THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Pour l'obtention du titre de

Docteur en Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Developpements d'outils d'aide au diagnostic en contexte incertain

Présenté par : Ahmed MABROUK

devant le jury composé de :

M. Philippe LERAY *Professeur des universités, Université de Nantes*	Rapporteur
M. Simon DE GIVRY *Chercheur à l'INRA, MIAT de Toulouse*	Rapporteur
M. Pierre MARQUIS *Professeur des universités, Université d'Artois*	Examinateur
M. Patrice PERNY *Professeur des universités, Université de Paris VI*	Examinateur
M. Christophe GONZALES *Professeur des universités, Université de Paris VI*	Directeur de Thèse
Mme. Karine CHEVALIER-JABET *Ingénieur de recherche, IRSN, Cadarache*	Directrice de Thèse

Remerciements

En premier lieu, je tiens à exprimer toute ma gratitude à mes directeurs de thèse, Christophe Gonzales et Karine Chevalier-Jabet de la confiance qu'ils m'ont accordée en acceptant ma candidature, de leurs conseils précieux et de l'encadrement qui était à la fois rigoureux et enthousiaste tout au long de ce travail doctoral. J'aimerais également leur exprimer à quel point j'ai apprécié leur disponibilité, leurs critiques pertinentes ainsi que leurs qualités humaines exceptionnelles, lesquelles m'ont été d'une très grande utilité durant ces trois ans.

Je remercie sincèrement mon tuteur Eric Chojnacki d'avoir accepté d'assurer le suivi de mes travaux de recherches. Sa large expérience dans le domaine de la statistique ainsi que ses encouragements incessants m'ont été d'une aide précieuse.

Je désire encore exprimer ma très vive reconnaissance à notre responsable de laboratoire Joëlle Fleurot, d'avoir suivi et coordonné l'avancement de mes travaux de recherches.

Je tiens à remercier les examinateurs Patrice Perny et Pierre Marquis pour l'honneur qu'ils m'ont fait d'être dans mon jury, et pour le temps qu'ils ont consacré pour juger mes travaux de thèse.

Je remercie également Philippe Leray et Simon De Givry pour avoir accepté de participer à mon jury en qualité de rapporteur de thèse, et pour les remarques et les améliorations judicieuses qu'ils m'ont faites.

Je tiens aussi à remercier mes collègues que j'ai cotoyés tout au long de ces trois années, à Cadarache, au LIP6, ..., qui m'ont témoigné leur bienveillance et leur gentillesse.

Enfin, je remercie ma famille et tous mes proches de leur soutien et sympathie, entre autres mes parents Habib et Naima qui, grâce à leur inlassable soutien et leurs sacrifices, m'ont permis de réussir toutes mes études et de concrétiser la plupart de mes objectifs. Merci infiniment!

Contents

Chapter 1

General introduction

A severe nuclear accident is defined as an unplanned event resulting from operating errors, equipment failures or external hazards and involving an important contamination of the environment, persons and animals with the released radionuclide materials. According to the International Atomic Energy Agency (IAEA) it is also defined as follows: "an event that has led to significant consequences to people, the environment or the facility".

Given the severity of radiological consequences resulting from severe accidents, the protection against nuclear risks and the study of the effectiveness of possible mitigation measures (that minimize risks to the public and to the environment) has been for several decades a common research issue for all nuclear power plant operators. In particular, the study involves identifying and modeling the physical and chemical phenomenology characterizing the set of possible severe nuclear accidents; exploiting such developed models to perform many safety analyses for nuclear reactors; and, finally propose an emergency plan enabling to mitigate (or prevent) the dangerousness of the accident rather than wait until its occurrence. Modeling severe nuclear accident phenomenology in nuclear power plants is complex and for the past decades extensive experimental programs have been conducted to gain knowledge and build computational tools to help the prediction of the accident progression and its consequences. ASTEC (Accident Source Term Evaluation Code) is such a tool. It is developed by the French Institute of Radioprotection and Nuclear Safety (IRSN) and its German counterpart, the Gesellschaft für Anlagen und Reaktorsicherheit (GRS). The ASTEC code allows to simulate the phenomena characterizing the nuclear accident, starting by the initiating events until the eventual release of radioactive materials in the environment (source term). This code has been widely used in support to many nuclear safety applications such as the analysis of nuclear pressurized water reactor (PWR), the evaluation of source term, and the prognosis of severe nuclear accident scenarios.

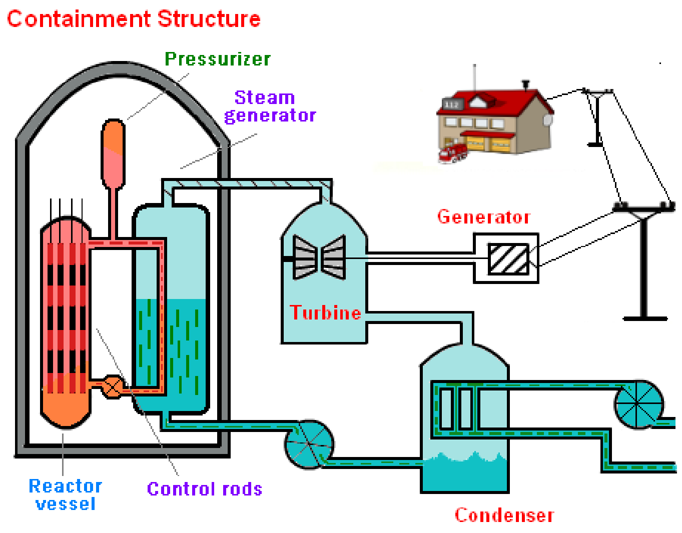


FIGURE 1.1: General schema of the pressurized water reactor.

Before discussing in more details the severe accident management measures and the main issues related to this field, we start in the following by explaining the functioning principle of a nuclear reactor and the main phenomenology occurring during a severe accident. Note that, in our context, we shall focus our discussion on the PWR because all French nuclear reactors (58 reactors distributed over 19 sites) are of this type.

As shown in Figure 1.1[1], in the PWR, the cooling water circulating in the primary circuit (colored in orange/red) is pumped under high pressure (ensured by the pressurizer) into the reactor vessel where it is heated by the energy released by the fission of uranium and/or plutonium atoms, enclosed in a thousand of protective sheaths (which form the fuel assemblies of the core). The overheated water (between 290 and 325 °C) then flows in the steam generator where it mixes the cold water circulating in the secondary circuit (colored in blue/green). This contact converts the thermal energy produced in the reactor core into steam. Through a set of pipes, the steam (at a pressure of 72 bars) flows to the turbines which are then set in turn, transforming therefore the produced steam's energy into a mechanical one. The turbine's mechanical energy spins an electric generator, which generates electrical energy. The heated water of the primary circuit passing through tubes of the steam generator then cools and is carried back into the reactor vessel by the coolant pumps. Concerning the steam at the outlet of the turbine, it is transformed into liquid water by the condenser and, then, it is driven by condensate extraction pumps to the steam generator (secondary circuit).

Given the functional mechanism of the PWR, the severe nuclear accidents occur following a more or less complete reactor core melting. Such a fusion occurs gradually in the reactor due a to a prolonged lack of core reactor cooling with the primary circuit following a series of hardware failures and/or human errors. A typical sequence characterizing the nuclear

[1]The picture has been taken from: http://chemistry.tutorvista.com/nuclear-chemistry/nuclear-chain-reaction.html

accident progress follows hereafter: initially, one or several events cause the degradation of the reactor cooling system; this then leads to the uncover of the core reactor or even the total dewatering. During this time, the fuel sheath (inside the reactor core) are heated, and then break, leading consequently to the escape of radioactive products which are contained in the space between the sheath and the pellets. It should be emphasized that above a certain temperature, of the order of 1100 °C, the superheated steam generated in the reactor vessel leads to a severe oxidation of the sheath, resulting therefore in a substantial production of hydrogen (a flammable gas) which, in case of explosion, may lead to the failure of the containment (that is precisely what happened at Fukushima). Once the sheath of the fuel is totally degraded, the more volatile and semi-volatile fission products in the fuel pellets are released at first into the primary circuit and, then, into the containment via the break. During the core melting phase, the temperature of the fuel increases more and more; the latter is liquefied and then mixed with the molten material structures, forming therefore a magma characterized by a high temperature which is called "corium". At this time, the bottom of the vessel is gradually eroded and deformed by the thermal effect of the corium. Once the phase of vessel degradation is completed, the corium falls on the concrete slab of the reactor, and interacts with it. This phenomenon is called 'corium-concrete interaction'. During this phase, the corium erodes the concrete slab; at the same time, the pressure in the containment increases more and more under the effect of the important release of gases which therefore leads to the failure of the containment integrity and then to the release of radioactive materials into the environment.

The task of severe nuclear accident management has attracted a lot of attention after the accidents in the Three Mile Island Unit in 1979 and in Chernobyl in 1986. In this context, several approaches (notably those using a probabilistic framework) have been developed. Among them, we can mention the Probabilistic Safety Assessment (PSA) which has been published in 1975 in Rasmussen's report [C+75]. Besides the classical deterministic approach, which is generally used during the reactor design phase, the PSA performs a detailed study of scenarios for hypothetical accidents that might lead to the core damage, and then it estimates the probability (or the frequency) of their occurrences. Moreover, the impact of potentially influential parameters, in such a complex system, has been usually assessed by the traditional Monte Carlo method with a number of independent numerical calculations, using a probabilistic framework. All developed safety approaches are of interest not only in determining the extent of the nuclear risk or the set of pertinent parameters explaining the latter, but also for the information they provide about the various components (with their respective weight) characterizing the severe accident and their ability to identify the appropriate safety rules to be established in the reactors. However, the last severe accident that occurred in the Japanese nuclear power plant of Fukushima at Daiichi in 2011 has showed the limitations of the state-of-the-art methods in all that concerns the understanding of relevant phenomena occurring during the severe accidents. As a matter of fact, the Fukushima accident has revealed a number of unexpected problems and weaknesses related to a very wide spectrum of technical fields and operators responsibilities that were not yet considered in the nuclear safety guidelines. In this context, finding a method that allows the understanding of both relevant phenomenology and the main causes explaining the Fukushima accident is crucial. In other words, beyond the wish to make the integral ASTEC code simulating the whole accident, there is also the will to perform a faster diagnosis and to provide a detailed explanation about what happened exactly and which event configuration is responsible for such and such observed phenomena in the accidental reactor, i.e., unlike the simulation (or the prognosis) task addressed by the ASTEC code, the diagnosis analysis exploits the opposite implication of the accident progression. At present, none of the used methods and tools proves to be satisfactory to address efficiently the diagnosis since, according to the domain experts, this task requires:

i) a complex model exploiting efficiently the different interactions between the physical phenomena encoded in ASTEC code;

ii) the consideration of the domain expert's knowledge and that of the uncertainties that may arise from lack of information or imprecision when performing the analysis;

iii) a high level reasoning engine enabling the inference of the original accident scenario (which is unknown) from any set of available partial and possibly unreliable observations.

This thesis is part of the CESAM project, WORKPACKAGE 3.2, which concerns the development of an extension of the ASTEC code for the diagnosis inside and/or outside the nuclear reactor. More precisely, the contribution of this work consists in developing a surrogate model allowing to exploit the expert knowledge present in the ASTEC tool, to improve the understanding of an observed accidental situation and, finally, to identify the most probable scenarios.

For this purpose, we propose to exploit Bayesian networks (BNs). BNs, also known as belief networks, have emerged as one of the most successful tool for diagnosis tasks and have been applied in many real world applications (cancer diagnosis, robotics, machine diagnosis, etc). Unlike many state-of-the-art predictive models such as neural networks, random forests, support vector machine (SVM), BNs provide a prominent model that enables to represent complex systems using graphical and probabilistic formalisms (easy for human to understand). They allow to make a fast automated reasoning about a very wide range of diagnosis queries (Most probable explanation, Maximum *a posteriori*, *etc.*). Besides, such a model can capture the expert's relevant knowledge and it can exploit them to reproduce the expert's problem and efficiently solve it. Such characteristics are very useful in our case since, as discussed earlier, the study of severe nuclear accidents is a complex task that often requires the expert's judgments to facilitate the understanding of many phenomena occurring during the accident. However, building a BN is usually difficult and cannot be manually specified by the expert because it contains a lot of numeric (and graphical) information that are essential to its proper functioning. To handle this issue, numerous efforts have been devoted in the past twenty years to learn both BN's graphical structures and the parameters of their conditional probability tables from datasets. Moreover, the different contexts on which BNs have applied have led researchers to propose tailored learning algorithms. Unfortunately, despite all efforts that have been made in the BN learning area, there exist important critical applications for which no current BN learning algorithm proves to be satisfactory. Such a situation arises in our context, notably in problems of nuclear accident scenario reconstruction from sensors' partial observations. In this context, the task of BN learning from datasets is complex due to the presence of deterministic dependences between random variables that essentially represent some phenomenology occurring in the damaged reactor's core. These deterministic relations present a real issue from the BN learning viewpoint since they rule out the *faithfulness* property on which rely the majority of learning algorithms. Another problem related to the BN learning task arises from the existence of both continuous and discrete variables in our dataset. In the majority of real-world application domains, when such a situation arises, continuous variables are discretized prior to learning because this greatly simplifies the learning task and, in addition, BNs are defined over discrete variables, not continuous ones. However such an approach is doomed to be ineffective because the conditional dependences/arcs learnt during the learning phase are not exploited by the discretization process, thereby leading to an important information loss that prevents the BN learning process to be very effective.

