7.1 Methods

7.1.1 Study design, participants, and procedures

We retrospectively studied a cohort of 231 consecutive TLE patients who underwent surgical treatment at UCSF (M=167) and at the MNI (M=64) between years 2000-2015. All patients fulfilled the International League Against Epilepsy definition of drug-resistant epilepsy [Kwan et al., 2010]. Adults were included if (i) unilateral temporal lobe seizure onset was demonstrated during scalp and/or intracranial electroencephalography (EEG) monitoring, (ii) pre-surgical 1.5 or 3.0 Tesla MRIs revealed no lesions, or showed unilateral hippocampal atrophy, (iii) at least one year of post-surgical follow-up was available. To increase the homogeneity of the selected study population, patients with other overt epileptogenic abnormalities in the temporal lobes on imaging (such as cortical dysplasias, neoplasms or cavernomas) were excluded. Prior to their surgery, all patients were discussed at a pre-surgical team conference, during which their seizure history, clinical findings, video EEG and neuroimaging scans were reviewed, and intracranial EEG implantation and/or surgical strategies planned. This study was approved by the research ethics committee from both institutions.

Patients' demographics, clinical, neurophysiological, and imaging variables were collected, including known predictors of surgical outcomes (Table 7.1). Prior to data collection, both institutions (UCSF and MNI) agreed on the definitions of all variables, so as to optimize inter-rater reliability. As specialized imaging modalities such as magnetoencephalography (MEG) and single-photon emission computed tomography (SPECT) were only available for a small subset of patients, and are also not routinely performed as part of pre-surgical evaluation in TLE, these variables were not included. Post-surgery Engel [Engel Jr, 1993] outcomes at year one, two, and five (when available) were analyzed as primary endpoints.

Table 7.1: Patient demographics, and their clinical, neurophysiological, and imaging variables collected from UCSF and MNI.

Patient demographics and feature variables	**UCSF (M=167)**	**MNI (M=64)**
Male (%)	84 (50.3)	22 (34.4)
Race (%):		
White	135 (80.8)	33 (51.6)
Black/African American	13 (7.8)	3 (4.7)
Asian	12 (7.2)	3 (4.7)
Others	4 (2.4)	3 (4.7)
Unknown	3 (1.8)	22 (34.4)

continued on next page

continued from previous page

Handedness (%):		
Right	152 (91.0)	51 (79.7)
Left	13 (7.8)	11 (17.2)
Ambidextrous	2 (1.2)	2 (3.1)
History of heavy alcohol/substance use (%)	20 (12.0)	3 (4.7)
Pre-operative migraine (%)	29 (17.4)	5 (7.8)
Pre-operative mood disorder (%)	44 (26.3)	12 (18.8)
Pre-operative psychosis (%)	12 (7.2)	4 (6.3)
Static encephalopathy (%)	22 (13.2)	4 (6.3)
History of co-existing psychogenic non epileptic seizures (%)	4 (2.4)	3 (4.7)
Age at seizure onset, years (mean)	16.0	18.3
Age at surgery, years (mean)	37.1	37.8
Duration of epilepsy till surgery, years (mean)	21.1	19.5
N. of FIAS/month (mean)	6.3	4.7
N. of GTC seizures/year (mean)	4.9	1.9
History of GTC seizures (%)	143 (85.6)	59 (92.2)
At least 6-month seizure-free interval after 1st afebrile seizure:		
Yes (%)	41 (24.6)	27 (42.2)
Mean seizure-free interval (years), SD	7.5 ± 4.5	5.3 ± 5.7
Presence of risk factors present for epilepsy (status epilepticus, perinatal difficulty, head trauma, or encephalitis) (%)	58 (34.7)	24 (37.5)
Family history of seizure disorder (%):		
Yes	22 (13.2)	9 (14.1)
Unknown	1 (0.6)	0
History of febrile seizure (%)	36 (21.6)	21 (32.8)
Type of aura (%):		
Mesial TLE	67 (40.1)	29 (45.3)
Neocortical TLE/extratemporal	16 (9.6)	5 (7.8)
Non-specific/mixed	51 (30.5)	17 (26.6)
None	33 (19.8)	13 (20.3)
Presence of abdominal aura (%)	27 (16.2)	23 (35.9)
Automatisms observed first during seizure (%):		
Yes	80 (47.9)	43 (67.2)
Unknown	6 (3.6)	3 (4.7

continued on next page

continued from previous page

Tonic/clonic/hyperkinetic movements observed first during seizure (%):		
Yes	19 (11.4)	5 (7.8)
Unknown	6 (3.6)	3 (4.7)
N. of disabling seizures captured/duration of vEEG in days (mean)	3.6/4.4 (0.8)	5.8/10.3 (0.6)
Proportion of GTC seizures/N. of disabling seizures captured during vEEG	0.3	0.1
Only nocturnal seizures or seizures arising from sleep recorded (%)	22 (13.2)	10 (15.6)
Total no. of AEDs tried (mean)	5.7	4.8
N. of AEDs at time of surgery (mean)	2.2	2.0
History of documented AED non-compliance (%)	23 (13.8)	6 (9.4)
Laterality of interictal spike (%):		
Ipsilateral	81 (48.5)	38 (59.4)
Bilateral/Generalized/Contralateral	45 (26.9)	21 (32.8)
No spike	14 (8.4)	5 (7.8)
Unknown	27 (16.2)	0
Oligospiker (%):		
Yes	17 (10.2)	15 (23.4)
Unknown	30 (18.0)	0
Typical interictal spike with maximum negativity over anterior temporal region (%):		
Yes	68 (40.7)	44 (68.8)
Unknown	47 (28.1)	6 (9.4)
Ipsilateral temporal slowing seen on EEG (%):		
Yes	31 (18.6)	24 (37.5)
Unknown	56 (33.5)	0
Ictal onset on scalp EEG localized to ipsilateral anterior temporal region (%):		
Yes	92 (55.1)	46 (71.9)
No (e.g., posterior temporal onset, non-lateralized onset, ...)	62 (37.1)	11 (17.2)
Independent bilateral onset	2 (1.2)	6 (9.4)
No seizure recorded	1 (0.6)	1 (1.6)
Unknown	10 (6.0)	0

continued on next page

continued from previous page

Ictal EEG onset (Ebersole [Ebersole and Pacia, 1996]) (%):		
Type 1	53 (31.7)	24 (37.5)
Type 2	46 (27.5)	19 (29.7)
Type 3	21 (12.6)	7 (10.9)
Unknown	47 (28.1)	14 (21.9)
Ictal activity remains localized to ipsilateral temporal region (%):		
Yes	30 (18.0)	17 (26.6)
Unknown	60 (35.9)	9 (14.1)
Intracranial EEG performed (%)	48 (28.7)	8 (12.5)
Intracranial EEG seizure onset (%):		
Mesial	18 (37.5)	7 (87.5)
Neocortical	1 (2.1)	0
Both mesial and neocortical	17 (35.4)	1 (12.5)
Involves extratemporal region	10 (20.8)	0
No seizure recorded	1 (2.1)	0
Unknown	1 (2.1)	0
MRI findings (%):		
Hippocampal atrophy	100 (59.9)	42 (65.6)
Subtle hippocampus changes	20 (12.0)	10 (15.6)
Normal hippocampus	47 (28.1)	12 (18.8)
Bilateral HS present (%)	7 (4.2)	1 (1.6)
Extrahippocampal lesion present (e.g., encephalomalacia, neurocysticercosis) (%)	19 (11.4)	5 (7.8)
Temporal meningoencephalocele present (%)	5 (3.0)	2 (3.1)
PET hypometabolism over ipsilateral temporal region (%)	65/76 (85.5)	19/21 (90.5)
Concordant lateralizing memory deficit on neuropsychology (%):		
Yes	40 (24.0)	25 (39.1)
Unknown	81 (48.5)	4 (6.3)
Side of TLE surgery (%):		
Left	87 (52.1)	30 (46.9)
Right	80 (47.9)	34 (53.1)

continued on next page

continued from previous page

Type of surgery (%):		
ATL	146 (87.4)	20 (31.3)
Selective amygdalohippocampectomy	2 (1.2)	42 (65.6)
Minimally invasive (Visualase/Gamma knife)	9 (5.4)	0
Extended ATL/Tailored surgery	10 (6.0)	2 (3.1)
Histology (%):		
MTS	116 (69.5)	27 (42.2)
Gliosis	33 (19.8)	21 (32.8)
FCD, or MTS+FCD	12 (7.2)	6 (9.4)
Other (microinfarcts, lymphocytic infiltration, etc)	6 (3.6)	10 (15.6)
Reoperation case (%)[1]	13 (7.8)	3 (4.7)
Engel I outcomes (%):		
Year 1	95/167 (56.9)	40/64 (62.5)
Year 2	62/119 (52.1)	36/50 (72.0)
Year 5	26/51 (51.0)	9/16 (56.3)

7.1.2 Data analysis

We used an MBC to predict surgical outcomes one (Y1), two (Y2), and five (Y5) years post-operatively. We distinguished between patients with an Engel I score (i.e.,free of disabling seizures) versus patients with an Engel score of II-IV (i.e.,persistence of disabling seizures).

The presence of both categorical and continuous variables in the datasets greatly increases the complexity of the prediction of TLE surgery outcome. Thus, we discretized the continuous variables into categorical variables with a reduced number of intervals. For this, we used the fixed frequency discretization (FFD) [Yang and Webb, 2009]. Given a sufficient interval frequency (M_I), FFD discretizes the ascendingly sorted values into intervals of approximately M_I instances. Note that the interval frequency may not be close to M_I in the presence of many identical values. The main difference between FFD and the well-known equal frequency discretization (EFD) [Catlett, 1991; Dougherty et al., 1995] is that the former adapts the number of intervals to the number of observed values, which can help to control the discretization variance.

To determine the best subset of features we used the next wrapper approach: First, the feature variables were sorted in descending order according to the information gain (IG) [Cover and Thomas, 2012] provided by each feature variable in relation to the TLE surgery outcomes (maximum IG among Y1, Y2, and Y5). Then, the top k variables were used to learn a model (see Section 7.1.3). To set k, we selected the value that gave the model output a larger macro-averaged area under the ROC curve (macro-AUC) using 5-fold cross-validation in the learning dataset. The macro-AUC is the mean among the AUCs in each time scale.

[1] If this was a reoperation, the Engel outcome after 2nd surgery was used.

7.1.3 Model selection

We used MBCs, an extension of BN classifiers [Bielza and Larrañaga, 2014], to build the predictive model. It was evident that outcomes of TLE surgery at different time scales were related. We learned an MBC that considered these relationships, in which the three class variables were TLE surgery outcome at Y1, Y2, and Y5, and the feature variables were the predictors.

To train the model, we used the hill-climbing BN learning method [Heckerman et al., 1995]. To ensure that the output was always an MBC, the arcs from the feature variables to the class variables were included in a blacklist. The scoring function used to evaluate each MBC structure was AIC [Akaike, 1974], which consists of the log-likelihood of each structure candidate penalized by the number of parameters of the models (see Section 2.3). We used available-case-analysis [Pigott, 2001] to deal with missing values in the training dataset.

When the parameters of the MBC were estimated by maximum likelihood, the resulting model predicted probabilities that fluctuated excessively when the value of certain individual variables changed. The main reason for this problem is that the small number of instances in the learning dataset caused some model parameters to be too extreme. Thus, we used a Bayesian estimation of the parameter with uniform Dirichlet priors (which we set to 1) to improve the stability of the predicted probabilities. The overall performance of the classifier was not affected by this change.

7.1.4 Validation

To evaluate the model performance, we computed the area under the ROC curve (AUC), the classification accuracy (acc), the sensitivity or true positive rate (TPR), and the specificity or true negative rate (TNR) of the proposed MBC against the current gold standard (i.e., the nomograms proposed by Jehi et al. [2015]) in the testing dataset. We used a threshold of 0.5 to assign predicted labels to the instances. This means that when the classifier returns a probability of having an Engel score of I greater than or equal to 0.5, the instance is assigned to class E I. Otherwise, the instance is assigned to class E II-IV.

An in-house developed Python 2.7 package was used to learn the MBC. This package can be found at `https://github.com/marcobb8/tr_bn`. A custom script was written to select the subset of features and can be found at the same location. We used scikit-learn, version 0.18.1, to compute all the evaluation metrics.

7.2 Results

The procedure used to learn the MBC was preliminarily tested using 10-fold cross-validation in the UCSF dataset, yielding a macro-AUC of 0.74. We also tested different well-known methods for learning the MBC, but they produced worse results. Specifically, we tried BIC [Schwarz, 1978] instead of AIC as the scoring function (macro-AUC 0.73), the tree-tree MBC [van der Gaag and de Waal, 2006] learning method instead of the greedy search (macro-AUC

Table 7.2: Intervals obtained after applying the EFD algorithm to discretize the continuous variables.