At this stage of this thesis, we could not yet get accurate datasets providing a full description about the possible nuclear accidents. Therefore, in this PhD thesis, we worked only on the theoretical problems that arise in the severe nuclear accident BN learning task. Our goal is to provide a set of algorithms that can be applied successfully on real nuclear accidents datasets. For this purpose, we proposed a new BN learning approach that can cope efficiently with the deterministic relations necessarily contained in the nuclear accidents datasets. We also designed and implemented two multivariate discretization approaches that are very effective to produce high quality BNs, i.e., BNs whose "faithfulness" to the data does not suffer too much from the loss of information induced by the discretization.

This document is organized as follows. In Chapter 2, we present Bayesian networks, their main properties and we show how they can be exploited for probabilistic reasoning. Notably we recall the main classes of algorithms used for this task. In Chapter 3, we provide a non-exhaustive review of state-of-the-art algorithms dedicated to learn both BN graphical structures and their probabilistic components (CPT parameters). In this context, we focus our discussion on discrete BNs. In Chapter 4, we highlight the most common sources leading to the *unfaithfulness* of the probability distribution and we show how this impacts the BN structure learning task. We also discuss the most common state-of-art approaches dedicated to establish the BN structure learning correctness when the *faithfulness* property is ruled out. In Chapter 5, we propose a new approach (with new rules) for learning the structure of BNs when some variables have deterministic relations with others. We also provide the formal proof of correctness of our proposed rules. An experimental comparison of our algorithm with some classical algorithms using a set of synthetic datasets (generated from random BNs) containing deterministic nodes is carried out in Chapter 6. In Chapter 7, we provide two versions of multivariate discretization algorithms designed for the BN structure learning task, which take into account the set of dependences among variables during data transformation. Experimental comparison results of our approaches w.r.t. existing methods using synthetic and real-world datasets are given in Chapter 8. Finally, we draw some conclusions and directions for future researches in Chapter 9.

Part I

State of the art

Chapter 2

Bayesian networks

How to represent compactly the joint probability distribution P of a complex system has received a lot of attention in probability theory. Actually, the knowledge of the joint distribution allows to reason probabilistically about the value of one or more random variables given observations about other variables. As a consequence, it allows to answer a broad range of useful queries. For example, in medical diagnosis applications, the specification of the joint distribution P of the possible diseases and symptoms that a patient might have allows the computation of the posterior probability of any disease given any set of symptoms, tests, *etc.* However, specifying the joint distribution over a high-dimensional space of variables appears completely intractable since the size of such distribution grows exponentially with respect to the number of variables of interest. This problem thus induces significant issues, both from the modeling and the computational points of view.

To cope with this problem, Graphical Probabilistic Models (PGM) provide a powerful framework: they exploit conditional independence properties of the distribution, which they represent by a graph, and those enable both the compact representation of complex distributions and their fast automated reasoning. Besides the compact representation of probability P, PGMs are also intuitively easier for a human to understand than joint probabilities because they highlight the direct dependences between random variables and their overall semantics is easily captured visually through their graphical part (which is usually a sparse graph). Finally, P is usually described as the combination of local distributions, which can be assessed and interpreted by experts. These advantages explain why PGMs are currently a very popular framework for reasoning under uncertainty and are exploited in many real world applications in risk analysis, medicine, robotics, machine diagnosis, data mining, handwriting recognition, *etc.*

This chapter introduces a kind of graphical model called a Bayesian network (BN), for synthesizing joint probability distributions in a manner which can be both intuitive and computationally effective. Then we provide the formal definitions and the semantics of the graphical properties of BNs. Finally, we discuss how BNs can be used to perform probabilistic reasoning through the answering of many relevant queries.

2.1 Definition of Bayesian networks

Bayesian networks, also referred to as Belief networks, are a kind of graphical probabilistic model. They were initially introduced by Judea Pearl [Pea88]. BNs offer mechanisms to accurately represent the dependences between random variables and to perform automated reasoning under uncertainty. They are supplied with fast inference engines that enable to answer efficiently various types of probabilistic queries (computation of marginal, *a priori*, *a posteriori*, probabilities [Dar01, MJ99], of most probable explanations [Nil98], of maximum *a posteriori* [PD04], *etc.*). In this chapter, we shall focus our discussion on discrete Bayesian networks.

Definition 2.1. *A (discrete) BN is a pair* $(\mathcal{G}, \boldsymbol{\Theta})$ *where* $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ *is a directed acyclic graph (DAG),* $\mathcal{X} = \{X_1, X_2, ..., X_n\}$ *represents a set of random variables*[1]*,* $\mathbf{A}$ *is a set of arcs, and* $\boldsymbol{\Theta} = \{P(X_i|\mathbf{Pa}(X_i))\}_{i=1}^n$ *is the set of the conditional probability distributions (CPT) of the variables* X_i *in* $\mathcal{G}$ *given their parents* $\mathbf{Pa}(X_i)$ *in* $\mathcal{G}$*. The BN encodes the joint probability over all the nodes as the product of conditional probabilities as follows:*

$$P(\mathcal{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}(X_i)). \tag{2.1}$$

Equation (2.1) is also called the chain rule or the general product rule. As a concrete example, Figure 2.1 depicts a simple BN related to the causes of the grass of a garden being wet. This BN contains four random variables that represent some physical phenomena: *cloud (C)*, *rain (R)*, *sprinkler (S)*, *wet grass (W)*. Each node in this example can assume two values: TRUE (t) if the phenomenon is present and FALSE (f) otherwise. The directed edges in this network represent probabilistic dependences (or correlations) between nodes: node C is the parent of S and R, and both of the latter are the parents of W. The absence of arcs between pairs (C,W) and (S,R) implies the absence of direct probabilistic dependences between those nodes. Given these dependences/independences, we can say for instance that the presence of rain and sprinkler dictates the state of the wet grass.

As mentioned above, in a BN, it is compulsory to quantify the relationship between each node and its immediate predecessors (its parents). In particular, these relationships are quantified by conditional probability tables (CPTs) for each node, which contain the set

[1]By abuse of notation, we use interchangeably $X_i \in \mathcal{X}$ to denote a node in the BN and its corresponding random variable.

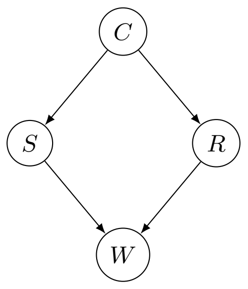


FIGURE 2.1: Example of a classical BN.

of conditional probabilities of each of its values given all the possible combinations of values for its parents, as shown in Figure 2.2. In our example, we thus need to specify for instance the conditional probability $P(W|R, S)$ that grass has a particular value (wet or not), given the presence or absence of rain, and the operational state of the sprinkler. For example, the CPT of node W in Figure 2.2 tells us that $P(W = t|S = t, R = t) = 0.99$, i.e., the grass is almost assuredly wet when it rains and the sprinkler is operational. Concerning variable C in our example, we can notice that this one has no parent (it is a root node), hence we cannot talk about conditional probability. In such a case, a prior (or unconditional) probability for C, i.e., $P(C)$, needs to be specified. As can be seen in Figure 2.2, we assumed that the prior probability for C is represented by a uniform distribution.

$P(C)$	
C	
t	f
0.5	0.5

	$P(R\|C)$	
	R	
C	t	f
t	0.5	0.5
f	0.1	0.9

	$P(S\|C)$	
	S	
C	t	f
t	0.8	0.2
f	0.2	0.8

		$P(W\|S,R)$	
		W	
S	R	t	f
f	f	0	1
f	t	0.9	0.1
t	f	0.9	0.1
t	t	0.99	0.01

FIGURE 2.2: CPTs for nodes C, S, R, W.

The network structure represented by a DAG and the CPTs of the BN together define the joint distribution over all the random variables. In our example, the distribution over the four variables C, R, S, and W can thus be decomposed as follows:

$$P(C, R, S, W) = P(C)P(R|C)P(S|C)P(W|S, R).$$

C	R	S	W	$P(\mathcal{X})$
f	f	f	f	36%
f	f	f	t	0%
f	f	t	f	0.9%
f	f	t	t	8.1%
...	...	...	...	...
t	f	t	t	18%
t	t	f	f	0.5%
t	t	f	t	4.5%
t	t	t	t	19.8%

Table 2.1: The joint probability distribution of the wet grass example.

It should be noted that the joint probability distribution can be visualized through the enumeration of all the possible combinations of assignments of the variables. Table 2.1 lists an excerpt of such assignments with their corresponding probabilities.

As can be seen, the size (i.e., the number of entries) of the joint probability distribution grows exponentially with respect to the number of random variables. In contrast, BNs alleviate this issue by representing the joint distribution with a set of local (conditional) distributions, as shown in Figure 2.2. Note that its CPTs are relatively compact compared to the whole specification of the joint distribution. But they enable the reconstruction of the latter simply by the computation of their product. Through product, division and summation operations, they also enable the determination of any conditional probability of the form $P(\mathbf{W}=\mathbf{w}|\mathbf{V}=\mathbf{v})$, where $\mathbf{W}$ and $\mathbf{V}$ are sets of variables and $\mathbf{w}$ and $\mathbf{v}$ are one of their instantiations. Similarly, any marginal probability of the form $P(\mathbf{W}=\mathbf{w})$ can be computed. For example, the probability $P(R=t)$ in the above BN example can be calculated as follows:

$$\begin{aligned} P(R=t) &= P(R=t|C=t)\times P(C=t)+P(R=t|C=f)\times P(C=f) \\ &= 0.5\times 0.5+0.1\times 0.5 \\ &= 0.3 \end{aligned}$$

Note that, computing conditional probabilities and answering probabilistic queries by means of a BN will be detailed in Section 2.4.

2.2 Graphical properties of Bayesian networks

In the preceding section we only introduced the relationship between a BN and its joint probability distribution. It should be noted that the semantics of the graphical representation of a BN is much more powerful than just representing the presence/absence of direct dependences through the existence/lack of an arc between pairs of nodes. Actually, the BN's DAG can encode more complicated properties about conditional independences that characterize the underlying probability distribution. To facilitate the understanding of the syntax and the semantics of BNs, we start this section by introducing some formal definitions of conditional independences [Daw79] and by recalling some graph theoretic notions.

Definition 2.2. *(Conditional independences) Let* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z} \subset \mathcal{X}$ *be three disjoint sets of variables, and let* P *be a probability distribution defined on* $\mathcal{X}$*. We say that* $\mathbf{X}$ *and* $\mathbf{Y}$ *are conditionally independent given* $\mathbf{Z}$*, which is denoted by* $\mathbf{X} \perp_P \mathbf{Y} | \mathbf{Z}$*, if and only if:*

$$P(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) = P(\mathbf{X}|\mathbf{Z}).$$

The set of all the (conditional) independence assertions present in a given probability distribution will be subsequently denoted by $\mathcal{I}(P)$*.*

When $\mathbf{Z} = \varnothing$*, the conditional independence of* $\mathbf{X}$ *and* $\mathbf{Y}$ *given* $\mathbf{Z}$ *is called an (unconditional) independence. It is denoted by* $\mathbf{X} \perp_P \mathbf{Y}$ *and is equivalent to* $P(\mathbf{X}|\mathbf{Y}) = P(\mathbf{X})$*.*

In other words, a conditional independence between two sets of variables $\mathbf{X}$ and $\mathbf{Y}$ given a set of variables $\mathbf{Z}$ implies that learning some knowledge on $\mathbf{Y}$ does not make any difference on our belief in $\mathbf{X}$ given our knowledge about $\mathbf{Z}$ (and conversely).

As mentioned above, the graphical structure of the BN can be interpreted as a representation of the conditional independence properties of a given probability distribution. Before explaining how conditional independences can be graphically encoded in a BN, let us start this section by recalling some useful graph theoretic notions. In Definition 2.1, we mentioned that a BN is represented by a **DAG** $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ composed by a set of nodes $\mathcal{X}$ representing random variables and a set of oriented edges (or arcs) $\mathbf{A}$ between those nodes. When there exists an arc connecting two nodes X and Y of $\mathcal{X}$, we say that X and Y are **adjacent** in $\mathcal{G}$. The set of all the nodes adjacent to X in $\mathcal{G}$ is denoted by $\mathbf{Adj}(X)_{\mathcal{G}}$ or, simply, $\mathbf{Adj}(X)$ when there is no ambiguity about which graph $\mathcal{G}$ is concerned. Of course, if Y is adjacent to X, X is also adjacent to Y. Given a **DAG** $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ and two nodes X and Y in $\mathcal{X}$, node X is referred to as a **parent** of Y if and only if there exists an arc from X to Y ($X \to Y$) in $\mathcal{G}$. In such a case, Y is referred to as a **child** of X. The set of parents of a node X is denoted by $\mathbf{Pa}(X)_{\mathcal{G}}$ and, when there is no ambiguity about $\mathcal{G}$, by $\mathbf{Pa}(X)$. To generalize these notions, an **ancestor** (resp. a **descendant**) of a given node X is any node Y such that there exists a directed path in $\mathcal{G}$ from Y to X (resp. from X to Y): a **directed path** from Y to X (resp. from X to Y) consists of a sequence of nodes $\{X_{i_1}, X_{i_2}, \ldots, X_{i_k}\}$ such that i) for all $j \in \{1, \ldots, k-1\}$, $\mathcal{G}$ contains the arc $X_{i_j} \to X_{i_{j+1}}$, and ii) $X_{i_1} = Y$ and $X_{i_k} = X$ (resp. $X_{i_1} = X$ and $X_{i_k} = Y$). Figure 2.3 shows an example of a **directed path** from node X to node V. In this example, X, Y, Z, W represent the **ancestors** of V and Y, Z, W, V are the **descendants** of X. Finally, a **trail** between two nodes X and Y is a sequence of nodes

$\{X_{i_1} = X, X_{i_2}, \ldots, X_{i_k} = Y\}$ such that, for all $j \in \{1, \ldots, k-1\}$, $\mathcal{G}$ contains either the arc $X_{i_j} \to X_{i_{j+1}}$ or the arc $X_{i_{j+1}} \to X_{i_j}$. A trail is thus similar to a directed path except that we do not take into account the direction of the arcs.

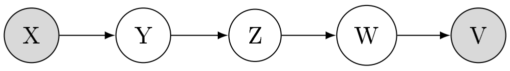


FIGURE 2.3: Example of directed path from node X to node V.

Given the definition of a directed path, the DAG of the BN must guarantee that there is no node that can be at the same time an ancestor and a descendant of itself, i.e., the graphical structure of a BN contains no directed path which is also a loop ($X_{i_k} = X_{i_1}$).

By taking into account all these notions, we can now state the definition of the *local Markov property*:

Definition 2.3. ***(Local Markov property)*** *Let $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$ be a BN, where $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ is a DAG, $\mathcal{X}$ represents a set of random variables and $\mathbf{\Theta}$ is a set of CPTs whose product is equal to a joint distribution P. Let $\mathbf{ND}(X_i)$ denote the set of variables that are non-descendants (X_i and its parents excluded) of X_i in $\mathcal{G}$, then $\mathcal{B}$ encodes the following set of conditional independences:*

$$\forall X_i \in \mathcal{X}, \quad \{X_i\} \perp_P \mathbf{ND}(X_i) | \mathbf{Pa}(X_i). \tag{2.2}$$

For root nodes X_i (the nodes without parents), the *Local Markov Property* states that $\{X_i\} \perp_P \mathbf{ND}(X_i)|\varnothing$ or, in other words, that $\{X_i\} \perp_P \mathbf{ND}(X_i)$. For instance, in Figure 2.4, the *Local Markov Property* states that $\{F\} \perp_P \{D, C, I, A, B\}|\{E\}$ and that $\{E\} \perp_P \{A, B\}$.

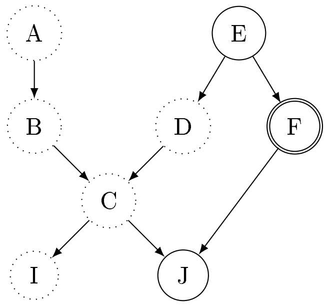


FIGURE 2.4: The *Markov Local property* applied to node F.

Note that the conditional independences entailed by the *Local Markov property* are not the only ones satisfied by distribution P. Additional independences can be inferred from the distribution using a set of properties called the *graphoid axioms* [PP86] which contain: *Symmetry, Decomposition, Weak union, Contraction and Intersection.* Except *Intersection*, all the other axioms are known to hold for any probability distribution.

We start by introducing the formal definitions of these axioms, then we define a graphical criterion called *d-separation* [Pea88] that will be used for retrieving all independences from the graph (those entailed by the *Local Markov property* and the *graphoid axioms*). Note that in the series of DAGs shown in Figures 2.5, 2.6, 2.7 and 2.9 below, which illustrate conditional independences of the type $\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z}$, nodes $\mathbf{X}$ at the left of the $\perp_P$ independence sign are represented by double circles, nodes $\mathbf{Y}$ on the right of the $\perp_P$ sign are represented by dotted circles and nodes $\mathbf{Z}$ at the right of the conditioning bar are represented by shaded circles.

- **Symmetry.** Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z} \subseteq \mathcal{X}$ be disjoint sets of random variables, then:

  $$\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z} \Leftrightarrow \mathbf{Y} \perp_P \mathbf{X}|\mathbf{Z}.$$

  As can be seen in Figure 2.5, if learning information on $\mathbf{X}$ does not influence our belief in $\mathbf{Y}$ given information about $\mathbf{Z}$, then the reverse also holds.

- **Decomposition.** Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{W} \subseteq \mathcal{X}$ be disjoint sets of random variables, then:

  $$\mathbf{X} \perp_P (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z} \Rightarrow (\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z}) \wedge (\mathbf{X} \perp_P \mathbf{W}|\mathbf{Z}).$$

  This property states that if learning new information on both $\mathbf{Y}$ and $\mathbf{W}$ does not influence our belief in $\mathbf{X}$ given information about $\mathbf{Z}$, then those new information are also irrelevant separately. An example of decomposition property is shown in Figure 2.6.

- **Weak union.** Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{W} \subseteq \mathcal{X}$ be disjoint sets of random variables, then:

  $$\mathbf{X} \perp_P (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z} \Rightarrow \mathbf{X} \perp_P \mathbf{Y}|(\mathbf{Z} \cup \mathbf{W}).$$

  This property states that if $\mathbf{W}$ and $\mathbf{Y}$ are irrelevant for $\mathbf{X}$ given $\mathbf{Z}$, then partial information on $\mathbf{Y}$ is also necessarily irrelevant given $\mathbf{Z} \cup \mathbf{W}$. Figure 2.7 illustrates this property.

  In particular, for any $\mathbf{W} \subset \mathbf{ND}(X_i)$, by setting $\mathbf{ND}(X_i) = \mathbf{Y} \cup \mathbf{W}$ and $\mathbf{Pa}(X_i) = \mathbf{Z}$, the application of the weak union axiom on Equation (2.2) of Definition 2.3 implies that:

  $$\forall X_i \in \mathcal{X}, \quad \{X_i\} \perp_P (\mathbf{ND}(X_i) \setminus \mathbf{W})|(\mathbf{Pa}(X_i) \cup \mathbf{W}).$$

- **Contraction.** Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{W} \subseteq \mathcal{X}$ be disjoint sets of random variables, then:

  $$(\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z}) \wedge (\mathbf{X} \perp_P \mathbf{W}|(\mathbf{Z} \cup \mathbf{Y})) \Rightarrow \mathbf{X} \perp_P (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z}.$$

  The contraction axiom is illustrated in Figure 2.8.

- **Intersection.** Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{W} \subseteq \mathcal{X}$ be disjoint sets of random variables, then:

  $$(\mathbf{X} \perp_P \mathbf{Y}|(\mathbf{Z} \cup \mathbf{W})) \wedge (\mathbf{X} \perp_P \mathbf{W}|(\mathbf{Z} \cup \mathbf{Y})) \Rightarrow \mathbf{X} \perp_P (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z}.$$

One example of application of the *Intersection* axiom can be seen in the DAG shown in Figure 2.9: first, the conditional independence $\{X\} \perp_P \{Y\}|\{Z,W\}$ is directly deduced from the combination of the *local Markov property* applied on Y and W (see Definition 2.3) with the *Decomposition* and *Weak union* axioms. $\{X\} \perp_P \{W\}|\{Z,Y\}$ can be inferred in a similar way. Then, by using *Intersection*, we can infer that independence $\{X\} \perp_P \{Y,W\}|\{Z\}$ holds in probability distribution P.

It must be noted that the intersection property is known to hold only for strictly positive probability distributions[2].

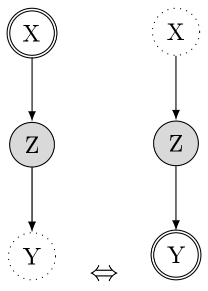


FIGURE 2.5: Symmetry

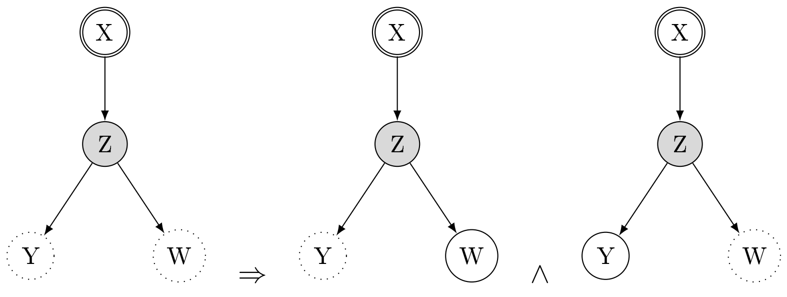


FIGURE 2.6: Decomposition

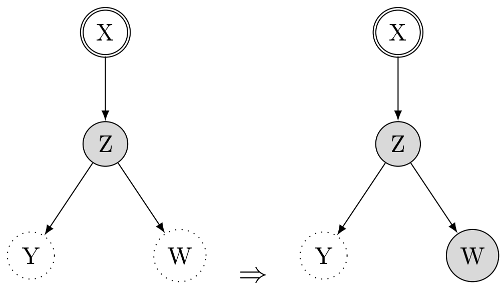


FIGURE 2.7: Weak union

2.2.1 d-separation

We have discussed above how the DAG of a BN can be considered as a map of independence statements. We have seen also how the *local Markov property* can be used to

[2]A probability distribution P is strictly positive if for any assignment x of $\mathcal{X}$, $P(\mathcal{X} = x) > 0$.

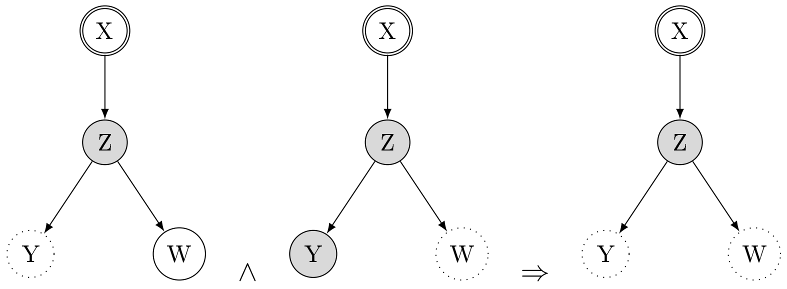


FIGURE 2.8: Contraction

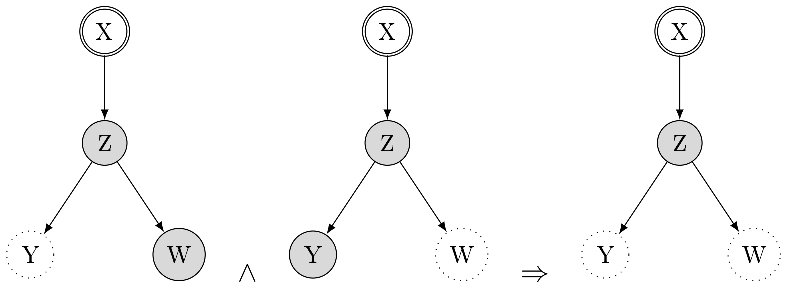


FIGURE 2.9: Intersection

identify a first set of conditional independences between random variables and how the latter can be completed using the set of graphoid axioms. In fact, conditional independence statements derived from these axioms or from the *local Markov property* can also be read from the DAG of the BN using a graphical criterion called *d-separation* [Pea88]. The purpose of the *d-separation* criterion is to provide one simple *graphical* criterion to assert whether some sets of nodes $\mathbf{X}$ are independent of some nodes $\mathbf{Y}$ given some nodes $\mathbf{Z}$. By only relying on the graphical structure of the BN rather than on the probability distribution P, *d-separation* can reveal fewer conditional independences than the whole set related to distribution P. This explains why we will distinguish hereafter the conditional independences detectable by *d-separation*, which we will denote by $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z}$, from those detectable from the probability distribution P, i.e., $\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z}$. The main advantage of using *d-separation* is that it is much easier and faster to handle than using the above axioms. Essentially, *d-separation* consists of computing whether the trails linking nodes of $\mathbf{X}$ and nodes of $\mathbf{Y}$ are "blocked" by $\mathbf{Z}$:

Definition 2.4. *(**Blocked trail**) Let $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ be a directed acyclic graph (DAG) and let X and Y be two nodes of $\mathcal{X}$. Finally, let $\mathbf{Z} \subset \mathcal{X}$ be a set of nodes. Then a trail $\{X_{i_1} = X, \ldots, X_{i_k} = Y\}$ is said to be* **blocked** *by $\mathbf{Z}$ if and only if there exists a node X_{i_j}, $j \in \{2, \ldots, k-1\}$, on the trail such that one of the following conditions holds:*

- *X_{i_j} has converging arcs on the trail, i.e., the trail contains $X_{i_{j-1}} \rightarrow X_{i_j} \leftarrow X_{i_{j+1}}$, and neither X_{i_j} nor any of its descendants are in $\mathbf{Z}$,*

- *X_{i_j} does not have converging arcs on the trail, i.e., the trail contains sequences $X_{i_{j-1}} \rightarrow X_{i_j} \rightarrow X_{i_{j+1}}$ or $X_{i_{j-1}} \leftarrow X_{i_j} \leftarrow X_{i_{j+1}}$ or $X_{i_{j-1}} \leftarrow X_{i_j} \rightarrow X_{i_{j+1}}$, and $X_{i_j} \in \mathbf{Z}$.*

Definition 2.5. *(**d-separation**) Let $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ be a directed acyclic graph (DAG). Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be three disjoint sets of nodes in $\mathcal{G}$. $\mathbf{X}$ and $\mathbf{Y}$ are said to be d-separated by $\mathbf{Z}$, which is denoted by $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z}$, if all the trails between each node in $\mathbf{X}$ and each node in $\mathbf{Y}$ are blocked by $\mathbf{Z}$. When $\mathbf{Z} = \varnothing$, $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z}$ is also denoted as $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}$. The set of d-separation properties $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z}$ of a given graph $\mathcal{G}$ is denoted by $\mathcal{I}(\mathcal{G})$.*

It should be emphasized that if $\mathbf{X}$ and $\mathbf{Y}$ are *d-separated* by $\mathbf{Z}$ in $\mathcal{G}$, then the conditional independence $\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z}$ between $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$ holds in probability distribution P (but the converse may not hold). An example of blocking trails between two single nodes X and Y are shown in Figures 2.10 and 2.11, where conditioning nodes in $\mathbf{Z}$ are shaded:

- In Figure 2.10, trail $\{X, U, Z, V, Y\}$ is blocked by $\mathbf{Z} = \{Z\}$ because the arcs incident to Z are not converging and Z belongs to $\mathbf{Z}$. As this is the only trail between X and Y, this implies that $\{X\} \perp_{\mathcal{G}} \{Y\}|\mathbf{Z}$.