Variable	Intervals
Age at seizure onset, years	$[0, 3], (3, 9], (9, 15.6], (15.6, 27], > 27$
N. of FIAS/month	$[0, 1.2], (1.2, 3], (3, 7], (7, 10], > 10$
N. of GTC seizures/year	$[0, 1], (1, 12], > 12$
N. of disabling seizures captured /duration of vEEG in days	$[0, 0.33], (0.33, 0.6], (0.6, 0.83], (0.83, 1.33], > 1.33$
Proportion of GTC seizures /N. of disabling seizures captured during vEEG	$[0, 0.718], > 0.718$
Total N. of AEDs tried	$[0, 4], 5, 6, 7, \geq 8$
N. of AEDs at time of surgery	$[0, 2], 3, \geq 4$
Age at surgery, years	$[0, 25.2], (25.2, 33], (33, 40], (40, 47], > 47$
Duration of epilepsy till surgery, years	$[0, 9], (9, 14], (14, 23], (23, 31], > 31$

0.64), and the structural expectation-maximization algorithm [Friedman, 1998] (macro-AUC 0.64) as an alternative to available-case-analysis.

We used the data of 167 UCSF patients as the training cohort, and that of 64 MNI patients for the validation cohort. The UCSF dataset was used exclusively for preprocessing and modeling to ensure fairness of the experimental results. The intervals obtained after discretizing the continuous variables are shown in Table 7.2. The subsets of features selected for each time scale are shown in Table 7.3. The maximum IG among all time scales is shown near each feature. The MBC structure induced from the UCSF dataset is shown in Figure 7.1.

We evaluated the predictive performance of the MBC in the MNI dataset, and compared it with the nomograms proposed by Jehi et al. [2015]. Figure 7.2 shows the ROC curves obtained with both approaches. Table 7.4 gives the AUC, acc, TPR, and TNR obtained with each model for the different time scales. Figure 7.3 shows the calibration of the MBC in the MNI dataset.

The MBC trained on UCSF data showed good discriminative power for MNI data at Y1 and Y2, achieving AUCs of 0.784 and 0.760 respectively. It also performed reasonably well at Y5 (AUC 0.683). The difference in AUC of Y5 with respect to the other time scales may be caused by the small number of observations of Y5 in the MNI dataset.

The MBC clearly discriminated better than the nomograms in all comparable time scales. Only the accuracy at Y2 is slightly higher in the nomogram than in the MBC. This happens because 72% of the samples are labelled as Engel I at year 2 in the MNI data, and the nomogram classifies all the samples as Engel I when we use 0.5 as the cutoff (i.e., it does not discriminate), while the MBC has a TPR and TNR of 0.75 and 0.643 respectively. As the nomograms never assign a probability smaller than 0.5 to any instance at Y2 or Y5, they obtain a TPR of 1 and a TNR of 0 in both cases with a cutoff of 0.5.

In summary, the superior performance of the MBC compared to the nomograms highlights the predictive value and discriminative ability of the subset of selected feature variables, and the capability of the MBC to encode the probabilistic relationships between the predictors

Table 7.3: Variables selected for the prediction of TLE surgery at Y1, Y2, and Y5. Variable V11 combines variables "Automatisms observed first during seizure" and "Tonic/clonic/hyperkinetic movements observed first during seizure" from Table 7.1.

Variable	**ID**	**IG**
Race	V1	0.038
Handedness	V2	0.029
History of heavy alcohol/substance use	V3	0.042
Age at seizure onset, years	V4	0.02
N. of FIAS/month	V5	0.021
N. of GTC seizures/year	V6	0.099
History of GTC seizures	V7	0.021
Presence of risk factors present for epilepsy (status epilepticus, perinatal difficulty, head trauma, or encephalitis)	V8	0.014
Family history of seizure disorder	V9	0.016
Type of aura	V10	0.022
First clinical manifestation during a seizure (automatisms, tonic-clonic or hyperkinetic movements)	V11	0.039
N. of disabling seizures captured /duration of vEEG in days	V12	0.049
Proportion of GTC seizures /N. of disabling seizures captured during vEEG	V13	0.053
Only nocturnal seizures or seizures arising from sleep recorded	V14	0.029
Total no. of AEDs tried	V15	0.07
N. of AEDs at time of surgery	V16	0.017
History of documented AED noncompliance	V17	0.018
Laterality of interictal spike	V18	0.066
Oligospiker	V19	0.037
Typical interictal spike with maximum negativity over anterior temporal region	V20	0.043
Ipsilateral temporal slowing seen on EEG	V21	0.054
Ictal onset on scalp EEG localized to ipsilateral anterior temporal region	V22	0.019
Ictal EEG onset (Ebersole)	V23	0.057
Ictal activity remains localized to ipsilateral temporal region	V24	0.013
MRI findings	V25	0.043
Temporal meningoencephalocele present	V26	0.013
PET hypometabolism over ipsilateral temporal region	V27	0.014
Age at surgery, years	V28	0.145
Duration of epilepsy till surgery, years	V29	0.067
Type of surgery	V30	0.014
Reoperation case	V31	0.127

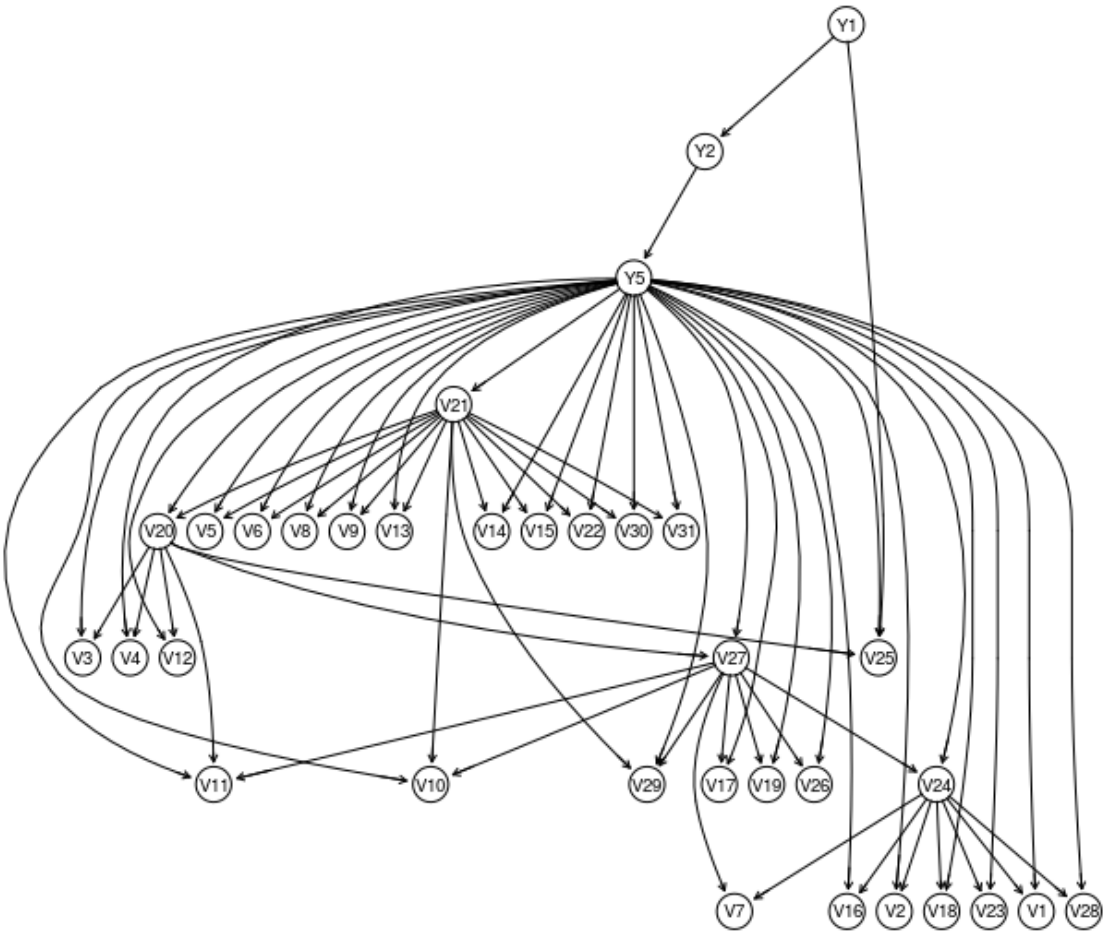


Figure 7.1: Structure of the MBC learned for predicting the outcome of Engel at Y1, Y2, and Y5. Some relevant relationships induced by the structure are analyzed in the discussion. Table 7.3 displays the correspondence between node labels and variable names.

Table 7.4: Comparison of the MBC versus the nomograms for predicting the outcome of TLE surgery in the MNI dataset. Note that Jehi et al. [Jehi et al., 2015] did not provide a nomogram for predicting TLE surgery outcome at Y1. Hence, we only show the results obtained with the MBC at Y2 and Y5.

Time scale	Y1	Y2		Y5	
Method	MBC	MBC	Nomogram	MBC	Nomogram
AUC	0.784	0.760	0.541	0.683	0.611
acc	0.719	0.720	0.729	0.750	0.563
TPR	0.825	0.750	1.000	0.889	1.000
TNR	0.542	0.643	0.000	0.571	0.000

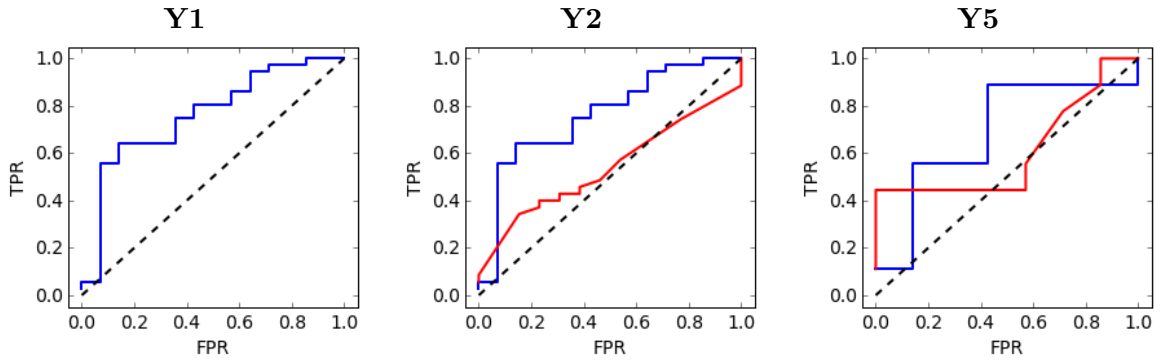


Figure 7.2: ROC curves obtained with the MBC (blue) and the nomograms (red) at Y1 (left), Y2 (center), and Y5 (right). Note that Jehi et al. [Jehi et al., 2015] did not provide a nomogram for predicting TLE surgery outcome at Y1. Hence, we only show the results obtained with the MBC at Y2 and Y5.

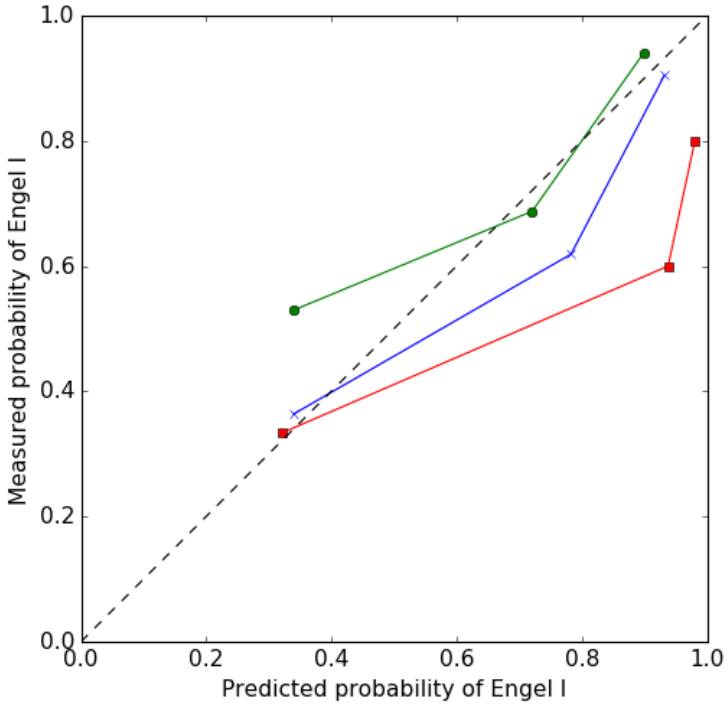


Figure 7.3: Calibration curves obtained with the MBC at Y1 (green), Y2 (blue), and Y5 (red).

and the class variables (i.e., Engel outcomes).

As a deliverable of the current work towards clinical setting, we have developed an online calculator that is freely available on the web (link can be found at http://manikkavacakar.dcmohan.com:3838/tlesop), allowing the reader to enter feature variables for individual TLE patients and to obtain an automated individualized prediction of seizure-free probability at different time scales.

7.3 Discussion

Our MBC generated meaningful results despite minimal supervision and few constraints on the model. This study demonstrates the potential for such a model to give us valuable clinical insights and express complex relationships in patient data beyond what can be offered by expert-designed, highly constrained, hypothesis-driven statistical models. The excellent performance of the MBC on the independent MNI dataset suggests that our machine learning approach can capture general relationships between feature variables and surgical outcomes.

7.3.1 MBC outperformed established nomograms

The established nomogram Jehi et al. [2015] utilized nine feature variables identified to be predictors of seizure freedom after surgery, e.g. sex, age at onset of seizures, age at time of surgery, etc. These variables were combined in a non-linear proportional hazards model that assumes a multiplicative relationship between covariates and prediction. The method considered pairwise interactions between type of surgery and other variables. In contrast, the complex non-linear conditional relationships between the different features and the three outcome probabilities described by the MBC were induced from data. Unlike proportional hazard models, MBCs can robustly handle missing values, which is essential to make the proposed tool useful when the value of some of the predictors is unknown. The outcome probabilities are computed simultaneously and are interdependent. The fact that these results generalize across different datasets suggests that the model that best describes the underlying statistical relationships between clinical predictor variables and surgical outcomes at the three time-scales might be best represented by an MBC-style network.

7.3.2 Clinical interpretation of the MBC

On analysis of the relationships encoded by the structure, the learning algorithm added an arc from Y5 to all the features. From this, we may infer that the relation between the features and Y5 is stronger than the relation between the features and Y1 or Y2. This does not mean that Y1 or Y2 are independent of the features, and more importantly, Y5 is not needed for the prediction of Y1 or Y2. Also, the presence of an arc from Y1 to "MRI findings" (V25) suggests a strong relationship between the presence vs absence of hippocampal atrophy on MRI and seizure freedom outcome at Y1. This is concordant with clinical studies, which

have shown seizure freedom rates a year after TLE surgery to be higher in the presence of a lesion on MRI ("MRI +ve" TLE 75% vs "MRI -ve" TLE 51%) [Téllez-Zenteno et al., 2010].

To further lend transparency to the MBC structure, we also generated MPE values (available online at https://github.com/marcobb8/tr_bn/blob/master/Supplementary%20Material/mpes.pdf) to interrogate the model for clinical robustness. Specifically, we show the most likely value of the features for each possible outcome of the class variables. Reassuringly, most of the MPEs satisfied clinical intuition, for example a combination of features such as a shorter duration of epilepsy at time of surgery, presence of hippocampal atrophy, and type I Ebersole onset portended the best surgical outcome (Engel I). The few instances where features did not discriminate between outcomes or appeared counterintuitive were often a result of skewed frequency distributions and/or influence of missing data at Y5 on the MBC structure.

The interpretation of the predicted probabilities generated by the MBC should be informed by the associated calibration curves (Figure 7.3). Classifier performance was evaluated using a predicted probability of 0.5 as a threshold for predicted seizure freedom, thus probabilities in excess of 0.5 can be expected to favour seizure freedom while predicted probabilities below 0.5 do not. Where the curve is above the identity line the MBC classifier underestimates the probability of seizure freedom (as observed in the test sample), while the probability of seizure freedom is overestimated where the calibration curve falls under the line. We note that the classifier tends towards an overestimation of seizure freedom prediction rates especially for years 2 and 5. For example, an Engel I estimation of 50% at Y2 is a slight overestimation, with the true probability being closer to 45%. An Engel I estimation of 90% at Y5 is a gross overestimation, with the true probability being closer to 55%. At Y5, each point of the calibration curve is obtained from at most six cases, and this may affect the calibration results for this time scale.

The outputs of the classification model should also be interpreted in the proper clinical context when presurgical counseling takes place, even if the algorithm predicts a 40-50% chance of long-term seizure freedom after temporal lobe epilepsy surgery, surgery still confers a far greater chance of seizure freedom compared to additional trials of anti-epileptic medications; fewer than 5% of patients become seizure-free with a third medication regimen after drug-resistance is observed [Kwan and Brodie, 2000].

7.3.3 Limitations

Although the strengths of the study lay in the availability of datasets from two independent tertiary epilepsy centers, the inherent limitations of analyzing retrospectively collected data were missing data values and patients lost to follow-up over time. One can argue that these patients who "disappeared" from our clinics were those who indeed most likely were cured by surgery. MBCs can handle missing values during inference by marginalizing the unobserved variables. If we assume that the mechanism for "drop-out" of the incomplete variables is independent on all the variables in the dataset, then the available-case-analysis method of dealing with missing data should still suffice and provide a reliable output.

7.4 Conclusions

Seizure-freedom outcome prediction for TLE surgery based on MBC modeling outperformed the available nomograms when tested on an independent dataset. The model yielded 78%, 76%, and 68% AUC for outcome predictions at year 1, 2 and 5 respectively, indicating good predictive power. Before this could become a clinical tool in aiding pre-operative counseling, further testing is needed in prospective large cohorts to ensure its reproducibility. Moreover, follow-up information beyond five years would provide a more realistic timeframe for long-term prediction. To further enhance this tool, our long term goal is to allow the MBC classifier to continuously learn from data. Clinicians worldwide could input into the online calculator (i.e., feature variables and Engel outcomes of their own TLE surgical patients), which will optimize its classification accuracy. Future iterations of this model may utilize priors derived from other studies regarding the effects of individual predictors on overall seizure freedom.

Chapter 8

Conclusions and future work

8.1 Summary of contributions

Chapters 3–7 describe the contributions of this thesis.

- Chapter 3 proposes a framework for learning bounded treewidth BNs by efficiently moving in the space of EOs. To this end, we demonstrate that valid ETs avoid the redundancy in the combined space of DAGs and EOs, and provide methods for incrementally compiling valid ETs. The proposed strategies can be included in most score+search BN learning methods. The rest of the chapters make use of these advances to address other highly computationally demanding problems. Experimental results show that our approach successfully bounds the inference complexity of the learned models, and outperforms other state of the art methods in terms of fitting to data.

- Chapter 4 presents a novel approach to efficiently learn BNs from incomplete datasets. The main difference with other methods in the state-of-the-art is that: first, it directly optimizes the log-likelihood of the observed data instead of an expectation, leading to more accurate results, and second, it uses the incremental compilation of ETs to ensure the tractability of the models and the learning procedure. The experimental results support our claims empirically.

- Chapter 5 studies the complexity of computing MPEs with MBCs. We demonstrate that when the features are fully observed, multidimensional classification can be performed efficiently in MBCs with a pruned graph with bounded treewidth, even if the treewidth of the complete graph is not bounded. We propose a method that takes advantage of this property to learn tractable MBCs. The benefits of this strategy are supported by the experiments.

- Chapter 6 provides strategies for the discriminative learning of MBCs. During the structure search, we use penalized CLL to score the models, which can be computed efficiently in MBCs with bounded treewidth. The experiments suggest that this approach leads to more accurate predictions compared to other generative methods.

- Chapter 7 provides a real-world application of MBCs. The data of 167 TLE patients from the UCSF is used to learn an MBC with the goal of predicting seizure freedom at different time scales in patients that have undergone TLE surgery. The explicit representation of the MBC and its generative nature allowed the clinicians to analyze the results in detail. Our proposal outperformed the gold standard in an independent dataset consisting of 64 patients from the MNI.

8.2 List of publications

The research for this thesis has produced the following publications and submissions:

JCR articles

- M. Benjumeda, C. Bielza, and P. Larrañaga. Learning tractable Bayesian networks in the space of elimination orders. *Artificial Intelligence*, 274:66–90, 2019a

- M. Benjumeda, S. Luengo-Sanchez, P. Larrañaga, and C. Bielza. Tractable learning of Bayesian networks from partially observed data. *Pattern Recognition*, 91:190–199, 2019c

- M. Benjumeda, C. Bielza, and P. Larrañaga. Tractability of most probable explanations in multidimensional Bayesian network classifiers. *International Journal of Approximate Reasoning*, 93:74–87, 2018a

Other articles

- M. Benjumeda, P. Larrañaga, and C. Bielza. Learning Bayesian networks with low inference complexity. *Progress in Artificial Intelligence*, 5(1):15–26, 2015a

Proceedings

- M. Benjumeda, C. Bielza, and P. Larrañaga. Learning tractable multidimensional Bayesian network classifiers. In *Proceedings of the 8th International Conference on Probabilistic Graphical Models*, volume 52, pages 25–32. Proceedings of Machine Learning Research, 2016

- M. Benjumeda, P. Larrañaga, and C. Bielza. Learning low inference complexity Bayesian networks. In *Proceedings of the 16th Conference of the Spanish Association for Artificial Intelligence*, pages 1–10. AEPIA, 2015b

Workshops

- M. Benjumeda, S. Luengo-Sanchez, P. Larrañaga, and C. Bielza. Bounding the complexity of structural expectation-maximization. In *Workshop on Tractable Probabilistic Models (ICML)*, 2018b

Submitted

- M. Benjumeda, D. Lowd, P. Larrañaga, and C. Bielza. Discriminative learning of multidimensional Bayesian network classifiers. *Submitted*, 2019b

- M. Benjumeda, Y.-L. Tan, K. A. González-Otárula, D. Chandramohan, E. F. Chang, J. A. Hall, C. Bielza, P. Larrañaga, E. Kobayashi, and R. C. Knowlton. Patient specific prediction of temporal lobe epilepsy surgery. *Submitted*, 2019d

Chapter 3 is derived from Benjumeda et al. [2019a], and extends the work in Benjumeda et al. [2015a,b] to EOs that are not topological. Chapter 4 contains the work of Benjumeda et al. [2019c]. Chapter 5 adapts the work of Benjumeda et al. [2018a], which is an extension of Benjumeda et al. [2016]. Section 5.1 is directly derived from Benjumeda et al. [2018a]. However, the proposed method in Section 5.2 was adapted to use ETs to bound the pruned graph treewidth of the models. Section 5.3 has been extended to test the method in a wider variety of real-world datasets. Chapter 6 contains work in progress, that will be submitted as Benjumeda et al. [2019b]. Finally, Chapter 7 includes the content of Benjumeda et al. [2019d].

8.3 Future work

This section points out future research lines related to the topics covered in this dissertation.

- The use of latent variables in BNs has been extensively discussed [Pearl, 1988; Elidan et al., 2000, 2007]. One of the most appealing properties of latent variables is that a single latent variable can induce dependences among a set of observed variables without the need of having a dense graph connecting them. We think that latent variables are potentially useful to learn more expressive bounded treewidth BNs.

- We intend to study how to learn tractable probabilistic models with large treewidth by taking advange of the local structures or the exchangebility between the variables.

- This thesis focuses mainly on the frequentist approach for learning BNs. Although using Bayesian scoring functions in Chapters 3 and 5 is straightforward, the work in Chapters 4 and 6 assumes the use of a frequentist metric. The Bayesian approach would allow us to incorporate prior knowledge and may improve the generalization properties of the models.

- In Chapter 6 we showed that the CLL can be computed efficiently in MBCs with bounded treewidth. However, the computational cost of computing the CLL of each MBC candidate during the structure search was too high, and we had to rely on a heuristic to reduce the number of times that we computed the CLL. We are interested in extending the work of Carvalho et al. [2011] to approximate the CLL of MBCs.

- In Chapters 5 and 6, we assume that the input datasets are completely observed. We intend to extend this work to problems that involve learning MBCs from incomplete

datasets. For example, in semi-supervised learning the class variables are partially observed. Similar strategies could be applied in multidimensional clustering [Zhang, 2004; Poon et al., 2013; Keivani and Peña, 2016], where all the class variables are not observed and their cardinality is unknown.

- In the last years there is a growing interest in making machine learning algorithms robust under adversarial attacks [Lowd and Meek, 2005; Papernot et al., 2017]. Although these concepts have been used for training the most simple BN classifiers (i.e., naïve Bayes) [Dalvi et al., 2004], it would be interesting to provide algorithms that ensure the robustness of more general BN classifiers and MBCs under adversarial attacks.

Appendix A

Appendix

A.1 Proof of Theorem 1

We use the following lemmas to prove that the compilation and optimization methods proposed in this paper always return valid ETs.