- In Figure 2.11, trail $\{X, U, Z, V, Y\}$ is blocked by $\mathbf{Z} = \varnothing$ because neither Z nor its descendant Q belong to $\mathbf{Z}$ and the arcs incident to Z on the trail are converging (this is also called a *v-structure*). As this is the only trail between X and Y, this implies that $\{X\} \perp_{\mathcal{G}} \{Y\}|\varnothing$ or, equivalently, that $\{X\} \perp_{\mathcal{G}} \{Y\}$.

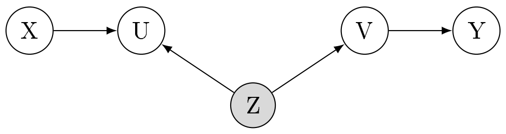


FIGURE 2.10: The trail between X and Y is blocked by $\mathbf{Z} = \{Z\}$.

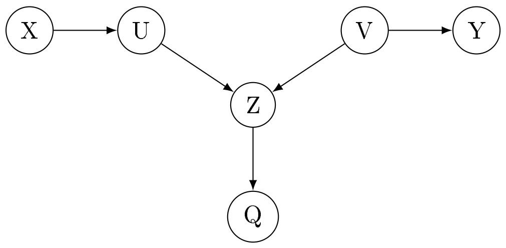


FIGURE 2.11: Blocked trail between X and Y.

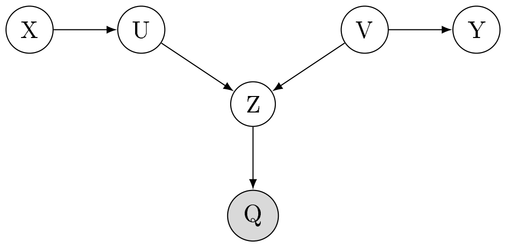


FIGURE 2.12: Unblocked trail between X and Y.

When two variables X and Y are not *d-separated* by $\mathbf{Z} \subset \mathcal{G}$, they are said to be *d-connected*:

Definition 2.6. *(**d-connection**) Let $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ be a directed acyclic graph (DAG). Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be three disjoint sets of variables in $\mathcal{G}$. $\mathbf{X}$ and $\mathbf{Y}$ are said to be d-connected by $\mathbf{Z}$, which is denoted by $\mathbf{X} \not\perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z}$, if there exists a trail between a node of $\mathbf{X}$ and a node of $\mathbf{Y}$ which is not blocked by $\mathbf{Z}$.*

Figure 2.12 provides an example of two variables X and Y d-connected by $\mathbf{Z} = \{Q\}$: actually, trail $\{X, U, Z, V, Y\}$ is not blocked since the nodes without converging arcs (nodes U and V) do not belong to $\mathbf{Z}$ and node Z, with converging arcs, has a descendant in $\mathbf{Z}$.

Due to the notions of blocked/unblocked trails used by the *d-separation* criterion, probabilistic independence relationships defined previously can be represented now by means of graphs. All these definitions will be used later in this chapter to derive more formal definitions about BNs and they will be exploited later to explain in more details our proposed approaches.

2.2.2 I-map, D-map and P-map properties

In this section we will use the *d-separation* criterion to describe notions about the relationships between the conditional independences encoded by a DAG $\mathcal{G}$ and those encoded by a probability distribution P. In particular, we will provide the definition of the following notions: *Independence map* (*I-map*), *Dependence map* (*D-map*) and *Perfect map* (*P-map*).

Definition 2.7. *(**I-map**) We say that a DAG $\mathcal{G}$ is an Independence map of a probability distribution P, which is referred to as an I-map, if every conditional independence captured from $\mathcal{G}$ (e.g., using the d-separation criterion) is also valid in the joint probability distribution P. More formally, given three disjoint sets of variables $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, we have that, if $\mathcal{G}$ is an I-map, then:*

$$\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \perp_{P} \mathbf{Y}|\mathbf{Z}. \tag{2.3}$$

By abuse of notation, the *I-map* property can also be denoted by $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P)$. From the *I-map* definition, we can obviously conclude that distribution P can contain more independences than those entailed by $\mathcal{G}$ but all those that can be read from $\mathcal{G}$ are assured to also belong to P. $\mathcal{G}$ is considered as a minimum *I-map* if and only if deleting any arc of $\mathcal{G}$ makes it no longer an *I-map* of distribution P. Remind that the implication of equation (2.3) does not go in both directions. This means that if two nodes are *d-connected* (i.e. they are not *d-separated*) in $\mathcal{G}$, they are not necessarily dependent in the joint probability distribution P. Therefore, a complete connected DAG (given a topological ordering) is always an *I-map* of any probability distribution P.

Definition 2.8. ***(D-map)*** *We say that a DAG $\mathcal{G}$ is a Dependence map of P, which is also referred to as a D-map, if every conditional independence encoded by distribution P can also be derived from graph $\mathcal{G}$. More formally, given three disjoint sets of variables* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, *we have that, if $\mathcal{G}$ is a D-map, then:*

$$\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z} \Rightarrow \mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z}.$$

The *D-map* property can also be denoted by $\mathcal{I}(P) \subseteq \mathcal{I}(\mathcal{G})$. We say that $\mathcal{G}$ is a minimal *D-map* if and only if adding any new arc to $\mathcal{G}$ makes it no longer a *D-map*. Obviously, a BN with no arc is necessarily a *D-map*. A D-map $\mathcal{G}$ thus encodes more independences than those existing in P.

Definition 2.9. ***(P-map)*** *A DAG $\mathcal{G}$ is a perfect map (P-map) if it captures all the independences/dependences in distribution P. More formally, given three disjoint sets of variables* $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, *we have that, if $\mathcal{G}$ is a P-map, then:*

$$\mathbf{X} \perp_P \mathbf{Y}|\mathbf{Z} \Leftrightarrow \mathbf{Y} \perp_{\mathcal{G}} \mathbf{X}|\mathbf{Z}.$$

In other words, graph $\mathcal{G}$ is considered as a *P-map* if and only if it is in the same time an *I-map* and a *D-map* of P. Hence, this leads to $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{P})$. Therefore, a *P-map* is unique for a given distribution P up to *DAG equivalence* (details about *DAG equivalence* will be illustrated in the next section). Given the definition of a *P-map*, one important question may arise: is there a *P-map* for any given P since our purpose is to precisely represent all independences of P? As it turns out, the answer is generally negative. It can be proved that strictly positive distributions can be represented by perfect maps but, in most practical situations, probability distributions are not strictly positive and, in this case, perfect maps do not often exist. The problems induced by P not being possibly represented by a *P-map* is a serious issue for learning BNs from databases. This will be discussed in the third chapter of this thesis because, in some sense, it is at the core of our research.

2.2.3 Markov blanket

Given a Bayesian network, the posterior probability $P(X|\mathbf{Z} = \mathbf{z})$ of a given node X is usually altered whenever a set of nodes $\mathbf{Z}$ are instantiated ($\mathbf{Z} = \mathbf{z}$). However, due to the notion of conditional independences previously defined, it turns out that there exists a set of nodes $\mathbf{MB}(X)$, called the *Markov blanket* of node X, such that X is guaranteed to be unaffected by any instantiation of the nodes in $\mathbf{Z} \setminus \mathbf{MB}(X)$ given $\mathbf{MB}(X)$. To put it

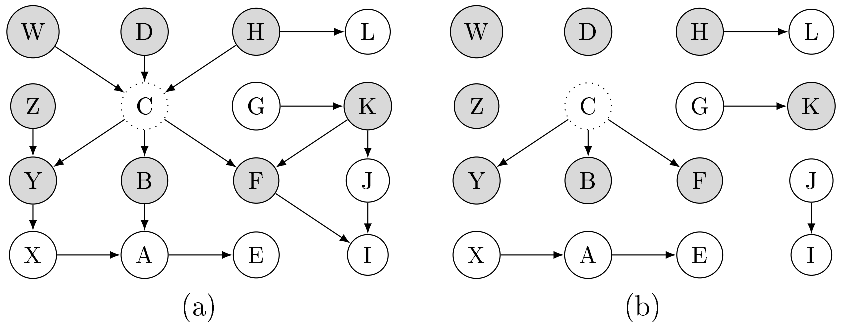


FIGURE 2.13: (a) The Markov blanket of node C. (b) DAG resulting from the deletion of the arcs outgoing from the nodes of $\mathbf{MB}(C)$

differently, the *Markov blanket* of a node X is the smallest set of nodes that completely determines our beliefs on X, whatever any additional knowledge we may have on the other nodes of the BN.

Definition 2.10. ***(Markov blanket)*** *For any distribution P decomposable w.r.t. DAG $\mathcal{G} = (\mathcal{X}, \mathbf{A})$, the* Markov blanket *of a node $X \in \mathcal{X}$, which is denoted by $\mathbf{MB}(X)$, is the smallest set of nodes in $\mathcal{X} \setminus \{X\}$ such that, given $\mathbf{MB}(X)$, X is independent from all the other variables of the network. Formally, $\mathbf{MB}(X)$ is defined as follows:*

$$\forall X \in \mathcal{X}, \mathbf{MB}(X) = \underset{|\mathbf{Y}|}{Argmin}\{\mathbf{Y} : \{X\} \perp_P (\mathcal{X} \setminus (\{X\} \cup \mathbf{Y}))|\mathbf{Y}\}.$$

In a BN, the Markov blanket *of a given node X can be easily identified from $\mathcal{G}$: it consists of the union of the set of its parents ($\mathbf{Pa}(X)$), of the set of its children ($\mathbf{Ch}(X)$) and of its spouses (the parents of the children of X, except X itself).*

Consider the example of BN given in Figure 2.13.a. By Definition 2.10, it is easy to see that the *Markov blanket* of node C is equal to $\{W, D, H, Z, Y, B, F, K\}$, i.e., the set of shaded nodes. As it is shown in Figure 2.13.b, when deleting all the arcs outgoing from the nodes of the Markov blanket, we obtain a node X disconnected from all the nodes in $\mathcal{G}$ except from its children, hence X is disconnected from all the nodes not belonging to its Markov blanket.

2.3 Markov equivalence class

Several DAGs can represent the same set of conditional independences. As an example, a DAG $\mathcal{G}_1$ composed by two nodes X and Y and one arc $X \to Y$ represents a world in which X and Y are dependent. The same world can be represented by DAG $\mathcal{G}_2$ composed by the same nodes X and Y and one arc $Y \to X$. We consider that two graphs belong to the same *Markov equivalence class* if they encode precisely the same set of conditional independence assertions:

Definition 2.11. ***(Markov equivalence class)*** *Let $\mathcal{G}_1 = (\mathcal{X}, \mathbf{A_1})$ and $\mathcal{G}_2 = (\mathcal{X}, \mathbf{A_2})$ be two DAGs containing the same set of random variables $\mathcal{X}$. Graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ belong to the same Markov equivalence class if and only if each* d-separation *statement between any two disjoint set of variables $\mathbf{X}$ and $\mathbf{Y}$ given a set $\mathbf{Z}$ in $\mathcal{G}_1$ implies the same d-separation statement in $\mathcal{G}_2$, and conversely. That is:*

$$\mathbf{X} \perp_{\mathcal{G}_1} \mathbf{Y}|\mathbf{Z} \Leftrightarrow \mathbf{X} \perp_{\mathcal{G}_2} \mathbf{Y}|\mathbf{Z}.$$

By abuse of notation, the Markov equivalence between $\mathcal{G}_1$ and $\mathcal{G}_2$ can be denoted by $\mathcal{I}(\mathcal{G}_1) \equiv \mathcal{I}(\mathcal{G}_2)$.

For more details about the definition of *Markov equivalence*, we consider four examples of BNs represented in Figure 2.14. All of these BNs encode the same set of conditional independence assertions: for instance, $\{Y\} \perp_{\mathcal{G}} \{Z\}|\{W\}$ and $\{A\} \perp_{\mathcal{G}} \{Y, Z\}|\{W\}$. Therefore, any distribution P that can be factorized w.r.t. one of these DAGs can also be factorized w.r.t. the other three.

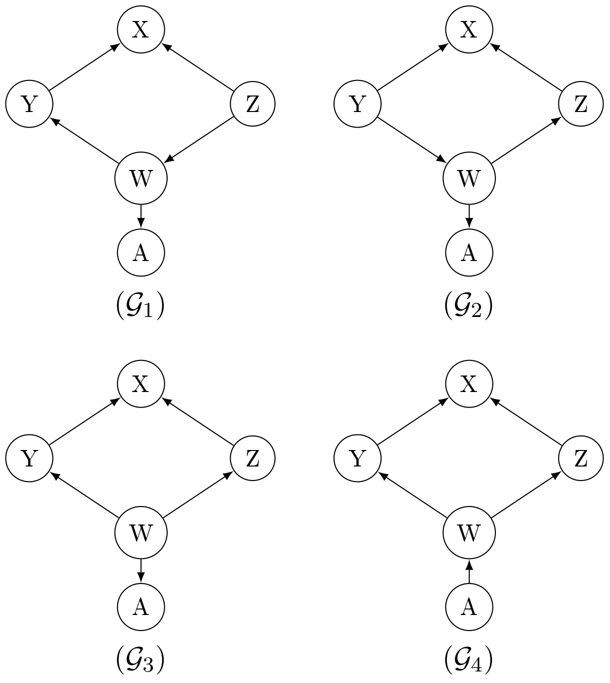


FIGURE 2.14: Four DAGs encoding the same set of independence assertions.

Given Definition 2.11, it is straightforward to see that the *d-separation* criterion enables to check whether some DAGs are *Markov equivalent* or not. In addition to the *d-separation* criterion, another graphical technique can be used to identify whether a set of BNs are equivalent: as we can observe in the example of Figure 2.14, all the DAGs share a common skeleton, i.e., they have the same graphical structure if the arcs are substituted by (undirected) edges. This implies that the set of trails between any two nodes X and Y are the same in both $\mathcal{G}_1$ and $\mathcal{G}_2$. By definition of d-separation, it is clearly a requirement for $\mathcal{G}_1$ and $\mathcal{G}_2$ to be Markov equivalent. Actually, if the DAGs did not share the same skeleton, there would exist at least one trail in one DAG that would not belong to the other DAG, which would in turn imply that both DAGs do not have the same set of conditional independences. But sharing the same skeletons is not sufficient to prove the Markov equivalence. A counterexample can be found in Figure 2.15 in which all the networks share the same skeleton but $\{X\} \perp_{\mathcal{G}} \{Z\}|\{Y\}$ in graph (a) whereas $\{X\} \not\perp_{\mathcal{G}} \{Z\}|\{Y\}$ in graph (b) (see the d-separation criterion). As can be noticed, the only difference between these two DAGs is the existence of a *v-structure* (converging arcs structure) at node Y in the second graph. This is precisely the additional information needed to assess Markov equivalence:

Definition 2.12 (v-structure)**.** *Let X, Y, Z be three nodes in a DAG $\mathcal{G}$. Then, in triple (X, Y, Z), Y is a v-structure if and only if the arcs $X \rightarrow Y$ and $Y \leftarrow Z$ belong to $\mathcal{G}$ but not arc $X \rightarrow Z$ nor arc $Z \rightarrow X$.*

In Figure 2.15.(b), node Y is thus a v-structure for triple (X, Y, Z), but W is not a v-structure for triple (Y, W, Z) because the graph contains arc $Z \rightarrow Y$. Now, we can present the Markov equivalence criterion:

Theorem 2.1. *Two DAGs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent if they share the same skeleton and the same set of v-structures.*

Referring to Theorem 2.1, the *Markov equivalence class* of a set of BNs can then be represented with a *partially directed graph (PDAG)* in which all the arcs not being part of any *v-structure* are substituted by (undirected) edges. The resulting partially directed graph can be completed by orienting the edges compelled to a specific direction (to avoid creating new v-structures). The resulting graph is then called a *completed partially directed acyclic graph (CPDAG)*. It should be noted that the CPDAG of an *equivalence Markov class* is unique. Figure 2.16 depicts an example of a CPDAG for the networks mentioned in Figure 2.14: the orientation of edges meeting at X is compelled since it corresponds to a *v-structure* while the rest of edges can be freely oriented.

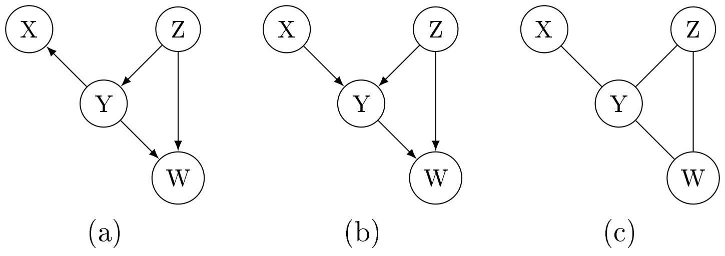


FIGURE 2.15: Two DAGs (a) and (b), and their skeleton (c).

If we go back to Subsection 2.2.2 and make the relation with the notion of *Markov equivalence class*, we can deduce that all graphs $\mathcal{G}_i$ that are *P-map* of a distribution P are necessarily *Markov equivalent.*

Theorem 2.2. *All P-map for a given distribution P, if they exist, belong to the same* Markov equivalence class.

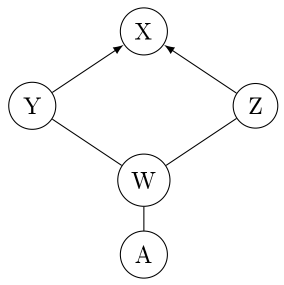


FIGURE 2.16: The Markov equivalence class of DAGs of figure 2.14

2.4 Probabilistic inference

In this section, we discuss how to perform inference (or belief propagation) in a BN. As discussed earlier, BNs allow a compact and expressive representation of joint probability distribution P. Through the specification of P, we are able to answer a large range of probabilistic queries which, as we shall see, enables us to perform many useful reasoning patterns, e.g., prediction, diagnosis, *etc.* All of the probabilistic reasoning performed with BNs somehow involve computing some conditional probability distribution of the form $P(\mathbf{Q}|\mathbf{E}=\mathbf{e})$, where $\mathbf{Q}$ denotes a set of *query (or target) variables* and $\mathbf{E}$ denotes the set of *evidence variables* on which observations $\mathbf{e}$ are available. By abuse of notation, this distribution is often denoted by $P(\mathbf{Q}|\mathbf{e})$. By definition, it can be computed as:

$$P(\mathbf{Q}|\mathbf{E}=\mathbf{e}) = \frac{P(\mathbf{Q},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})}. \tag{2.4}$$

As can be seen from this equation, it relies on the calculation of two joint probabilities, $P(\mathbf{Q},\mathbf{E}=\mathbf{e})$ and $P(\mathbf{E}=\mathbf{e})$. The former can be computed from P by summing out all the irrelevant variables $\mathbf{W}=\mathcal{X}\setminus(\mathbf{Q}\cup\mathbf{E})$. That is:

$$P(\mathbf{Q},\mathbf{E}=\mathbf{e}) = \sum_{\mathbf{w}\in\mathbf{W}} P(\mathbf{Q},\mathbf{W}=\mathbf{w},\mathbf{E}=\mathbf{e}).$$

Probability $P(\mathbf{E}=\mathbf{e})$ can be computed similarly from the above equation:

$$P(\mathbf{E}=\mathbf{e}) = \sum_{\mathbf{q}\in\mathbf{Q}} P(\mathbf{Q}=\mathbf{q},\mathbf{E}=\mathbf{e}).$$

In principle, by performing summations over irrelevant variables on the joint probability distribution P, it is possible to answer many probabilistic queries. However, this approach is not always tractable since the size of probability distribution P grows exponentially with respect to the number of variables n ($O(2^n)$ if all the variables are assumed to be Boolean). Hence, summing over the joint distribution P is very time consuming, and, actually, the inference task (i.e., answering queries of the above type) is known to be a $\mathcal{NP}$-hard problem.

In the rest of this section, we will discuss exact and approximate state-of-the-art inference algorithms that have been proposed to cope with the complexity of answering probabilistic queries with BNs. Among the exact methods, we can cite the *Variable Elimination algorithm* [Dec99]: the key idea of this algorithm consists in marginalizing out $\mathbf{W}$ one variable at a time from the joint distribution until the desired conditional probability $P(\mathbf{Q}, \mathbf{e})$ is computed. To be efficient, however, the joint distribution is never used as is. Instead, the algorithm starts from the pool of CPTs whose product is equal to the joint distribution P. Then, each time a variable needs to be marginalized out, the set of CPTs that contain this variable are removed from the pool, their product is computed and the CPT resulting from the marginalization of the variable from this product is inserted back into the pool. When no more variable needs to be marginalized out, the product of the CPTs remaining in the pool corresponds to probability $P(\mathbf{Q}, \mathbf{e})$. Depending on the query at hand, this algorithm tries to find the best elimination order of variables, i.e., the one which results in the lowest complexity of this process. This problem is also $\mathcal{NP}$-hard and, in practice, heuristics are exploited.

In [Pea88], a *Message passing* algorithm propagating information within the BN structure is proposed, but it produces mathematically correct results only in tree or polytree BN structures. To cope with more general BNs, i.e., BNs possibly containing undirected cycles, Pearl proposed a *cutset conditioning* algorithm. Essentially, the idea consists of instantiating a set of random variables (the cutset variables) in order to remove all the cycles: when a variable/node is instantiated, its outgoing arcs are actually not useful anymore and can thus be discarded. To produce correct results, it is needed to perform the message passing algorithm for each possible instantiation of the set of cutset variables. Unfortunately, this tends to increase exponentially the number of computations with the number of cutset variables, hence also with the number of cycles. There were attempts to reduce this increase, notably using *local cutset conditioning* [Die96, FJ00], but this kind of inference algorithm remains very time consuming. *Tree clustering*, also known as the *junction tree* algorithm [JOL89], is another exact method which derives a clique tree structure from the original graph $\mathcal{G}$ in order to be used to control the process of variable eliminations. This kind of algorithm has received a lot of attention because, in practice, it is very efficient. Different messages have been proposed, for instance in [JLO90], all messages represent (small) joint probability distributions whereas, in [Sha96, She97], messages represent conditional probabilities and, in [MJ99], messages correspond to sets of probabilities.

To summarize, the purpose of exact algorithms consists in reducing the amount of calculations required by the inference process by treating each variable in a local way or by grouping them into some “small” clusters. However, it should be noted that the complexity of exact methods increases exponentially w.r.t. the number of intertwined cycles in the BN, hence they reach their limit when the BN structure becomes too complex.

In this context, approximate inference algorithms can be considered as an efficient alternative because they are much less sensitive to the topology of BN. Since the results returned by this class of methods are not exact, their main issue is related to the quality of the obtained results which, in general, depend on the number of iterations budgeted by the algorithm. Among the most common approximate inference approaches, we can find those based on *Markov Chain Monte Carlo (MCMC)* [CC90, Pea87]. Their principle can be described as follows: they start with a random sample $\mathbf{q}_0$ for the set of query (target) variables $\mathbf{Q}$, which is consistent with evidence $\mathbf{E} = \mathbf{e}$. Then, iteratively, they create new samples $\mathbf{q}_i$ from $\mathbf{q}_{i-1}$ which tend to better represent distribution $P(\mathbf{Q}|\mathbf{e})$. They stop when the samples represent the stationary distribution $P(\mathbf{Q}|\mathbf{e})$. Such popular MCMC methods include *Gibbs sampling* and *Metropolis-Hastings*. Other methods such as *likelihood weighting* [SP90], *self importance sampling* [SP90], *particle filter* [KKR13] could also be used to perform an approximate inference.

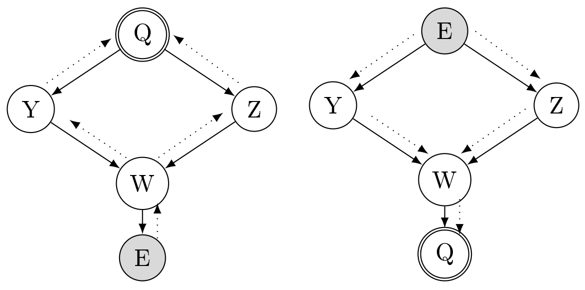
Before ending this section, it should be noted that the above list of inference algorithms is not exhaustive. There also exist algorithms based on recursive conditioning [AD03], on weighted model counting [BDTP03, SBK05, CD08], *etc.* The reader can also refer to [Pea88, KF09] for additional techniques.

2.5 Reasoning with BN

Once the BN is built, it can be used as a powerful tool for reasoning about beliefs in complex systems and for answering a large range of probabilistic queries. Next, we thus describe some of the most common types of queries used in real-world problems. Those include the computation of conditional probabilities *(CP)*, of most probable explanations *(MPE)* and maximum *a posteriori (MAP)*.

2.5.1 Conditional probability query (CP)

It is considered as the most simple and usual query. Given instantiations of some evidence variables $\mathbf{E} = \mathbf{e}$, the question asked in this query is the following: what is the distribution $P(\mathbf{Q}|\mathbf{E} = \mathbf{e})$, for some set of target variables $\mathbf{Q}$? Usually, $\mathbf{Q}$ is limited to a singleton, i.e., only one target random variable is of interest. This type of query is useful in many real-world applications stemming from many different domains. Take a medical diagnosis system as an example. In such a system, the doctor can be interested in calculating the probability distribution that a patient may have a cancer (or not) given some of his symptoms and/or some information about him, e.g., $P(\text{Cancer}|\text{Smoker=yes}, \text{Age=35})$. Here, $\mathbf{E} = \{\text{Smoker}, \text{Age}\}$ and $\mathbf{Q} = \{\text{Cancer}\}$. Such a query can be answered using the algorithms mentioned in the preceding section. This kind of query allows for two



(a) Diagnosis (b) Prognosis

FIGURE 2.17: Example of flow propagation in diagnosis and prognosis reasonings.

relevant reasoning patterns: diagnosis and prognosis. Figure 2.17 depicts the principles of diagnosis and prognosis propagations in the BN framework. In this example, evidence ($\mathbf{E}$) and query variables ($\mathbf{Q}$) are represented respectively with shaded circles and with double circles respectively. Diagnosis performs reasoning from consequences to causes (usually the opposite direction of the arcs in the BN). For example, in the nuclear field, observations about post nuclear accident variables (amount of iodine in the environment, containment state, *etc.*) can be used to update the expert's belief about the values of some other physical variables. Prognosis (or predictive) reasoning is performed from new information about causes to the new beliefs about consequences. Note that this reasoning usually occurs in the same direction as the arcs of the BN (the opposite of the diagnosis principle). For example, observations coming from reactor sensors about temperature and pressure may be used to update the expert's belief about the containment state (before being assessed).