Lemma 3.1. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET that represents $\mathcal{B}$ over $\mathcal{X}$. Given $X_{out}, X_{in} \in \mathcal{X}$, the ET $\mathcal{E}'_{\mathcal{B}'}$ yielded after applying add$(\mathcal{E}_{\mathcal{B}}, X_{out}, X_{in})$ in Algorithm 3.2, is also a valid ET representing $\mathcal{B}'$ over $\mathcal{X}$.*

Proof. By cases:

We prove that, for each possible arc addition scenario, Algorithm 3.2 always outputs valid ETs. We show that, in each case, all the nodes are complete (i.e., for each node its cluster in $\mathcal{E}'_{\mathcal{B}'}$ contains its parents (Definition 3.5)) and sound (i.e., for each node its cluster in $\mathcal{E}'_{\mathcal{B}'}$ is a subset of its predecessors and itself (Definition 3.4)).

- *Case 1*: $X_{\text{out}} \in \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}}$.

  This occurs when neither of the conditions in lines 4 and 6 (Algorithm 3.1) are fulfilled. Algorithm 3.1 does not produce any change in the structure of $\mathcal{E}_{\mathcal{B}}$ (see Figure A.1). Hence, neither the parents nor the predecessors of each node in $\mathcal{E}_{\mathcal{B}}$ change. For each node $X_i \in (\mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}'_{\mathcal{B}'}}_{X_{\text{out}}}) \cup \{\phi_{X_{\text{in}}}\}$, the cluster of X_i now contains X_{out}, but $X_{\text{out}} \in \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_i}$. Hence, each X_i is sound. As $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i) \supseteq \mathbf{Cls}_{\mathcal{E}_{\mathcal{B}}}(X_i)$, each X_i is complete, and therefore valid. There are no changes in the clusters of the other nodes. Hence, they are valid.

- *Case 2*: $X_f = \text{Pa}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \in \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_{\text{out}}}$ (line 4 of Algorithm 3.2).

  Here, Algorithm 3.1 sets $\text{Pa}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}}$ to X_{out} (line 5), and the predecessors and parents of the other nodes are unchanged (see Figure A.2).

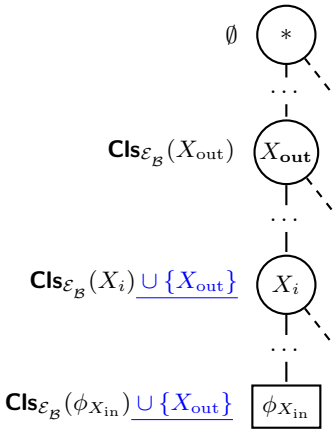


Figure A.1: ET $\mathcal{E}'_{\mathcal{B}'}$ yielded after compiling an arc addition $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{E}_\mathcal{B}$ when $X_{\text{out}} \in \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}}$. The value of the cluster of each node in $\mathcal{E}'_{\mathcal{B}'}$ is shown near to the respective node, and the changes in the clusters with respect to their value in $\mathcal{E}_\mathcal{B}$ are underlined.

- For each node $X_i \in \mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}'_{\mathcal{B}'}}_{X_f}$, we have that $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i) = \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_{\text{in}}})$. First, $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \subseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_i} = \mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{X_i}$. Also, $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_{\text{in}}}) \subseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} = \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_f} \cup \{X_f\} \subseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_i} = \mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{X_i}$. Therefore, $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i) \subseteq \mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{X_i}$, and X_i is sound in $\mathcal{E}'_{\mathcal{B}'}$. As $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(\phi_{X_{\text{in}}}) = \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_{\text{in}}}) \cup \{X_{\text{out}}\}$ and $\mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \supseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \cup \{X_{\text{out}}\}$, $\phi_{X_{\text{in}}}$ is sound. The rest of the nodes are sound given that there are no changes in their clusters.
- Node $\phi_{X_{\text{in}}}$ is complete given that $X_{\text{out}} = \text{Pa}^{\mathcal{E}'_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}}$ and $X_{\text{out}} \in \mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(\phi_{X_{\text{in}}})$. The rest of the nodes are complete given that $\mathbf{Cls}_{\mathcal{E}'_{\mathcal{B}'}}(X_i) \supseteq \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i)$ and $\text{Pa}^{\mathcal{E}'_{\mathcal{B}'}}_{X_i} = \text{Pa}^{\mathcal{E}_\mathcal{B}}_{X_i}$ for each $X_i \in (\mathcal{X} \cup \mathbf{Leaves}(\mathcal{E}_\mathcal{B})) \setminus \{\phi_{X_{\text{in}}}\}$.

- *Case 3*: $X_{\text{out}} \notin \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}}$ and $X_f \notin \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}}$ (line 6 of Algorithm 3.2).

  In this case, there are two possible output ETs, $\mathcal{E}^1_{\mathcal{B}'}$ and $\mathcal{E}^2_{\mathcal{B}'}$ (line 8).

  1. In $\mathcal{E}^1_{\mathcal{B}'}$ (see Figure A.3b), $\text{Pa}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_k}$ is set to X_{out} (line 10):
     - Each node X_i that is not in $(\mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_m}) \cup \{\phi_{X_{\text{in}}}\}$ has the same parents and clusters in $\mathcal{E}_\mathcal{B}$ and $\mathcal{E}^1_{\mathcal{B}'}$, and $\mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_i} \supseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_i}$. Hence, it is valid.
     - Each node X_i in $\mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_k} \cap \mathbf{Desc}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_m}$ has the same predecessors and parents in $\mathcal{E}_\mathcal{B}$ and $\mathcal{E}^1_{\mathcal{B}'}$, and $\mathbf{Cls}_{\mathcal{E}^1_{\mathcal{B}'}}(X_i) = \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_k) \setminus \{X_k\}$ (making X_i complete). As $\mathbf{Cls}_{\mathcal{E}^1_{\mathcal{B}'}}(X_k) \setminus \{X_k\} \subseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_k} \subseteq \mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_i}$, each X_i is also sound. Thus, each X_i is valid.

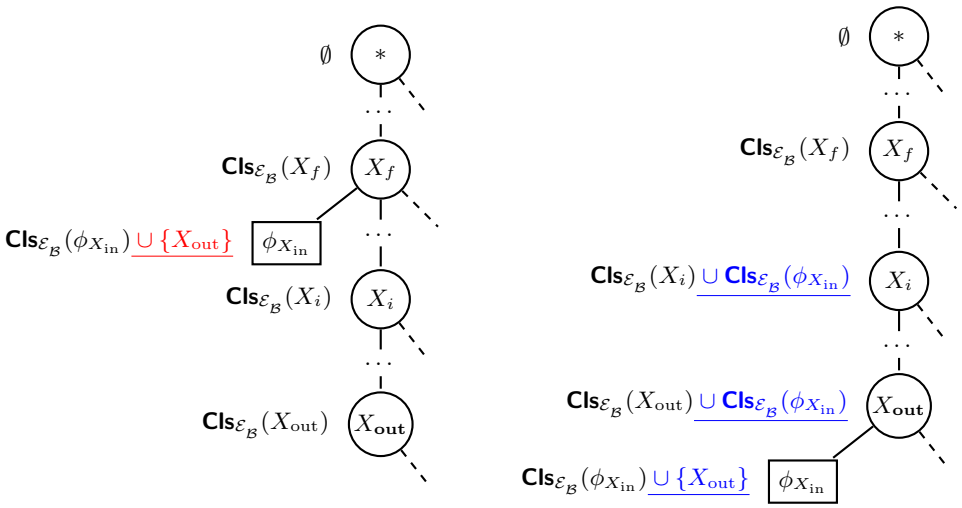


Figure A.2: (a) $\mathcal{E}_{\mathcal{B}'}$ is the ET yielded after adding $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{B}$ when $X_f \in \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}}$ before the compilation of the arc addition; (b) $\mathcal{E}'_{\mathcal{B}'}$ is the result of compiling the arc addition $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{E}_\mathcal{B}$. The value of the cluster of each node in $\mathcal{E}'_{\mathcal{B}'}$ is shown near to the respective node, and the changes in the clusters with respect to their value in $\mathcal{E}_\mathcal{B}$ are underlined. The changes that compromise the validity of the ET are highlighted in red.

- For each $X_i \in (\mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cup \{\phi_{X_{\text{in}}}\}) \cap \mathbf{Desc}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_{\text{out}}}$, $\mathbf{Cls}_{\mathcal{E}^1_{\mathcal{B}'}}(X_i) = \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \{X_{\text{out}}\}$, $\mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_i} \cup \{X_i\} \supseteq \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \{X_{\text{out}}\}$, $\text{Pa}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_i} = \text{Pa}^{\mathcal{E}_\mathcal{B}}_{X_i}$ if $X_i \neq X_k$, and $\text{Pa}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_k} = X_{\text{out}}$. Hence, X_i is valid.

  As each node in $\mathcal{E}^1_{\mathcal{B}'}$ is valid, $\mathcal{E}^1_{\mathcal{B}'}$ is valid.

2. In $\mathcal{E}^2_{\mathcal{B}'}$ (see Figure A.3c), $\text{Pa}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_h}$ is set to X_f (line 12).
   - Each node has the same parent in $\mathcal{E}^2_{\mathcal{B}'}$ and $\mathcal{E}_\mathcal{B}$, with the exception of $\phi_{X_{\text{in}}}$ and X_h, where $\text{Pa}^{\mathcal{E}^2_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} = X_{\text{out}}$ (line 13) and $\text{Pa}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_h} = X_f$. All the nodes are complete, given that $X_{\text{out}} \in \mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}'}}(\phi_{X_{\text{in}}})$, $\mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}'}}(X_h) \supseteq \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_{\text{in}}}) \supseteq \{X_f\}$, and for each other node X_i, $\mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}'}}(X_i) \supseteq \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i)$.
   - For each node X_i not in $(\mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_m}) \cup \{\phi_{X_{\text{in}}}\}$, the clusters of X_i are the same in $\mathcal{E}^2_{\mathcal{B}'}$ and in $\mathcal{E}_\mathcal{B}$ and $\mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_i} \supseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_i}$. Hence each X_i is sound.
   - For each $X_i \in \mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_h} \cap \mathbf{Desc}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_m}$, $\mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}'}}(X_i) = \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_h) \setminus \{X_h\}$ and $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_h) \setminus \{X_h\} \subseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_h} \subseteq \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_i} = \mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_i}$. Thus, each X_i is sound.
   - For each $X_i \in \mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cap \mathbf{Desc}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_f}$, $\mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}'}}(X_i) = \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_{\text{in}}})$ and $\mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{X_i} = \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_i} \cup \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}} \supseteq \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \setminus \{X_i\} \cup \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(\phi_{X_{\text{in}}})$. Hence, X_i is sound.

– Node $\phi_{X_{\text{in}}}$ contains X_{out} in its cluster and predecessors in $\mathcal{E}^2_{\mathcal{B}'}$. Hence, $\mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}'}}(\phi_{X_{\text{in}}}) = \mathbf{Cls}_{\mathcal{E}_{\mathcal{B}}}(\phi_{X_{\text{in}}}) \cup \{X_{\text{out}}\} \subseteq \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{\phi_{X_{\text{in}}}} \cup \{X_{\text{out}}\} \subseteq \mathbf{Pred}^{\mathcal{E}^2_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}}$, making $\phi_{X_{\text{in}}}$ sound.

As every node in $\mathcal{E}^2_{\mathcal{B}'}$ is sound and complete, $\mathcal{E}^2_{\mathcal{B}'}$ is valid.

□

Lemma 3.2. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET that represents $\mathcal{B}$ over $\mathcal{X}$. Given $X_{out}, X_{in} \in \mathcal{X}$, the ET $\mathcal{E}'_{\mathcal{B}'}$ that represents $\mathcal{B}'$ output after applying remove($\mathcal{E}_{\mathcal{B}}, X_{out}, X_{in}$) in Algorithm 3.3, is also valid.*

Proof. By induction. We show that after removing an arc from $\mathcal{E}_{\mathcal{B}}$ only one node X_i may not be complete (base case). In each iteration *iter* of Algorithm 3.3, the completeness of X_i is amended and $\mathcal{E}^{iter}_{\mathcal{B}'}$ is built, and only one other node X'_i, which was a predecessor of X_i in the previous ET, may not be complete after the change. It is evident that eventually node X'_i will be complete (e.g., when the parent of X'_i is the root node $*$).

Base case:

Given a valid ET $\mathcal{E}_{\mathcal{B}}$, removing arc $X_{\text{out}} \to X_{\text{in}}$ from $\mathcal{B}$ will produce an ET $\mathcal{E}_{\mathcal{B}'}$. For each $X_h \in \mathcal{X} \cup \mathbf{Leaves}(\mathcal{E}_{\mathcal{B}'})$, $\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}'}}_{X_h} = \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_h}$, $\text{Pa}^{\mathcal{E}_{\mathcal{B}'}}_{X_h} = \text{Pa}^{\mathcal{E}_{\mathcal{B}}}_{X_h}$, $\mathbf{Cls}_{\mathcal{E}_{\mathcal{B}}}(X_h) \supseteq \mathbf{Cls}_{\mathcal{E}_{\mathcal{B}'}}(X_h) \supseteq \mathbf{Cls}_{\mathcal{E}_{\mathcal{B}}}(X_h) \setminus \{X_{\text{out}}\}$ if $X_h \in \mathbf{Desc}^{\mathcal{E}_{\mathcal{B}'}}_{X_{\text{out}}} \cap \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cup \{\phi_{X_{\text{in}}}\}$, and $\mathbf{Cls}_{\mathcal{E}_{\mathcal{B}'}}(X_h) = \mathbf{Cls}_{\mathcal{E}_{\mathcal{B}}}(X_h)$ otherwise. Therefore, each node in $\mathcal{E}_{\mathcal{B}'}$ is sound, and only one node X_i such that $X_i \in \mathbf{Ch}^{\mathcal{E}_{\mathcal{B}'}}_{X_{\text{out}}} \cap (\mathbf{Pred}^{\mathcal{E}_{\mathcal{B}'}}_{\phi_{X_{\text{in}}}} \cup \{\phi_{X_{\text{in}}}\})$ may not be complete.

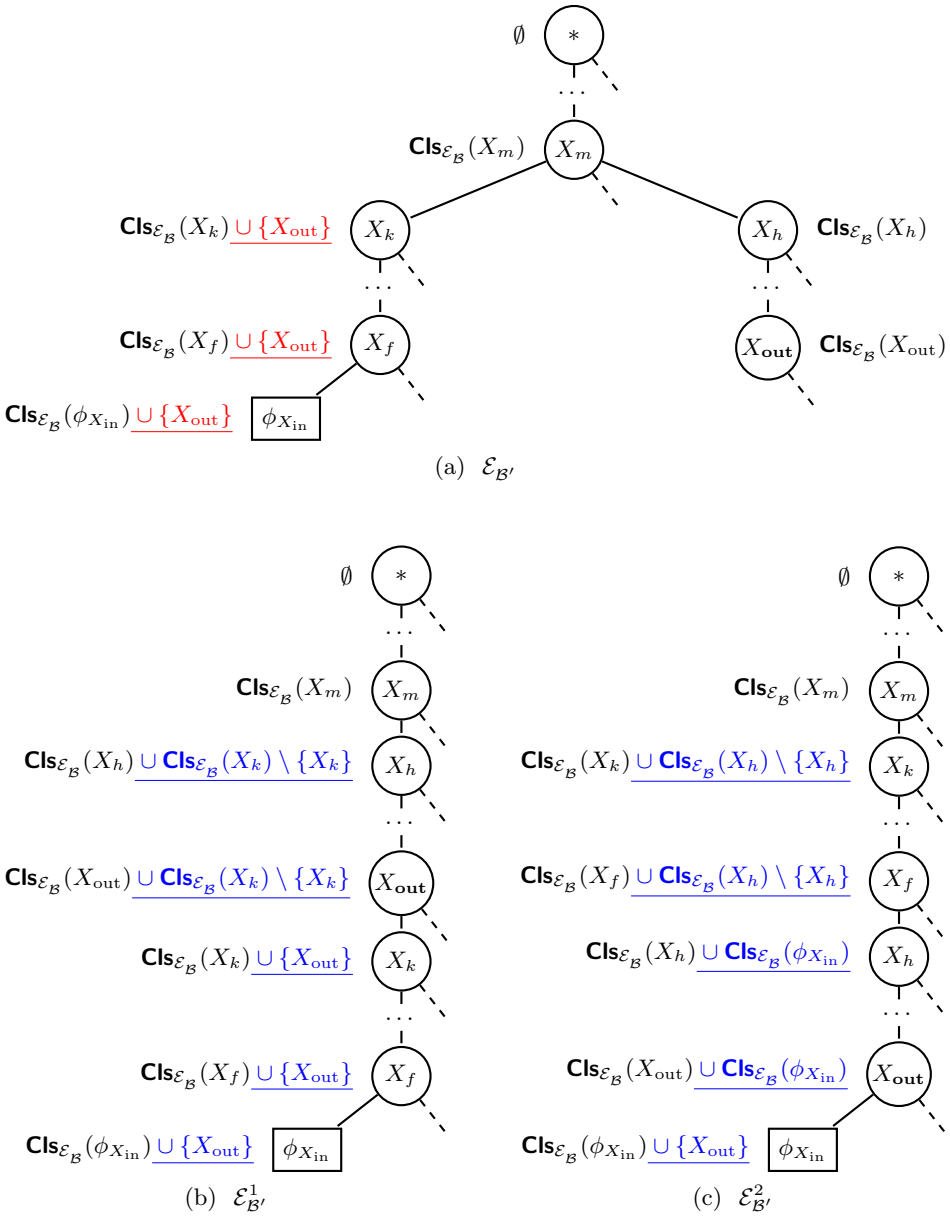
Iterative step:

Assume that $\mathcal{E}^1_{\mathcal{B}'}$ (Figure A.4a) is sound and only node X_i is not complete. Algorithm 3.3 sets $\text{Pa}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_i} = X_j$ (line 7), that is, the deepest node in $\mathcal{E}^1_{\mathcal{B}'}$ belonging to $\mathbf{Cls}_{\mathcal{E}_{\mathcal{B}}}(X_i) \setminus \{X_i\}$ (line 6). Hence, all X_i and their descendants are sound. Thus, node X_i is complete in the resulting ET $\mathcal{E}^2_{\mathcal{B}'}$ (Figure A.4b). Node $X'_i = \mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_i} \cap \mathbf{Ch}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_j}$ may not be complete in $\mathcal{E}^2_{\mathcal{B}'}$. For each node $X_h \in \mathbf{Pred}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_i} \cap \mathbf{Desc}^{\mathcal{E}^1_{\mathcal{B}'}}_{X'_i}$, $\mathbf{Cls}_{\mathcal{E}^2_{\mathcal{B}'}}(X_h) \subseteq \mathbf{Cls}_{\mathcal{E}^1_{\mathcal{B}'}}(X_h) \setminus \mathbf{Cls}_{\mathcal{E}^1_{\mathcal{B}'}}(X_i)$ and $\text{Pa}^{\mathcal{E}^1_{\mathcal{B}'}}_{X_h} \notin \mathbf{Cls}_{\mathcal{E}^1_{\mathcal{B}'}}(X_i)$. Hence, each X_h is complete in $\mathcal{E}^2_{\mathcal{B}'}$. The other nodes have the same clusters in $\mathcal{E}^1_{\mathcal{B}'}$ and in $\mathcal{E}^2_{\mathcal{B}'}$. Hence, they are complete. □

Lemma 3.3. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET that represents $\mathcal{B}$ over $\mathcal{X}$. Given $X_i \in \mathcal{X}$, the ET $\mathcal{E}'_{\mathcal{B}}$ representing $\mathcal{B}$ yielded after applying swap($\mathcal{E}_{\mathcal{B}}, X_i$) in Algorithm 3.4, is also valid.*

Proof. Let X_p be the parent of X_i in $\mathcal{E}_{\mathcal{B}}$ (Figure A.5a). Next, we prove that each node in $\mathcal{E}'_{\mathcal{B}}$ (Figure A.5b) is valid after the swap:

- Each node X_j in $(\mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{X_p} \setminus (\mathbf{Desc}^{\mathcal{E}_{\mathcal{B}}}_{X_i} \cup \{X_i\})) \cup \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_p}$ has the same parent and clusters in $\mathcal{E}'_{\mathcal{B}}$, and $\mathbf{Pred}^{\mathcal{E}'_{\mathcal{B}}}_{X_j} \supset \mathbf{Pred}^{\mathcal{E}_{\mathcal{B}}}_{X_j}$. Hence, each X_j is valid in $\mathcal{E}'_{\mathcal{B}}$.
- Let us divide the descendants of X_i in $\mathcal{E}_{\mathcal{B}}$ into two subsets D_1 and D_2. Let C_1 be the children of X_i in $\mathcal{E}_{\mathcal{B}}$ that do not contain X_p in its cluster. We use D_1 to refer to the



(a) $\mathcal{E}_{\mathcal{B}'}$

(b) $\mathcal{E}^1_{\mathcal{B}'}$

(c) $\mathcal{E}^2_{\mathcal{B}'}$

Figure A.3: $\mathcal{E}_{\mathcal{B}'}$ (Figure A.3a) is the ET yielded after adding $X_{\text{out}} \to X_{\text{in}}$ in $\mathcal{B}$ when $X_{\text{out}} \notin \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{\phi_{X_{\text{in}}}}$ and $X_f \notin \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_{\text{out}}}$ before the compilation of the arc addition. $\mathcal{E}^1_{\mathcal{B}'}$ (Figure A.3b) and $\mathcal{E}^2_{\mathcal{B}'}$ (Figure A.3c) correspond to the two possible outcomes of compiling the arc addition. The value of the cluster of each node in $\mathcal{E}'_{\mathcal{B}'}$ is shown near to the respective node, and the changes in the clusters with respect to their value in $\mathcal{E}_\mathcal{B}$ are underlined. The changes that compromise the validity of the ET are highlighted in red.

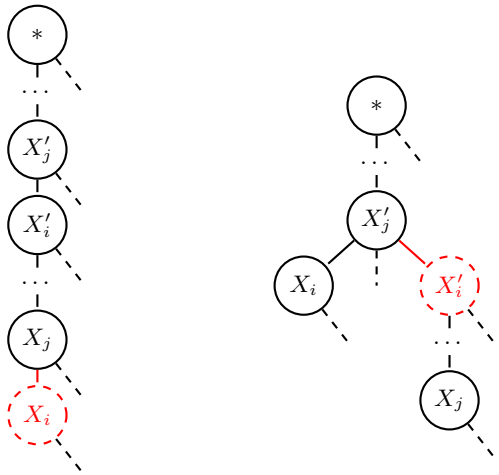


Figure A.4: (a) $\mathcal{E}^1_{\mathcal{B}'}$ is the ET visited at an iterative step of compiling an arc removal; (b) $\mathcal{E}^2_{\mathcal{B}'}$ is the result of performing this iterative step.

nodes in $\mathbf{Desc}^{\mathcal{E}_\mathcal{B}}_{X_i}$ such that for each $X_j \in D_1$, $(\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_j} \cup \{X_j\}) \cap C_1 \neq \emptyset$, and D_2 to refer to the nodes in $\mathbf{Desc}^{\mathcal{E}_\mathcal{B}}_{X_i} \setminus D_1$.

Each node X_j in D_1 has the same parent and cluster in $\mathcal{E}_\mathcal{B}$ and $\mathcal{E}'_\mathcal{B}$, and $\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_j} = \mathbf{Pred}^{\mathcal{E}'_\mathcal{B}}_{X_j} \setminus \{X_p\}$. As $X_p \notin \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_j)$, the respective X_j are valid in $\mathcal{E}'_\mathcal{B}$.

Each X_j in D_2 has the same predecessors and clusters in $\mathcal{E}_\mathcal{B}$ and $\mathcal{E}'_\mathcal{B}$. If $X_j \notin \mathbf{Ch}^{\mathcal{E}_\mathcal{B}}_{X_i}$, X_j has the same parent in $\mathcal{E}_\mathcal{B}$ and $\mathcal{E}'_\mathcal{B}$. Otherwise, $\mathrm{Pa}^{\mathcal{E}'_\mathcal{B}}_{X_j} = X_p$ and $X_p \in \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_j)$. Thus, each X_j is valid in $\mathcal{E}'_\mathcal{B}$.