2.5.2 Most probable explanation (MPE)

Unlike the preceding subsection, in the most probable explanation query, the goal is not to compute a posterior probability distribution $P(\mathbf{Q}|\mathbf{e})$ but to determine an instantiation $\hat{\mathbf{q}}$ of variables $\mathbf{Q} = \mathcal{X} \setminus \mathbf{E}$ with the highest posterior probability. In other words, computing the answer to this query amounts to determine:

$$\hat{\mathbf{q}} = \underset{\mathbf{q} \in \mathbf{Q}}{\text{Argmax}}\, P(\mathbf{Q} = \mathbf{q}, \mathbf{E} = \mathbf{e}).$$

In [Nil98], an efficient algorithm based on the junction tree algorithm is provided to answer this query, and even a more general one: finding the k most probable explanations, i.e., the k instantiations of $\mathbf{Q}$ with the highest posterior probabilities.

This class of query can be applied in many real-world applications. For example, it can be used to facilitate the diagnosis in the medical domain by allowing to find the most probable diseases given clinical tests' observations: the latter is considered as the set of evidence $\mathbf{E}$, while the former corresponds to the set of query variables $\mathbf{Q}$. Since a patient may suffer simultaneously from several diseases, the *MPE* query in such a case gives more relevant results than *CP*, since it exploits the compound effects of multiple diseases.

2.5.3 Maximum a posteriori (MAP)

Maximum a posteriori queries are a generalization of MPE in which $\mathbf{Q}$ is not necessarily equal to $\mathcal{X} \setminus \mathbf{E}$ anymore but it can be a subset of $\mathcal{X} \setminus \mathbf{E}$ [PD04]. So, if $\mathbf{W} = \mathcal{X} \setminus (\mathbf{Q} \cup \mathbf{E})$, then MAP consists in computing:

$$\hat{\mathbf{q}} = \operatorname*{Argmax}_{\mathbf{q} \in \mathbf{Q}} P(\mathbf{Q} = \mathbf{q} | \mathbf{E} = \mathbf{e}) = \operatorname*{Argmax}_{\mathbf{q} \in \mathbf{Q}} \sum_{\mathbf{w} \in \mathbf{W}} P(\mathbf{Q} = \mathbf{q}, \mathbf{W} = \mathbf{w} | \mathbf{E} = \mathbf{e}).$$

As maximizations and summations cannot commute, this type of query has a higher complexity than MPE because it adds constraints to the variables' elimination ordering.

2.6 Conclusion

In this chapter, we have introduced Bayesian networks and we have shown how they allow the compact description of large joint probability distributions. To achieve this result, Bayesian networks rely on conditional independence properties that can be read from their graphical structure. Therefore, we have presented the relationships between probabilistic conditional independence statements (those that actually hold in the joint probability distribution) and graphical conditional independence statements (those that can be derived from the BN graphical structure). Notably, the notions of I-maps, D-maps and Perfect maps and the semi-graphoid and graphoid axioms have been presented. Finally, we have introduced the *d*-separation criterion, which is the graphical conditional independence property actually represented by BNs.

Based on this criterion, we could introduce the notion of Markov blanket, the set of nodes that make another node independent from the rest of the BN. Markov blankets will prove to be important for learning the structure of BNs from datasets. We observed that several BNs can represent precisely the same set of conditional independence statements, so we presented the notion of Markov equivalence class: all the graphs that belong to the same class represent exactly the same set of conditional independence statements. We also discussed a simple criterion to determine whether different graphs belong to the same class (this is the case when they share the same skeleton and the same set of v-structures). Markov equivalence classes can also prove very useful for local search BN structure learning algorithms because they can allow them to avoid searching neighborhoods that belong to the same class, i.e., that do not improve the quality of the learnt BNs.

Finally, we discussed how BNs can be exploited for probabilistic reasoning. We therefore showed which probabilistic queries were of interest in practical applications and we presented the main classes of algorithms used to answer these queries.

Chapter 3

Learning Bayesian networks

Until now, we have discussed the main theoretical properties of BNs by assuming that their graphical component (DAG) as well as their parameters (CPTs) are completely specified. In this chapter, we study how BNs can be constructed. Two main possible approaches exist in the literature to perform such a task. The first one consists in hand-constructing the model, typically by exploiting some expert knowledge of the domain. Unfortunately, eliciting BNs from experts is not a trivial task since it becomes extremely laborious and time consuming when the size of the network becomes large. To alleviate this issue, a second alternative has been proposed, which consists in automatically learning BNs from data without needing the knowledge of the experts. In the past twenty years, numerous efforts have been devoted to learn automatically BNs —both their graphical structure and the parameters of their conditional probability tables— from datasets [CH92a, Hec95, HGC95, Pea88, SGS01, TBA06]. The advantage of these approaches results from their ability to construct BNs that precisely capture the properties of the distribution P that generated the data, even when it is high-dimensional.

It must be emphasized that the learning process is governed by two features. The first one is related to the amount of data available from which the model will be constructed as well as from the features of those data (the presence or lack of continuous variables or of deterministic relationships among random variables, the *a priori* over the parameters of the random variables or over the possible DAG structures, *etc.*). The performance metric by which the quality of the model is evaluated is usually chosen on the basis of this dimension. This metric induces the objective function that is optimized throughout the learning process. The second dimension concerns the goal for which the BN is learnt from dataset $\mathcal{D}$, i.e., whether it will be used to subsequently perform probabilistic inference or for a classification task for instance. This actually impacts on what needs be learnt: do we really need to learn a full BN or just the part related to a single target variable (the class). Given these two dimensions, several interesting questions emerge. The most common questions are related to the quality of the returned BN given the chosen objective function and to the set of assumptions under which the BN learning task remains both feasible and efficient. Actually, the learning process is not always tractable in practice as we shall see, notably in the next chapter.

In this chapter, we shall focus our discussion on the problem of learning the parameters and the structure of a BN from **discrete data** (i.e., the data were generated from a multivariate distribution over discrete random variables). We will discuss for each learning process (parameters and/or structure) the set of assumptions that must be satisfied before performing each of them. Then, we will mention the most common state-of-the-art algorithms dedicated to BN learning.

3.1 Parameters learning

3.1.1 Preliminary definitions

Consider a BN $\mathcal{B} = (\mathcal{G}, \mathbf{\Theta})$, where $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ is the BN's DAG structure. For every variable $X_i \in \mathcal{X}$, let r_i and q_i denote the domain sizes of X_i and of its parents in the BN respectively (by abuse of notation, $q_i = 1$ when $\mathbf{Pa}(X_i) = \emptyset$). Let $x_{i,j}$ denote jth value of the domain of X_i, i.e., the jth value that X_i can take. Similarly, let $\mathbf{pa}(X_i)_j$ denote the jth value of the domain of the set of random variables $\mathbf{Pa}(X_i)$.

The set of parameters $\mathbf{\Theta}$ of the BN can be decomposed as follows:

$$\begin{aligned} \mathbf{\Theta} &= \{\theta_i\}_{i=1}^{n} \\ \theta_i &= \{\theta_{i,j}\}_{j=1}^{q_i}, \;\; i = 1, ..., n \\ \theta_{i,j} &= \{\theta_{i,j,k}\}_{k=1}^{r_i}, \;\; j = 1, ..., q_i, \;\; i = 1, ..., n \end{aligned} \tag{3.1}$$

In these decompositions, θ_i denotes the CPT $P(X_i|\mathbf{Pa}(X_i))$ assigned to node X_i, i.e., it is the set of parameters that describe the probability distribution of X_i given its parents in $\mathcal{G}$. Parameter $\theta_{i,j}$ specifies the distribution of X_i conditioned on the jth value $\mathbf{pa}(X_i)_j$ taken by its set of parents, i.e., $P(X_i|\mathbf{Pa}(X_i) = \mathbf{pa}(X_i)_j)$. Finally, the parameter $\theta_{i,j,k}$ denotes the probability $P(X_i = x_{i,k}|\mathbf{Pa}(X_i) = \mathbf{pa}(X_i)_j)$, i.e., the probability that X_i has taken its kth value given that its set of parents has taken its jth value.

In the following, we will discuss the problem of parameter estimation for BNs whose structures are known *a priori*. We assume that data under which parameters will be estimated are fully observed. Given a structure $\mathcal{G}$ and dataset $\mathcal{D}$, the goal here consists in identifying the optimal set of parameter values $\mathbf{\Theta} = \{\theta_1, \theta_2, ..., \theta_n\}$ of the network. Usually, this is done by approximating $\mathbf{\Theta}$ as accurately as possible to the optimal set of parameters $\hat{\mathbf{\Theta}}$. To do this, the different values that $\mathbf{\Theta}$ may take are usually evaluated using a specific objective function. In the literature, different objective functions have been proposed, depending on some hypotheses done and the approaches followed (*maximum* *likelihood estimation*, *maximum a posteriori estimation*, *Bayesian approaches*). Before listing the principles of these methods, we introduce the set of assumptions under which they are supposed to work effectively.

1. **Samples independence:** the elements of dataset $\mathcal{D}$ are assumed to be mutually independent and identically distributed (*i.i.d. hypothesis*). Mutual independence means that no subset of records in $\mathcal{D}$ can provide any information on any other disjoint subset. Identical distribution means that all the records in $\mathcal{D}$ were generated from the same probability distribution.

2. **Parameters independence:** this assumption can be decomposed into two levels: global and local parameters independence. Parameters $\mathbf{\Theta}$ are said to satisfy global independence if they can be expressed as follows:

$$\pi(\mathbf{\Theta}) = \prod_{i=1}^{n} \pi(\theta_i), \tag{3.2}$$

where π represents the *a priori* distribution over all the possible parameter sets $\mathbf{\Theta}$. In other words, the parameters θ_i of random variable X_i are independent from those (θ_j) of any other random variable X_j. Regarding the local parameters independences, the following expression holds:

$$\pi(\theta_i) = \prod_{j=1}^{q_i} \pi(\theta_{i,j}). \tag{3.3}$$

In other words, the parameters $\theta_{i,j}$ for the jth value of X_i's parents are independent from the parameters $\theta_{i,k}$ for the other values of X_i's parents.

3. **Parameters modularity:** for two given graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, parameters modularity states that if a node X_i has the same parents in both networks, then the conditional probability distribution $P(X_i|\mathbf{Pa}(X_i))$ of this node shall be identical in both networks.

4. **Dirichlet prior:** priors about $\theta_{i,j}$ values are expressed by a *Dirichlet density* function. Given *a Dirichlet* distribution specified by a set of r_i hyperparameters $(\alpha_1, ..., \alpha_{r_i})$, the prior $\pi(\theta_{i,j})$ is defined by the following density function:

$$Dirichlet(\theta_{i,j}|\alpha_1, ..., \alpha_{r_i}) = \frac{\Gamma(\sum_{k=1}^{r_i} \alpha_k)}{\prod_{k=1}^{r_i} \alpha_k} \prod_{k=1}^{r_i} \theta_{i,j,k}^{\alpha_k - 1} \tag{3.4}$$

where $\Gamma(\cdot)$ represents the Gamma function.

3.1.2 Maximum likelihood estimation

This approach relies on the use of the *likelihood* function as an evaluation criterion to identify the best set of parameters' values w.r.t. dataset $\mathcal{D}$:

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{Argmax}} \, L(\mathcal{D}|\mathbf{\Theta}) \tag{3.5}$$

where $L(\cdot|\cdot)$ denotes the *likelihood* function.

For a better understanding of the principle of this approach, we consider an example of thumbtack tossing experiment inspired from [KF09].

Example 3.1. *After flipping a thumbtack in the air, it comes to land as either head* (H) *or tail* (T). *The thumbtack experiment is repeated several times in order to obtain a dataset containing the observations (*H *or* T*) resulting from many flippings. Given the resulting dataset* $\mathcal{D}$*, the purpose of the parameters learning process consists in finding a good approximation of* $\mathbf{\Theta}$ *that describes accurately the probability of obtaining* H *or* T*, hereafter denoted respectively by* θ_H *and* θ_T*. Suppose that after tossing 100 times the thumbtack, 47 came up heads and the rest were tails. In such a case, it is natural to suggest that the best estimation of* θ_H *is 0.47 and* $\theta_T = 1 - 0.47 = 0.53$*. If we decided to set for instance* θ_H *to 0.01, then the chance of obtaining 47 heads in this experiment would have been very low, hence far from reality. Maximum likelihood estimation would have resulted in* θ_H *being equal to 0.47.*

One important question may arise about the choice of the $\mathbf{\Theta}$ values in the previous experiment: how can we evaluate whether $\mathbf{\Theta}$ is a good predictor of the given thumbtack tossing data? To answer this question, a possible approach is to score the different values of parameters $\mathbf{\Theta}$ with the likelihood of obtaining dataset $\mathcal{D}$ given $\mathbf{\Theta}$. In this context, it is natural to select the parameter set $\mathbf{\Theta}$ that maximizes the *likelihood* of the data. Such an approach is called *Maximum Likelihood Estimation (MLE)*. Assuming that dataset $\mathcal{D}$ contains N records representing the realizations of a variable X, which are supposed to be i.i.d., the *likelihood* of $\mathcal{D}$ given $\mathbf{\Theta}$ is calculated as follows:

$$L(\mathcal{D}|\mathbf{\Theta}) = P(\mathcal{D}|\mathbf{\Theta}) = \prod_{m=1}^{N} P(x^{(m)}|\mathbf{\Theta}) \tag{3.6}$$

where $x^{(m)}$ represents the observed value of X in the mth record of $\mathcal{D}$.

Since the probability distribution of the thumbtack flipping example corresponds to a binomial distribution, the *likelihood* of the dataset $\mathcal{D}$ given $\mathbf{\Theta}$ is:

$$L(\mathcal{D}|\mathbf{\Theta}) = \theta_H^{47}(1 - \theta_H)^{100-47}. \tag{3.7}$$

It must be emphasized that the parameter values returned by *MLE* must be "legal", that is, they should represent a probability distribution. This means that, in Equation (3.5), only the $\mathbf{\Theta}$'s that correspond to a probability distribution are taken into account.