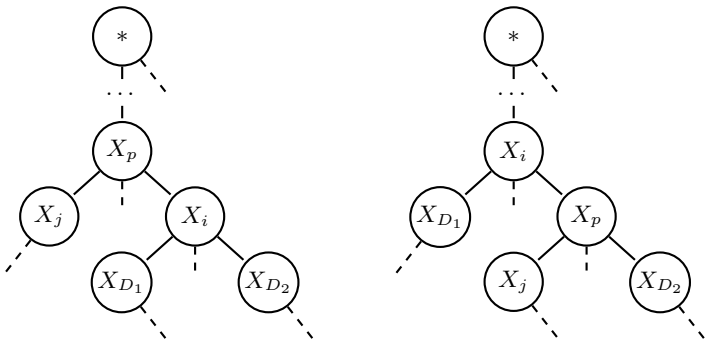
- $X_i \in \mathbf{Cls}_{\mathcal{E}'_\mathcal{B}}(X_p)$, given that there are nodes in D_2 whose cluster contains X_i (otherwise $\mathcal{E}_\mathcal{B}$ would not be complete). This means that X_p is complete for $\mathcal{E}'_\mathcal{B}$. As $\mathbf{Cls}_{\mathcal{E}'_\mathcal{B}}(X_p) \subseteq \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_p) \cup \{X_i\}$ and $\mathbf{Pred}^{\mathcal{E}'_\mathcal{B}}_{X_p} = \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_p} \cup \{X_i\}$, X_p is sound, and therefore valid for $\mathcal{E}'_\mathcal{B}$.

- As $\mathbf{Cls}_{\mathcal{E}'_\mathcal{B}}(X_i) = (\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_p)) \setminus \{X_p\}$, $\mathbf{Pred}^{\mathcal{E}'_\mathcal{B}}_{X_i} = (\mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_i} \cup \mathbf{Pred}^{\mathcal{E}_\mathcal{B}}_{X_p}) \setminus \{X_p\}$, and the parent of X_i in $\mathcal{E}'_\mathcal{B}$ is the parent of X_p in $\mathcal{E}_\mathcal{B}$, X_i is valid for $\mathcal{E}'_\mathcal{B}$.

We have shown that each node in $\mathcal{X} \cup \mathbf{Leaves}(\mathcal{E}'_\mathcal{B})$ is valid in $\mathcal{E}'_\mathcal{B}$. Thus, $\mathcal{E}'_\mathcal{B}$ is valid. □

Finally, Theorem 3.1 can be proved using the above lemmas.

Theorem 3.1. *Let $\mathcal{E}_\mathcal{B}$ be a valid ET over $\mathcal{X}$, and $\mathcal{E}'_{\mathcal{B}'}$ the result of incrementally compiling on $\mathcal{E}_\mathcal{B}$ any local change in $\mathcal{B}$ using Algorithms 3.2 and 3.3 and optimizing the resulting ET using Algorithm 3.5. Then $\mathcal{E}'_{\mathcal{B}'}$ is a valid ET.*



(a) $\mathcal{E}_{\mathcal{B}}$ (b) $\mathcal{E}'_{\mathcal{B}}$

Figure A.5: Swap of X_i and X_p. $\mathcal{E}_{\mathcal{B}}$ (a) and $\mathcal{E}'_{\mathcal{B}}$ (b) correspond to the ETs before and after swapping X_i and X_p, respectively. Note that if $\mathcal{E}_{\mathcal{B}}$ is valid, $\mathcal{E}'_{\mathcal{B}}$ is also valid.

Proof. By Lemmas 3.1 and 3.2 we know that if $\mathcal{E}_{\mathcal{B}}$ is valid, the tree returned after compiling a local change in $\mathcal{E}_{\mathcal{B}}$ is also valid. Hence, if the input for Algorithm 3.5 is a valid ET, we know, by Lemma 3.3 (the optimization is composed of a sequence of swaps), that it will also return a valid ET. □

A.2 Proof of Theorem 2

The following lemmas are used later to prove Theorem 2. First, we need to know the computational cost of outputting the cluster of a node in an ET.

Lemma A.1. *Let $\mathcal{E}_{\mathcal{B}}$ be an ET over $\mathcal{X} = \{X_1, \ldots, X_n\}$. The cluster of a node $X_i \in \mathcal{X}$ can be computed in time $O(|\boldsymbol{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}| \cdot width(\mathcal{E}_{\mathcal{B}}))$ given the clusters of the nodes in $\boldsymbol{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}$.*

Proof. The cluster of node X_i can be output by computing the union of the clusters of its children in $\mathcal{E}_{\mathcal{B}}$ (Definition 3.3). The union of sets $\mathcal{S}_1, \ldots, \mathcal{S}_m$ can be computed in time $|\mathcal{S}_1| + \cdots + |\mathcal{S}_m|$. As the size of each cluster in $\mathcal{E}_{\mathcal{B}}$ is less than or equal to $\text{width}(\mathcal{E}_{\mathcal{B}}) + 1$, then $\sum_{X_j \in \mathbf{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}} |\mathbf{Cls}_{\mathcal{E}_{\mathcal{B}}}(X_j)| \leq \sum_{X_j \in \mathbf{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}} (\text{width}(\mathcal{E}_{\mathcal{B}}) + 1) = |\mathbf{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}| \cdot (\text{width}(\mathcal{E}_{\mathcal{B}}) + 1)$. Hence, the cluster of X_i can be output in time $O(|\mathbf{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}| \cdot \text{width}(\mathcal{E}_{\mathcal{B}}))$. □

Lemma A.2. *Let $\mathcal{E}_{\mathcal{B}}$ be an ET over $\mathcal{X} = \{X_1, \ldots, X_n\}$. All the clusters in $\mathcal{E}_{\mathcal{B}}$ can be computed in time $O(n \cdot width(\mathcal{E}_{\mathcal{B}}))$.*

Proof. The cluster of a leaf node is its domain. The cluster of the inner nodes can be output bottom-up such that before computing the cluster of node X_i the cluster of each child of X_i in $\mathcal{E}_{\mathcal{B}}$ is known. From Lemma A.1, we know that the cluster of each $X_i \in \mathcal{X}$ can be output in time $O(|\mathbf{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}| \cdot \text{width}(\mathcal{E}_{\mathcal{B}}))$. Hence, the clusters of all the nodes in $\mathcal{X}$ can be computed in time $O(\sum_{X_i \in \mathcal{X}} |\mathbf{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}| \cdot \text{width}(\mathcal{E}_{\mathcal{B}}))$. As $\sum_{X_i \in \mathcal{X}} |\mathbf{Ch}_{X_i}^{\mathcal{E}_{\mathcal{B}}}| \cdot \text{width}(\mathcal{E}_{\mathcal{B}}) < 2n \cdot \text{width}(\mathcal{E}_{\mathcal{B}})$ (there are n inner nodes with only one parent, of which at least one is a child of the root node $*$,

and n edges including inner and leaf nodes), all the clusters of $\mathcal{E}_\mathcal{B}$ can be computed in time $O(n \cdot \text{width}(\mathcal{E}_\mathcal{B}))$. □

A local change in an ET $\mathcal{E}_\mathcal{B}$ produces changes in the clusters of the tree, which have an influence on the computational complexity of Algorithm 3.5.

Lemma A.3. *Let $\mathcal{E}_\mathcal{B}$ be a valid ET, and $\mathcal{E}_\mathcal{B}^{'}$ the result of swapping (Algorithm 3.4) a node X_i and its parent in $\mathcal{E}_\mathcal{B}$. Then*

$$\textit{width}(\mathcal{E}_\mathcal{B}^{'}) \leq 2 \cdot \textit{width}(\mathcal{E}_\mathcal{B}).$$

Proof. After swapping X_i and its parent X_p in $\mathcal{E}_\mathcal{B}$, only the clusters of X_i and X_p may change. On the one hand, $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}^{'}}(X_p) \subseteq \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_p)$. Hence, the width of X_p does not grow. On the other hand, the width of X_i may grow, but $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}^{'}}(X_i) \subseteq (\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i) \cup \mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_p)) \setminus \{X_p\}$. Hence, the width of $\mathbf{Cls}_{\mathcal{E}_\mathcal{B}^{'}}(X_i)$ is less than $|\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_i)| + |\mathbf{Cls}_{\mathcal{E}_\mathcal{B}}(X_p)| - 1 \leq 2(\text{width}(\mathcal{E}_\mathcal{B})) + 1$. This means that $\text{width}(\mathcal{E}_\mathcal{B}^{'}) \leq 2 \cdot \text{width}(\mathcal{E}_\mathcal{B})$. □

Next, we bound the time complexity of the compilation and optimization methods.

Lemma A.4. *Let $\mathcal{E}_\mathcal{B}$ be a valid ET over $\mathcal{X} = \{X_1, \ldots, X_n\}$. The addition of any arc in $\mathcal{B}$ can be compiled in $\mathcal{E}_\mathcal{B}$ in time $O(n^2)$ by Algorithm 3.2.*

Proof. There are no loops in Algorithm 3.2, and the only operations that cannot be completed in time $O(1)$ are:

- The intersection performed to compute X_m, X_k and X_h (lines 7, 9 and 11 of Algorithm 3.2), which can be computed in $O(n)$.
- The widht of $\mathcal{E}^1_{\mathcal{B}'}$ and $\mathcal{E}^2_{\mathcal{B}'}$. We need to output first the clusters of $\mathcal{E}^1_{\mathcal{B}'}$ and $\mathcal{E}^2_{\mathcal{B}'}$, which can be obtained in time $O(n \cdot \text{width}(\mathcal{E}_\mathcal{B}))$. The width of $\mathcal{E}_\mathcal{B}$ is the length of its largest cluster munus one (Definition 3.7), which takes $O(n)$. The complete process takes $O(n \cdot \text{width}(\mathcal{E}_\mathcal{B}) + n) = O(n \cdot \text{width}(\mathcal{E}_\mathcal{B})) \leq O(n^2)$.

Therefore, the addition of an arc can be compiled in time $O(n^2)$. □

Lemma A.5. *Let $\mathcal{E}_\mathcal{B}$ be a valid ET over $\mathcal{X} = \{X_1, \ldots, X_n\}$. The removal of any arc in $\mathcal{B}$ can be compiled in $\mathcal{E}_\mathcal{B}$ in time $O(n^2 \cdot \textit{width}(\mathcal{E}_\mathcal{B}))$ by Algorithm 3.3.*

Proof. In each iteration, Algorithm 3.3 checks if a node X_i contains its current parent in its cluster (line 5); else, the new parent of X_i is set to the deepest node X'_j, which appears in the cluster of X_i (line 6). Then, the child of X'_j, which was previously a predecessor of X_i, is set as the new X_i, and X'_j is set as the new X_j for the next iteration (lines 8 and 9 of Algorithm 3.3). Therefore, node X_i is not visited again in the next iterations. This means that there are fewer than n iterations.

The operations in lines 6–9 of Algorithm 3.3 can be completed in time $O(n)$.

The clusters of several nodes must be output after each iteration. By Lemma A.2, we know that the clusters of all the nodes in $\mathcal{X}$ can be computed in time $O(n \cdot \text{width}(\mathcal{E}_\mathcal{B}))$.

As there are fewer than n iterations, Algorithm 3.3 can be run in time $O(n^2 \cdot \text{width}(\mathcal{E}_\mathcal{B}) + n^2) = O(n^2 \cdot \text{width}(\mathcal{E}_\mathcal{B}))$. □

Lemma A.6. *Let $\mathcal{E}_\mathcal{B}$ be a valid ET over $\mathcal{X} = \{X_1, \ldots, X_n\}$. Swapping node X_i and its parent X_p in $\mathcal{E}_\mathcal{B}$ using Algorithm 3.4 and updating the clusters of $\mathcal{E}_\mathcal{B}$ can be completed in time $O(width(\mathcal{E}_\mathcal{B})(|\boldsymbol{Ch}_{X_i}^{\mathcal{E}_\mathcal{B}}| + |\boldsymbol{Ch}_{X_p}^{\mathcal{E}_\mathcal{B}}|))$.*

Proof. To swap a node $X_i \in \mathcal{X}$ with its parent in $\mathcal{E}_\mathcal{B}$, Algorithm 3.4 assigns a new parent to X_i and to its previous parent X_p. It also assigns X_p as the new parent of any children of X_i whose cluster contains X_p. This can be completed in time $O(|\mathbf{Ch}_{X_i}^{\mathcal{E}_\mathcal{B}}|)$. Note that we can check if X_p belongs to a cluster in time $O(1)$.

After swapping X_i and X_p, only the clusters of X_i and X_p change. By Lemma A.1, we know that this can be computed in $O(|\mathbf{Ch}_{X_i}^{\mathcal{E}_\mathcal{B}}|\text{width}(\mathcal{E}_\mathcal{B}) + |\mathbf{Ch}_{X_p}^{\mathcal{E}_\mathcal{B}}|\text{width}(\mathcal{E}_\mathcal{B})) = O(\text{width}(\mathcal{E}_\mathcal{B})(|\mathbf{Ch}_{X_i}^{\mathcal{E}_\mathcal{B}}| + |\mathbf{Ch}_{X_p}^{\mathcal{E}_\mathcal{B}}|))$. □

Lemma A.7. *Let $\mathcal{E}_\mathcal{B}$ be a valid ET over $\mathcal{X} = \{X_1, \ldots, X_n\}$. Algorithm 3.5 can be computed in time $O(n^2 \cdot width(\mathcal{E}_\mathcal{B}))$.*

Proof. The input of Algorithm 3.5 is a list of nodes $\boldsymbol{X}_{\text{opt}} = (X_{l(1)}, \cdots, X_{l(m)})$ for optimization. Assuming that these nodes are ordered from the shallowest to the deepest (i.e. the depth of $X_{l(i)}$ is greater than or equal to the depth of $X_{l(i+1)}$), Algorithm 3.5 starts swapping $X_{l(1)}$ while the width of the ET does not increase, and then it performs the same process with $X_{l(2)}, \cdots, X_{l(m)}$. Thus, the cost of Algorithm 3.5 is given by the cost of each swap performed during the optimization. According to Lemma A.6, Algorithm 3.5 can be computed in time

$$O(\sum_{i=1}^{m}\sum_{j=1}^{k_i} \text{width}(\mathcal{E}_\mathcal{B}^{i,j-1})(|\mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_\mathcal{B}^{i,j-1}}| + |\mathbf{Ch}_{\text{Pa}_{X_{l(i)}}^{\mathcal{E}_\mathcal{B}^{i,j-1}}}^{\mathcal{E}_\mathcal{B}^{i,j-1}}|)),$$

where:

- $k_i < n$ is the number of times that node $X_{l(i)}$ is swapped.
- $\mathcal{E}_\mathcal{B}^{i,j}$ is the ET obtained after swapping node $X_{l(i)}$ j times after nodes $X_{l(1)}, \ldots, X_{l(i-1)}$ have been optimized.
- $\mathcal{E}_\mathcal{B}^{i,0} = \mathcal{E}_\mathcal{B}^{i-1,k_i}$ if $\text{Pa}_{X_{l(i-1)}}^{\mathcal{E}_\mathcal{B}^{i-1,k_i}} = *$ (i.e., swapping node $X_{l(i-1)}$ always reduces the width of the ET candidates until its parent is the root node) and $\mathcal{E}_\mathcal{B}^{i,0} = \mathcal{E}_\mathcal{B}^{i-1,k_i-1}$ otherwise.
- $\mathcal{E}_\mathcal{B}^{1,0} = \mathcal{E}_\mathcal{B}$.

When the width of a candidate $\mathcal{E}_\mathcal{B}^{i,j}$ is bigger than $\text{width}(\mathcal{E}_\mathcal{B})$, $\mathcal{E}_\mathcal{B}^{i,j}$ is rejected. Thus, by Lemma A.3, $\text{width}(\mathcal{E}_\mathcal{B}^{i,j-1}) \leq 2 \cdot \text{width}(\mathcal{E}_\mathcal{B})$, and

$$\sum_{i=1}^{m}\sum_{j=1}^{k_i}\text{width}(\mathcal{E}_{\mathcal{B}}^{i,j-1})(|\mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}|+|\mathbf{Ch}_{\text{Pa}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}|)\leq$$

$$2\cdot\text{width}(\mathcal{E}_{\mathcal{B}})(\sum_{i=1}^{m}\sum_{j=1}^{k_i}|\mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}|+|\mathbf{Ch}_{\text{Pa}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}|).$$

The complexity of Algorithm 3.5 can be output by counting the number of children of each $X_{l(i)}$ and its parent in each iteration.

In any ET $\mathcal{E}_{\mathcal{B}}^{i,j}$, there are less than $2n$ arcs without counting arcs from the root node. Also, note that after swapping node $X_{l(i)}$ with its parent X_p in an ET $\mathcal{E}_{\mathcal{B}}^{i,j}$, $\mathbf{Ch}_{X_p}^{\mathcal{E}_{\mathcal{B}}^{i,j+1}} \supseteq \mathbf{Ch}_{X_p}^{\mathcal{E}_{\mathcal{B}}^{i,j}} \setminus \{X_{l(i)}\}$, $\mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j+1}} \subseteq \mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j}} \setminus \{X_p\}$ and $\mathbf{Ch}_{X_p}^{\mathcal{E}_{\mathcal{B}}^{i,j+1}} \cup \mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j+1}} = \mathbf{Ch}_{X_p}^{\mathcal{E}_{\mathcal{B}}^{i,j}} \cup \mathbf{Ch}_{l(i)}^{\mathcal{E}_{\mathcal{B}}^{i,j}}$. This implies that if a node X_c is the child of a node X_h in any $\mathcal{E}_{\mathcal{B}}^{i,0},\ldots,\mathcal{E}_{\mathcal{B}}^{i,k_i-1}$, it cannot be the child of another node that is not X_h or $X_{l(i)}$ in $\mathcal{E}_{\mathcal{B}}^{i,0},\ldots,\mathcal{E}_{\mathcal{B}}^{i,k_i-1}$. Therefore, it is evident that $\sum_{j=1}^{k_i}|\mathbf{Ch}_{\text{Pa}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}| < 2n$.

To bound $\sum_{i=1}^{m}\sum_{j=1}^{k_i}|\mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}|$, let us focus on the number of children that each node has when it is swapped. As the nodes in $\mathcal{X}_{\text{opt}}$ are visited from the shallowest to the deepest, if a node X_h is a child of node $X_{l(i)}$ when $X_{l(i)}$ is optimized, it cannot be a child of another node $X_{l(j)} \in \mathcal{X}_{\text{opt}}$ when $X_{l(j)}$ is optimized. Thus, each node X_h can be counted less than n times, and given that there are $2n$ inner and leaf nodes in $\mathcal{E}_{\mathcal{B}}$, $\sum_{i=1}^{m}\sum_{j=1}^{k_i}|\mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}| < 2n^2$.

Finally, $2\cdot\text{width}(\mathcal{E}_{\mathcal{B}})\cdot(\sum_{i=1}^{m}\sum_{j=1}^{k_i}|\mathbf{Ch}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}|+\sum_{i=1}^{m}\sum_{j=1}^{k_i}|\mathbf{Ch}_{\text{Pa}_{X_{l(i)}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}}^{\mathcal{E}_{\mathcal{B}}^{i,j-1}}|) < \text{width}(\mathcal{E}_{\mathcal{B}})(4n^2+4n)$. Therefore, Algorithm 3.5 can be computed in time $O(n^2\cdot\text{width}(\mathcal{E}_{\mathcal{B}}))$. □

Theorem 3.2 can be proved using the lemmas shown above.

Theorem 3.2. *Let $\mathcal{E}_{\mathcal{B}}$ be a valid ET over a set of variables $\mathcal{X}=\{X_1,\ldots,X_n\}$. The process described in Theorem 3.1 to output $\mathcal{E}'_{\mathcal{B}'}$ can be performed in time $O(n^2\cdot width(\mathcal{E}_{\mathcal{B}}))$.*

Proof. By Lemmas A.4, A.5 and A.7, we know that both the compilation and optimization process can be performed in time $O(n^2\cdot\text{width}(\mathcal{E}_{\mathcal{B}}))$. □

Bibliography

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

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

J. Arias, J. A. Gámez, T. D. Nielsen, and J. M. Puerta. A scalable pairwise class interaction framework for multidimensional classification. *International Journal of Approximate Reasoning*, 68:194–210, 2016.

S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

F. R. Bach and M. I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, pages 569–576, 2001.

J. Bekker, J. Davis, A. Choi, A. Darwiche, and G. van den Broeck. Tractable learning for complex probability queries. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 2242–2250. MIT Press, Proceedings of Machine Learning Research, 2015.

M. Benjumeda, P. Larrañaga, and C. Bielza. Learning Bayesian networks with low inference complexity. *Progress in Artificial Intelligence*, 5(1):15–26, 2015a.

M. Benjumeda, P. Larrañaga, and C. Bielza. Learning low inference complexity Bayesian networks. In *Proceedings of the 16th Conference of the Spanish Association for Artificial Intelligence*, pages 1–10. AEPIA, 2015b.

M. Benjumeda, C. Bielza, and P. Larrañaga. Learning tractable multidimensional Bayesian network classifiers. In *Proceedings of the 8th International Conference on Probabilistic Graphical Models*, volume 52, pages 25–32. Proceedings of Machine Learning Research, 2016.

M. Benjumeda, C. Bielza, and P. Larrañaga. Tractability of most probable explanations in multidimensional Bayesian network classifiers. *International Journal of Approximate Reasoning*, 93:74–87, 2018a.

M. Benjumeda, S. Luengo-Sanchez, P. Larrañaga, and C. Bielza. Bounding the complexity of structural expectation-maximization. In *Workshop on Tractable Probabilistic Models (ICML)*, 2018b.

M. Benjumeda, C. Bielza, and P. Larrañaga. Learning tractable Bayesian networks in the space of elimination orders. *Artificial Intelligence*, 274:66–90, 2019a.

M. Benjumeda, D. Lowd, P. Larrañaga, and C. Bielza. Discriminative learning of multidimensional Bayesian network classifiers. *Submitted*, 2019b.

M. Benjumeda, S. Luengo-Sanchez, P. Larrañaga, and C. Bielza. Tractable learning of Bayesian networks from partially observed data. *Pattern Recognition*, 91:190–199, 2019c.

M. Benjumeda, Y.-L. Tan, K. A. González-Otárula, D. Chandramohan, E. F. Chang, J. A. Hall, C. Bielza, P. Larrañaga, E. Kobayashi, and R. C. Knowlton. Patient specific prediction of temporal lobe epilepsy surgery. *Submitted*, 2019d.

J. Berg, M. Järvisalo, and B. Malone. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, volume 33, pages 86–95, 2014.

A. Berry, J. R. Blair, P. Heggernes, and B. W. Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39(4):287–298, 2004.

C. Bielza and P. Larrañaga. Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1):5, 2014.

C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705–727, 2011.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 226–234. ACM, 1993.

H. L. Bodlaender and A. M. Koster. Treewidth computations I. Upper bounds. *Information and Computation*, 208(3):259–275, 2010.

H. L. Bodlaender, F. V. Fomin, A. M. Koster, D. Kratsch, and D. M. Thilikos. *On Exact Algorithms for Treewidth*. Springer, 2006.

H. Borchani, C. Bielza, and P. Larrañaga. Learning CB-decomposable multi-dimensional Bayesian network classifiers. pages 25–32. PGM'05, 2010.

R. R. Bouckaert. Probabilistic network construction using the minimum description length principle. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 747, pages 41–48. Springer, 1993.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the 12th International Conference on Uncertainty in Artificial Intelligence*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1):129–156, 1994.

A. M. Carvalho. Scoring functions for learning Bayesian networks. Technical report, University of Lisbon, 2009.

A. M. Carvalho, T. Roos, A. L. Oliveira, and P. Myllymäki. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. *Journal of Machine Learning Research*, 12(7):2181–2210, 2011.

A. M. Carvalho, P. Adao, and P. Mateus. Efficient approximation of the conditional relative entropy with applications to discriminative learning of Bayesian network classifiers. *Entropy*, 15(7):2716–2735, 2013.

J. Catlett. On changing continuous attributes into ordered discrete attributes. In *European Working Session on Learning*, pages 164–178. Springer, 1991.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems*, pages 273–280, 2008.

D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data*, pages 121–130. Springer, 1996.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

F. Clautiaux, A. Moukrim, S. Nègre, and J. Carlier. Heuristic and metaheuristic methods for computing graph treewidth. *RAIRO-Operations Research-Recherche Opérationnelle*, 38 (1):13–26, 2004.

G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2):393–405, 1990.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

G. Corani, A. Antonucci, D. D. Mauá, and S. Gabaglio. Trading off speed and accuracy in multilabel classification. In *Proceedings of the 7th European Workshop on Probabilistic Graphical Models*, pages 145–159. Springer, 2014.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

J. Cussens. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 153–160. AUAI Press, 2011.

P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

N. Dalvi, P. Domingos, S. S. Mausam, and D. Verma. Adversarial classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108. ACM, 2004.

A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1):5–41, 2001.

A. Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

P. R. de Waal and L. C. van der Gaag. Inference and learning in multi-dimensional Bayesian network classifiers. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 501–511. Springer, 2007.

R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85, 1999.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 1:1–38, 1977.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*, pages 194–202. Elsevier, 1995.

J. S. Ebersole and S. V. Pacia. Localization of temporal lobe foci by ictal EEG patterns. *Epilepsia*, 37(4):386–399, 1996.

G. Elidan and S. Gould. Learning bounded treewidth Bayesian networks. In *Advances in Neural Information Processing Systems*, pages 417–424, 2009.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: A structure-based approach. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, pages 458–464. MIT Press, 2000.

G. Elidan, I. Nachman, and N. Friedman. "Ideal parent" structure learning for continuous variable Bayesian networks. *Journal of Machine Learning Research*, 8:1799–1833, 2007.