Given the principle of the *MLE* technique, we can now generalize it to the context of parameters estimation in a BN. It may be pointed out that the principle of parameters estimation in BNs remains the same as for a single variable since it tries to find the set of parameters' values of the BN that maximize the overall *likelihood* of dataset $\mathcal{D}$. The only difference is that instead of determining the parameters' values for one variable, we calculate them for every CPT that corresponds to every node X_i given its parents in $\mathcal{G}$, i.e., $P(X_i = x_{i,k}|\mathbf{Pa}(X_i) = \mathbf{pa}(X_i)_j) = \theta_{i,j,k}$. Given a BN, the *likelihood* for a set of parameters $\mathbf{\Theta}$ is calculated as follows:

$$\begin{aligned} L(\mathcal{D}|\mathbf{\Theta}) &= \prod_{m=1}^{N} P(\mathcal{X}^{(m)}|\mathbf{\Theta}) \\ &= \prod_{i=1}^{n}\prod_{m=1}^{N} P(X_i = x_{i,k(m)}|\mathbf{Pa}(X_i) = \mathbf{pa}(X_i)_{j(m)}, \mathbf{\Theta}) \\ &= \prod_{i=1}^{n}\prod_{m=1}^{N} \theta_{i,j(m),k(m)} \end{aligned} \tag{3.8}$$

where $x_{i,k(m)}$ and $\mathbf{pa}(X_i)_{j(m)}$ represent the values of X_i and of its parents in the mth record of $\mathcal{D}$ respectively, and where $\theta_{i,j(m),k(m)}$ represents the corresponding parameter in CPT $P(X_i|\mathbf{Pa}(X_i))$.

Let $N_{i,j,k}$ denote the number of records with configuration $(X_i = x_{i,k}, \mathbf{Pa}(X_i) = \mathbf{pa}(X_i)_j)$ in $\mathcal{D}$. Then the *likelihood* given by the above expression can be rewritten as follows:

$$L(D|\mathbf{\Theta}) = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}} . \tag{3.9}$$

The logarithm of Equation (3.9) is therefore:

$$LL(D|\mathbf{\Theta}) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{i,j,k} \log \theta_{i,j,k}. \tag{3.10}$$

As the logarithm is a strictly increasing function, the solution $\hat{\mathbf{\Theta}} = \{\hat{\theta}_1, \ldots, \hat{\theta}_n\}$ of Equation (3.5) also maximizes the above *log-likelihood*. In addition, by global parameter independence, each θ_i can be optimized separately from the other θ_j's. Therefore:

$$\hat{\theta}_i = \underset{\theta_i}{\text{Argmax}} \sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{i,j,k} \log \theta_{i,j,k}, \quad \text{for all } i \in \{1, \ldots, n\}.$$

Remember that $\theta_i = \{\theta_{i,1}, \dots, \theta_{i,j}, \dots, \theta_{i,q_i}\}$. Now, by local parameter independence, we have that each set of parameters $\theta_{i,j}$ is independent of the other sets $\theta_{i,j'}$, $j' \neq j$. As a consequence, we also have that:

$$\hat{\theta}_{i,j} = \underset{\theta_{i,j}}{\text{Argmax}} \sum_{k=1}^{r_i} N_{i,j,k} \log \theta_{i,j,k}, \quad \text{for all } i \in \{1, \dots, n\}, j \in \{1, \dots, q_i\}. \tag{3.11}$$

So, all these quantities can be optimized separately: they are actually independent sub-problems. Finally, note that $\theta_{i,j} = \{\theta_{i,j,k}\}_{k=1}^{r_i}$ are the parameters of a probability distribution over random variable X_i. This imposes the following constraints on $\theta_{i,j,k}$: i) all the $\theta_{i,j,k}$ are non-negative; and ii) $\sum_{k=1}^{r_i} \theta_{i,j,k} = 1$. The second constraint can be equivalently expressed as $\theta_{i,j,r_i} = 1 - \sum_{k=1}^{r_i - 1} \theta_{i,j,k}$. After substituting θ_{i,j,r_i} by this expression in Equation (3.11), we get:

$$\hat{\theta}_{i,j} = \underset{\theta_{i,j}}{\text{Argmax}} \left[\left(\sum_{k=1}^{r_i-1} N_{i,j,k} \log \theta_{i,j,k} \right) + N_{i,j,r_i} \log \left(1 - \sum_{k=1}^{r_i-1} \theta_{i,j,k} \right) \right].$$

If we denote by $LL_{i,j}(\mathcal{D}|\theta_{i,j})$ the function inside square brackets in the above equation, then, the optimal solution $\hat{\theta}_{i,j}$ is obtained when:

$$\frac{\partial LL_{i,j}(\mathcal{D}|\theta_{i,j})}{\partial \theta_{i,j,k}} = \frac{\partial LL(\mathcal{D}|\boldsymbol{\Theta})}{\partial \theta_{i,j,k}} = 0, \quad \text{for all } k \in \{1, \dots, r_i\}.$$

Now, it is easy to see that:

$$\begin{aligned} \frac{\partial LL_{i,j}(\mathcal{D}|\theta_{i,j})}{\partial \theta_{i,j,k}} &= \frac{N_{i,j,k}}{\theta_{i,j,k}} - \frac{N_{i,j,r_i}}{\left(1 - \sum_{k'=1}^{r_i-1} \theta_{i,j,k'}\right)} \\ &= \frac{N_{i,j,k}}{\theta_{i,j,k}} - \frac{N_{i,j,r_i}}{\theta_{i,j,r_i}}. \end{aligned} \tag{3.12}$$

From this result, we can immediately conclude that Equation (3.12) is equal to 0 when $\theta_{i,j,k}$ verifies:

$$\frac{N_{i,j,k}}{\theta_{i,j,k}} = \frac{N_{i,j,r_i}}{\theta_{i,j,r_i}}, \forall k \in \{1, ..., r_i - 1\}. \tag{3.13}$$

Therefore:

$$\frac{N_{i,j,1}}{\theta_{i,j,1}} = \frac{N_{i,j,2}}{\theta_{i,j,2}} = \dots = \frac{N_{i,j,r_i}}{\theta_{i,j,r_i}} = \frac{\sum_{k=1}^{r_i} N_{i,j,k}}{\sum_{k=1}^{r_i} \theta_{i,j,k}} = \sum_{k=1}^{r_i} N_{i,j,k}.$$

Substituting this result into Equation (3.12), we get:

$$\hat{\theta}_{i,j,k}^{MLE} = \frac{N_{i,j,k}}{\sum_{k'=1}^{r_i} N_{i,j,k'}}.$$

Exploiting the above equations, the maximum *likelihood* parameters values can be found by simply counting how many times each configuration of each X_i and $\mathbf{Pa}(X_i)$ occurs in dataset $\mathcal{D}$. To guarantee the correctness of this approach, a large dataset is however needed, else the estimation might be very approximate.

3.1.3 Bayesian estimation

In this approach, it is assumed that there exists a joint probability distribution P over all the possible pairs $(\mathcal{D}, \mathbf{\Theta})$, i.e., over all the possible datasets and all the possible sets of parameters $\mathbf{\Theta}$. A consequence of this assumption is that there also exists a prior distribution $\pi(\mathbf{\Theta}) = P(\mathbf{\Theta}) = \sum_{\mathcal{D}} P(\mathcal{D}, \mathbf{\Theta})$ over the possible values of $\mathbf{\Theta}$. This prior and the dataset $\mathcal{D}$ together are used to assign a posterior probability to the different values of $\mathbf{\Theta}$ using a probabilistic reasoning based on Bayes rule:

$$P(\mathbf{\Theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathbf{\Theta})\pi(\mathbf{\Theta})}{P(\mathcal{D})} \propto P(\mathcal{D}|\mathbf{\Theta})\pi(\mathbf{\Theta}). \tag{3.14}$$

Unlike the *MLE* approach, in the *Bayesian approach* $\mathbf{\Theta}$ is treated as a random variable over which we maintain a probability distribution $\pi(\mathbf{\Theta})$. In Equation (3.14), factor $P(\mathcal{D}|\mathbf{\Theta})$ corresponds to the *likelihood function* (discussed earlier). Factor $\pi(\mathbf{\Theta})$ is the *a priori* distribution over the different parameter values $\mathbf{\Theta}$. The denominator is a normalizing factor which corresponds to the marginal *likelihood* of the data $P(\mathcal{D}) = \int_{\mathbf{\Theta}} P(\mathcal{D}|\mathbf{\Theta})\pi(\mathbf{\Theta})d\mathbf{\Theta}$. But this term can be discarded from our search of the best $\mathbf{\Theta}$ since it is a constant (it is independent of $\mathbf{\Theta}$). In other words, the *Bayesian estimation* approach can be seen as a kind of belief $\pi(\mathbf{\Theta})$ updated after we have observed dataset $\mathcal{D}$. Since the posterior distribution of parameters' values $\mathbf{\Theta}$ is a product of the *likelihood* $P(\mathcal{D}|\mathbf{\Theta})$ and of the prior $\pi(\mathbf{\Theta})$, it would be logical that both of them have the same form. Recall that the corresponding *likelihood* function for a multinomial distribution is given by the following expression:

$$P(D|\mathbf{\Theta}) = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}}. \tag{3.15}$$

Given the previous equation, it turns out that the *Dirichlet distribution* prior has a compatible form with this *likelihood* expression (it is a conjugate prior). For the set of random variables $\mathcal{X}$ in the BN, the prior distribution over the possible BN parameters $\mathbf{\Theta}$ values using the *Dirichlet distribution* (with hyperparameters $\alpha_{i,j,k} + 1$) is expressed as follows:

$$\pi(\mathbf{\Theta}) \propto \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} (\theta_{i,j,k})^{\alpha_{i,j,k}}, \tag{3.16}$$

where $\alpha_{i,j,k} > -1$. As a result, the posterior probability $P(\mathbf{\Theta}|\mathcal{D})$ is also Dirichlet:

$$\begin{aligned} P(\mathbf{\Theta}|\mathcal{D}) &\propto P(\mathcal{D}|\mathbf{\Theta})\pi(\mathbf{\Theta}) \\ &\propto \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i} (\theta_{i,j,k})^{N_{i,j,k}+\alpha_{i,j,k}}. \end{aligned} \tag{3.17}$$

Note that Equation (3.17) is equivalent to the *likelihood* function if and only if the prior $\pi(\mathbf{\Theta})$ is uniform (we can interpret such a prior as an uninformative one).

Similarly to the MLE approach, differentiating the log of $P(\mathbf{\Theta}|\mathcal{D})$ in Equation (3.17) w.r.t. $\theta_{i,j,k}$ results in:

$$\begin{aligned}\frac{\partial \log P(\mathbf{\Theta}|\mathcal{D})}{\partial \theta_{i,j,k}} &= \left(\frac{N_{i,j,k}+\alpha_{i,j,k}}{\theta_{i,j,k}}\right) - \left(\frac{N_{i,j,r_i}+\alpha_{i,j,r_i}}{1-\sum_{k'=1}^{r_i-1}\theta_{i,j,k'}}\right)\\ &= \left(\frac{N_{i,j,k}+\alpha_{i,j,k}}{\theta_{i,j,k}}\right) - \left(\frac{N_{i,j,r_i}+\alpha_{i,j,r_i}}{\theta_{i,j,r_i}}\right).\end{aligned} \tag{3.18}$$

By following the same computation principle as in the *MLE* approach, we obtain:

$$\hat{\theta}_{i,j,k}^{MAP} = \frac{N_{i,j,k}+\alpha_{i,j,k}}{\sum_{k'=1}^{r_i} N_{i,j,k'}+\alpha_{i,j,k'}}. \tag{3.19}$$

Another alternative to make *a Bayesian estimation* of $\theta_{i,j,k}$ consists in computing the expectation w.r.t. the posterior probability $\mathbb{E}_{P(\mathbf{\Theta}|\mathcal{D})}[\mathbf{\Theta}]$ instead of its maximum:

$$\begin{aligned}\hat{\theta}_{i,j,k}^{EAP} &= \mathbb{E}_{P(\mathbf{\Theta}|\mathcal{D})}[\mathbf{\Theta}] = \int_{\mathbf{\Theta}} \mathbf{\Theta} P(\mathbf{\Theta}|\mathcal{D}) d\mathbf{\Theta}\\ &= \frac{N_{i,j,k}+\alpha_{i,j,k}+1}{\sum_{k'=1}^{r_i} N_{i,j,k'}+\alpha_{i,j,k'}+1}.\end{aligned} \tag{3.20}$$

To summarize, Table 3.1 provides the main differences between the *MLE* and the *Bayesian* approaches.

Criteria	Maximum *likelihood*	*Bayesian approach*
Estimation	Maximization of $P(\mathcal{D}|\mathbf{\Theta})$	Maximization of $P(\mathbf{\Theta}|\mathcal{D})$
Complexity	Solved analytically	Solved analytically
Prior	Without prior	With prior

TABLE 3.1: Main differences between the *MLE* and the *Bayesian* approaches.

3.2 Structure learning

In this section we shall focus our discussion on the problem of BN structure learning from data. Here, we will not give an exhaustive list of state-of-the-art structure learning algorithms. Instead, we will present an overview on the most prevalent algorithms in the literature and we will focus mainly on those related to our works.