J. Engel Jr. Outcome with respect to epileptic seizures. *Surgical Treatment of the Epilepsies*, pages 609–621, 1993.

F. V. Fomin and Y. Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012.

F. V. Fomin, D. Kratsch, and I. Todinca. Exact (exponential) algorithms for treewidth and minimum fill-in. In *Automata, Languages and Programming*, volume 3142, pages 568–580. Springer, 2004.

N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the International Conference on Machine Learning*, volume 97, pages 125–133. Morgan Kaufmann Publishers Inc., 1997.

N. Friedman. The Bayesian structural EM algorithm. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 129–138. Morgan Kaufmann Publishers Inc., 1998.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

J. A. Gámez, J. L. Mateo, and J. 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.

S. Garcia and F. Herrera. An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

C. Garcia-Gracia, R. Yardi, M. W. Kattan, D. Nair, A. Gupta, I. Najm, W. Bingaman, J. Gonzalez-Martinez, and L. Jehi. Seizure freedom score: A new simple method to predict success of epilepsy surgery. *Epilepsia*, 56(3):359–365, 2015.

D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20(5):507–534, 1990.

N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 195–200. ACM, 2005.

K. Grant and M. C. Horsch. Methods for constructing balanced elimination trees and other recursive decompositions. *International Journal of Approximate Reasoning*, 50(9):1416–1424, 2009.

R. Greiner, X. Su, B. Shen, and W. Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59(3):297–322, 2005.

D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the 21st International Conference on Machine Learning*, pages 46–53. ACM, 2004.

Y. Guo and S. Gu. Multi-label classification using conditional dependency networks. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1300–1305. AAAI Press, 2011.

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

J. Hernández-González, I. Inza, and J. A. Lozano. Learning Bayesian network classifiers from label proportions. *Pattern Recognition*, 46(12):3425–3440, 2013.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, pages 65–70, 1979.

L. Jehi, R. Yardi, K. Chagin, L. Tassi, G. L. Russo, G. Worrell, W. Hu, F. Cendes, M. Morita, and F. Bartolomei. Development and validation of nomograms to provide individualised predictions of seizure outcomes after epilepsy surgery: A retrospective analysis. *The Lancet Neurology*, 14(3):283–290, 2015.

F. Jensen, S. Lauritzen, and K. Olsen. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

D. Karger and N. Srebro. Learning Markov networks: Maximum bounded tree-width graphs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–401. Society for Industrial and Applied Mathematics, 2001.

O. Keivani and J. M. Peña. Uni-and multi-dimensional clustering via Bayesian networks. In *Unsupervised Learning Algorithms*, pages 163–192. Springer, 2016.

J. H. Kim and J. Pearl. A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, volume 1, pages 190–193. Morgan Kaufmann Publishers Inc., 1983.

U. B. Kjærulff. Triangulation of graphs-algorithms giving small total state space. Technical report, R-90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1990.

U. B. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2(1):7–17, 1992.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

J. Korhonen and P. Parviainen. Exact learning of bounded tree-width Bayesian networks. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, volume 31, pages 370–378. Proceedings of Machine Learning Research, 2013.

A. Koster. *Frequency Assignment: Models and Algorithms.* PhD thesis, Maastricht University, 1999.

P. Kwan and M. Brodie. Early identification of refractory epilepsy. *The New England Journal of Medicine*, 342(5):414–419, 2000.

P. Kwan, A. Arzimanoglou, A. T. Berg, M. J. Brodie, W. Allen Hauser, G. Mathern, S. L. Moshé, E. Perucca, S. Wiebe, and J. French. Definition of drug resistant epilepsy: Consensus proposal by the ad hoc Task Force of the ILAE Commission on Therapeutic Strategies. *Epilepsia*, 51(6):1069–1077, 2010.

J. Kwisthout. Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning*, 52(9):1452–1469, 2011.

J. Kwisthout, H. Bodlaender, and L. van der Gaag. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 237–242. IOS Press, 2010.

W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(3):269–293, 1994.

P. Larrañaga, C. M. Kuijpers, M. Poza, and R. H. Murga. Decomposing Bayesian networks: Triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, 7(1): 19–34, 1997.

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

W. Liao and Q. Ji. Learning Bayesian network parameters under incomplete data with domain knowledge. *Pattern Recognition*, 42(11):3046–3056, 2009.

D. Lowd and J. Davis. Learning Markov network structure with decision trees. In *IEEE International Conference on Data Mining*, pages 334–343. IEEE, 2010.

D. Lowd and P. Domingos. Learning arithmetic circuits. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 383–392. AUAI Press, 2008.

D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 641–647. ACM, 2005.

S. Luengo-Sanchez, C. Bielza, and P. Larrañaga. Hybrid Gaussian and von Mises model-based clustering. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI)*, pages 855–862. IOS Press, 2016.

R. Mahdi and J. Mezey. Sub-local constraint-based learning of Bayesian networks using a joint dependence criterion. *Journal of Machine Learning Research*, 14(1):1563–1603, 2013.

R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, volume 20, pages 1–7. Association for Computational Linguistics, 2002.

O. N. Markand, V. Salanova, E. Whelihan, and C. L. Emsley. Health-related quality of life outcome in medically refractory epilepsy treated with anterior temporal lobectomy. *Epilepsia*, 41(6):749–759, 2000.

H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3(3):255–269, 1957.

A. M. McIntosh, R. M. Kalnins, L. A. Mitchell, G. C. Fabinyi, R. S. Briellmann, and S. F. Berkovic. Temporal lobectomy: Long-term seizure outcome, late recurrence and risks for seizure recurrence. *Brain*, 127(9):2018–2030, 2004.

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, 2008.

A. C. Müller and S. Behnke. pystruct - Learning structured prediction in Python. *Journal of Machine Learning Research*, 15:2055–2060, 2014.

S. Nie, D. D. Mauá, C. P. de Campos, and Q. Ji. Advances in learning Bayesian networks of bounded treewidth. In *Advances in Neural Information Processing Systems*, pages 2285–2293, 2014.

S. Nie, C. P. de Campos, and Q. Ji. Efficient learning of Bayesian networks with bounded tree-width. *International Journal of Approximate Reasoning*, 80:412–427, 2017.

M. Niepert and G. van den Broeck. Tractability through exchangeability: A new perspective on efficient probabilistic inference. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 2467–2475. AAAI Press, 2014.

J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

J. D. Park. MAP complexity results and approximation methods. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 388–396. Morgan Kaufmann Publishers Inc., 2002.

P. Parviainen, H. S. Farahani, and J. Lagergren. Learning bounded tree-width Bayesian networks using integer linear programming. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, volume 33, pages 751–759. Proceedings of Machine Learning Research, 2014.

A. Pastink and L. C. van der Gaag. Multi-classifiers of small treewidth. In *Proceedings of the 13th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 199–209. Springer, 2015.

J. Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the 2nd AAAI Conference on Artificial Intelligence*, pages 133–136. AAAI Press, 1982.

J. Pearl. A constraint propagation approach to probabilistic reasoning. In *Proceedings of the 1st Annual Conference on Uncertainty in Artificial Intelligence*, pages 357–369. Elsevier, 1985.

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

J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

J. Peña, J. Lozano, and P. Larrañaga. An improved Bayesian structural EM algorithm for learning Bayesian networks for clustering. *Pattern Recognition Letters*, 21(8):779–786, 2000.

T. D. Pigott. A review of methods for missing data. *Educational Research and Evaluation*, 7(4):353–383, 2001.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 337–346. AUAI Press, 2011.

L. K. Poon, N. L. Zhang, T. Liu, and A. H. Liu. Model-based clustering of high-dimensional data: Variable selection versus facet determination. *International Journal of Approximate Reasoning*, 54(1):196–215, 2013.

M. Ramoni and P. Sebastiani. Learning Bayesian Networks from incomplete databases. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 401–408. Morgan Kaufmann Publishers Inc., 1997.

P. M. Rancoita, M. Zaffalon, E. Zucca, F. Bertoni, and C. P. de Campos. Bayesian network data imputation with application to survival tree analysis. *Computational Statistics & Data Analysis*, 93:373–387, 2016.

J. Read, P. Reutemann, B. Pfahringer, and G. Holmes. MEKA: A multi-label/multi-target extension to Weka. *Journal of Machine Learning Research*, 17(21):1–5, 2016.

N. Robertson and P. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 59(3):267–296, 2005.

D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.

M. Scanagatta, C. P. d. Campos, G. Corani, and M. Zaffalon. Learning Bayesian networks with thousands of variables. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 1864–1872. MIT Press, 2015.

M. Scanagatta, G. Corani, C. P. de Campos, and M. Zaffalon. Learning treewidth-bounded Bayesian networks with thousands of variables. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1470–1478, 2016.

M. Scanagatta, G. Corani, C. P. de Campos, and M. Zaffalon. Approximate structure learning for large Bayesian networks. *Machine Learning*, 107(8):163–192, 2018a.

M. Scanagatta, G. Corani, M. Zaffalon, J. Yoo, and U. Kang. Efficient learning of bounded-treewidth Bayesian networks from complete and incomplete data sets. *International Journal of Approximate Reasoning*, 95:152–166, 2018b.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

F. Semah, M.-C. Picot, C. Adam, D. Broglin, A. Arzimanoglou, B. Bazin, D. Cavalcanti, and M. Baulac. Is the underlying cause of epilepsy a major prognostic factor for recurrence? *Neurology*, 51(5):1256–1262, 1998.

R. D. Shachter. Evidence absorption and propagation through evidence reversals. In *Proceedings of the 5th Annual Conference on Uncertainty in Artificial Intelligence*, pages 173–190. North-Holland Publishing Co., 1990.

J. P. Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831, 1986.

D. Shahaf and C. Guestrin. Learning thin junction trees via graph cuts. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5, pages 113–120. Proceedings of Machine Learning Research, 2009.

P. P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Proceedings of the 4th Annual Conference on Uncertainty in Artificial Intelligence*, pages 169–198. North-Holland Publishing Co., 1990.

K. Shoikhet and D. Geiger. A practical algorithm for finding optimal triangulations. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Conference on Innovative Applications of Artificial Intelligence*, pages 185–190. AAAI Press, 1997.

S. Spencer and L. Huh. Outcomes of epilepsy surgery in adults and children. *The Lancet Neurology*, 7(6):525–537, 2008.

S. S. Spencer. Long-term outcome after epilepsy surgery. *Epilepsia*, 37(9):807–813, 1996.

P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, 2000.

J. Su, H. Zhang, C. X. Ling, and S. Matwin. Discriminative parameter learning for Bayesian networks. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1016–1023. ACM, 2008.

B. K. Sy. Reasoning MPE to multiply connected belief networks using message passing. In *Proceedings of the 10th National Conference on Artificial intelligence*, pages 570–576. AAAI Press, 1992.

P. Szymański and T. Kajdanowicz. A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints*, 2017.

R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.

J. F. Téllez-Zenteno, L. H. Ronquillo, F. Moien-Afshari, and S. Wiebe. Surgical outcomes in lesional and non-lesional epilepsy: A systematic review and meta-analysis. *Epilepsy Research*, 89(2-3):310–318, 2010.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

S. G. Uijl, F. S. Leijten, J. B. Arends, J. Parra, A. C. Van Huffelen, and K. G. Moons. Prognosis after temporal lobe epilepsy surgery: The value of combining predictors. *Epilepsia*, 49(8):1317–1323, 2008.

L. C. van der Gaag and P. R. de Waal. Multi-dimensional Bayesian network classifiers. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models*, pages 107–114. PGM'06, 2006.

J. Van Haaren and J. Davis. Markov network structure learning: A randomized feature generation approach. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 1148–1154. AAAI Press, 2012.

S. Wang, J. Wang, Z. Wang, and Q. Ji. Enhancing multi-label classification by modeling dependencies among labels. *Pattern Recognition*, 47(10):3405–3413, 2014.

T. Wang, J. W. Touchman, and G. Xue. Applying two-level simulated annealing on Bayesian structure learning to infer genetic networks. In *Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference*, pages 647–648. IEEE Computer Society, 2004.

Y. Yang and G. I. Webb. Discretization for naive-Bayes learning: Managing discretization bias and variance. *Machine Learning*, 74(1):39–74, 2009.

N. A. Zaidi, G. I. Webb, M. J. Carman, F. Petitjean, W. Buntine, M. Hynes, and H. De Sterck. Efficient parameter learning of Bayesian network classifiers. *Machine Learning*, 106(9-10): 1289–1329, 2017.

N. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5(6):697–723, 2004.

N. L. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proceedings of the 10th Canadian Conference on Artificial Intelligence*, pages 171–178. CAIAC, 1994.