Until now, we have only seen how BNs represent joint probability distributions and allow reasoning under uncertainty. The probability distribution is represented by means of a structure that describes the set of dependences/independences among random variables in a compact and efficient manner. This constitutes one of the main advantages of BNs since the independence properties of the probability distributions can be easily read from the graph through the graphical criterion called *d-separation* described in the preceding chapter. Now, there remains to learn this structure from data. This raises issues: how can we guarantee the correctness of the learnt graph, i.e., that it represents correctly the dependences/independences among the random variables? This question suggests that some graphs are better than others because they represent more or less these dependences/independences. As a consequence, structure learning can be thought of as an optimization problem: find the graph that "best" represents the dependences/independences of the distribution that generated dataset $\mathcal{D}$. Another question of interest: which hypotheses are needed to ensure the correctness of the learning process? As we shall see, notably in the next chapter, there exist datasets for which checking this correctness is not easy. Even in cases where it is easy, is structure learning in general an easy task? the literature shows that this task is $\mathcal{NP}$-hard in general. This explains why most of the algorithms are not designed to find the optimal solution but rather try to approximate it, essentially relying on search algorithms that try to limit as much as possible the search space (the size of the DAG structures being superexponential in the number of nodes in the graph). As such, only local maxima are returned by such algorithms. Nevertheless, we should remember that the quality of the learnt structure plays an important role in the accuracy of the BN. If the dependences between variables are not well learnt, the resulting probability distribution will be far from the correct one. This is the reason why this task must be addressed with a very high accuracy, since a simple arc omission or a wrong orientation can lead to a drastic change in the semantics of the BN. To illustrate this problem, let us consider a simple example of two possible orientations of a BN with four variables, as shown in Figure 3.1. In this example, the only difference between $\mathcal{G}_1$ and $\mathcal{G}_2$ is the orientation of edge $Y - W$. As can be seen, two different orientations of the single edge $Y - W$ leads to the disappearance of a *v-structure* $Y \rightarrow W \leftarrow Z$ from $\mathcal{G}_1$ and the creation of another one $X \rightarrow Y \leftarrow W$ in $\mathcal{G}_2$. By using the *d-separation* criterion, the conditional independences involved in this change can be mentioned as follows:

- $\mathcal{G}_1 \Rightarrow (\{X\} \perp_{\mathcal{G}} \{W\}|\{Y\}) \wedge (\{X, Y\} \perp_{\mathcal{G}} \{Z\})$,
- $\mathcal{G}_2 \Rightarrow (\{Y\} \perp_{\mathcal{G}} \{Z\}|\{W\}) \wedge (\{Z, W\} \perp_{\mathcal{G}} \{X\})$,

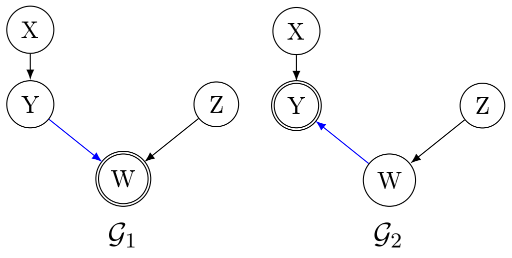


FIGURE 3.1: The impact of a single edge orientation modification on the semantic of the BN.

In general terms, the task of BN structure learning consists in finding the correct edges and their respective orientations according to some evaluation metrics that depends on the data. As mentioned in [Pea88], this task can be tackled efficiently (at least approximately) with many algorithms that, given a set of assumptions, guarantee the accuracy of the BN structure.

We begin this section by listing the set of assumptions made by the majority of BN structure learning algorithms. Then, we provide an overview of the different methods dedicated to BN structure learning from data.

3.2.1 Assumptions made by Bayes net structure learning algorithms

The main assumptions under which the BN learning task can be addressed effectively are the following: *faithfulness*, *causal sufficiency*, and *causal Markov.*

1. ***Faithfulness:*** distribution P is decomposable w.r.t. a perfect map $\mathcal{G}$. This assumption is most useful. Actually, the graph we wish to learn shall be an I-map because the structure of a BN is always an I-map. So all the independences present in the graph shall also be independences in distribution P. But, when learning from data, the latter only provide independences in P and, from them, we shall infer independences in $\mathcal{G}$, which makes $\mathcal{G}$ a D-map of P. As a consequence, $\mathcal{G}$ shall be both an I-map and a D-map, hence a perfect map: given three disjoint sets of variables $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, we shall therefore have:

$$\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y}|\mathbf{Z} \Longleftrightarrow \mathbf{X} \perp_{P} \mathbf{Y}|\mathbf{Z}. \qquad (3.21)$$

2. ***Causal sufficiency:*** it is also called the assumption of *completeness*. It states that for each pair of variables X_i and X_j in $\mathcal{X}$, there are no common unobserved (or latent) parents in $\mathcal{G}$. In other words, the variables $\mathcal{X}$ in a given dataset $\mathcal{D}$ are sufficient to learn the overall relationships between these variables.

3. ***Causal Markov assumption:*** it is satisfied if and only if, according to the distribution P, every node $X_i \in \mathcal{X}$ is independent of its non-descendants given its parents in $\mathcal{G}$ (see Definition 2.3 in Chapter 2).

3.2.2 Preliminary definitions

Given a dataset $\mathcal{D}$, the purpose of the BN structure learning task is to summarize graphically the set of conditional independence relationships among the variables that hold in distribution P. In this section, we assume that the BN learning task is performed under the following conditions:

- we restrict ourselves to the study of discrete variables, i.e., variables whose domain size is finite. Therefore, local probability distributions are represented by multidimensional tables;
- all the assumptions mentioned in Section 3.2.1 are satisfied;
- all data are fully observed, i.e., in every record of the dataset, all the variables have been assigned some value.

In the rest of this section, we will provide an illustration of the state-of-the-art BN structure algorithms. These algorithms can be divided into three main groups:

- **constraint-based approaches**: this class of algorithms relies on the use of statistical independence tests to find a DAG $\mathcal{G}$ for which the *local Markov property* entails the existing conditional independence assertions in P [SGS01, Pea00] (and only them under the faithfulness assumption).
- **score-based approaches**: using a scoring function, the problem of BN learning is transformed into a search problem for the structure that maximizes this function. At each iteration, the algorithms of this class examine all the possible local changes that may be performed on the current structure and then choose the one with the highest value of the scoring function. The process is iterated until a local optimum is reached.
- **Hybrid approaches:** they try to combine the advantages and overcome the limitations of the two above approaches: they start by learning a first graphical structure called a *skeleton* using a constraint-based approach and, then, starting from a graph induced by this skeleton, they carry out a score-based approach to find the best BN's DAG [TBA06, vDvdGT03a].

3.2.3 Constraint-based approaches

Constraint-based algorithms [SGS01, Pea00] attempt to learn a BN structure that accurately represents the conditional dependences/independences of the underlying probability distribution. In other words, given a distribution P, these algorithms try at each iteration to answer the following question: "does P satisfies the independence assertion $\{X\} \perp_P \{Y\}|\mathbf{Z}$?". The binary decision (yes/no) returned by a statistical test (which depends on a significance level α) for this question indicates whether an edge should exist or not between X and Y in $\mathcal{G}$. Therefore, the accumulation of all of these binary answers lead to the construction of an undirected skeleton where the undirected edge between any pair of variables indicates a direct probabilistic dependence. Besides the binary answers, statistical tests can provide another valuable information by means of the separating sets composed by conditioning variables $\mathbf{Z}$ that *d-separate* two variables X and Y. Given the independence assertion $\{X\} \perp_P \{Y\}|\mathbf{Z}$, the separating set of this example is denoted as follows:

$$SepSet_{XY} = \mathbf{Z}. \tag{3.22}$$

The learnt skeleton together with the discovery of the separating sets can correctly[1] recover the true *Complete Partially Directed Acyclic Graph (CPDAG)* [SGS01] which represents *the Markov equivalence class* of the BN (as defined in Section 2.3 of the preceding chapter). Many algorithms have been proposed in the literature to tackle this problem. They can be defined w.r.t. three criteria: (i) the type of statistical test used to determine conditional dependences/independences; (ii) the heuristics used to construct and orient the skeleton; (iii) the technique used to enhance the reliability of the statistical tests[2].

In the following, we will begin by presenting the most common used statistical tests to learn the skeleton. Then, we will introduce the basic versions of constraint-based approaches which constitute the PC and IC algorithms [SGS01, Pea00]. At the end of this part, we will give an overview on some extensions of constraint-based approaches that have been proposed in the literature.

3.2.3.1 Statistical tests for conditional independence

As mentioned earlier, constraint-based algorithms rely on statistical tests to compute conditional independences between random variables. In practice, there exist several types of statistical hypothesis tests that can be used to compute dependences (or correlations) between variables, hence allowing to learn the BN skeleton. All of these tests share a common principle which consists in the confrontation of two hypotheses to quantify the dependence between variables X_i and X_j given a set of variables $\mathbf{X}_k$:

- $\mathcal{H}_0$: X_i and X_j are conditionally independent given $\mathbf{X}_k$,
- $\mathcal{H}_1$: X_i and X_j are not conditionally independent given $\mathbf{X}_k$,

Given these hypotheses —$\mathcal{H}_0$ and $\mathcal{H}_1$ are called respectively the null and the alternative hypotheses—, the statistical test enables to decide whether to accept or reject the alternative hypothesis ($\mathcal{H}_1$) according to a significance level α. The most common independence tests used by constraint-based algorithms for multinomial distributions are Chi-square (χ^2) and G-square (G^2) based. Both of these tests are computed from a contingency table containing the number of occurrences of each instantiation of the variables

[1]This correctness is guaranteed under the following assumptions: *faithfulness, causal sufficiency and causal Markov.*

[2]They can be subject to error of type II (false negative edges), especially when the size of the conditioning set $\mathbf{Z}$ is large.

in the dataset. Given two discrete variables X_i and X_j, and a set $\mathbf{X}_k$ of other discrete variables, the χ^2 test compares the following models:

- the observed model $P_o = P(X_i, X_j|\mathbf{X}_k)$,
- the theoretical model (assuming independence) $P_t = P(X_i|\mathbf{X}_k) \times P(X_j|\mathbf{X}_k)$.

If $N_{i,j,k}$, $N_{i,k}$ and N_k denote the number of occurrences of triples $(X_i = x_i, X_j = x_j, \mathbf{X}_k = \mathbf{x}_k)$, pairs $(X_i = x_i, \mathbf{X}_k = \mathbf{x}_k)$ and singletons $(\mathbf{X}_k = \mathbf{x}_k)$ in the dataset respectively, then, provided that the latter is sufficiently large, we have that:

$$P(X_i = x_i, X_j = x_j, \mathbf{X}_k = \mathbf{x}_k) \approx \frac{N_{i,j,k}}{N},$$
$$P(X_i = x_i, \mathbf{X}_k = \mathbf{x}_k) \approx \frac{N_{i,k}}{N} \quad P(\mathbf{X}_k = \mathbf{x}_k) \approx \frac{N_k}{N},$$

where N is the number of records in the database. As a consequence,

$$P(X_i = x_i, X_j = x_j|\mathbf{X}_k = \mathbf{x}_k) \approx \frac{N_{i,j,k}}{N_k},$$
$$P(X_i = x_i|\mathbf{X}_k = \mathbf{x}_k) \approx \frac{N_{i,k}}{N_k} \quad P(X_j = x_j|\mathbf{X}_k = \mathbf{x}_k) \approx \frac{N_{j,k}}{N_k}.$$

The formula for calculating Chi-square is therefore given as follows:

$$\chi^2_{statistics}(X_i, X_j|\mathbf{X}_k) = \sum_{i=1}^{r_i}\sum_{j=1}^{r_j}\sum_{k=1}^{\mathbf{r}_k} N_k \frac{\left(\frac{N_{i,j,k}}{N_k} - \left(\frac{N_{i,k}}{N_k} \times \frac{N_{j,k}}{N_k}\right)\right)^2}{\frac{N_{i,k}}{N_k} \times \frac{N_{j,k}}{N_k}},$$

where r_i, r_j, $\mathbf{r}_k$ represent the domain sizes of X_i, X_j and $\mathbf{X}_k$ respectively. This is equivalent to:

$$\chi^2_{statistics}(X_i, X_j|\mathbf{X}_k) = \sum_{i=1}^{r_i}\sum_{j=1}^{r_j}\sum_{k=1}^{\mathbf{r}_k} \frac{\left(N_{i,j,k} - \left(\frac{N_{i,k}N_{j,k}}{N_k}\right)\right)^2}{\frac{N_{i,k}N_{j,k}}{N_k}} \tag{3.23}$$

Then, we can calculate the *p-value* which represents the probability under $\mathcal{H}_0$ of getting the observed value of $\chi^2_{statistic}$ or something even larger. This probability is a χ^2 distribution with $(r_i - 1) \times (r_j - 1) \times \mathbf{r}_k$ degrees of freedom. The null hypothesis $\mathcal{H}_0$ under which X_i is conditionally independent from X_j given $\mathbf{X}_k$ is considered to hold if and only if the *p-value* $\geq \alpha$, where α is generally fixed to 0.05. If $\mathcal{H}_0$ is rejected, then the edge between X_i and X_j remains in $\mathcal{G}$, otherwise it is removed by the algorithm.

Another statistical test has been proposed in the literature, which relies on the likelihood ratio G^2. It also follows a χ^2 distribution with $df = (r_i - 1) \times (r_j - 1) \times \mathbf{r}_k$ degrees of freedom:

$$G^2_{statistics}(X_i, X_j|\mathbf{X}_k) = 2\sum_{i=1}^{r_i}\sum_{j=1}^{r_j}\sum_{k=1}^{\mathbf{r}_k} N_{i,j,k}\log\left(\frac{N_{i,j,k}N_k}{N_{i,k}N_{j,k}}\right). \tag{3.24}$$

Sharing the same principle as the χ^2 test, the G^2 can be used either to accept or reject $\mathcal{H}_0$. As mentioned earlier, both χ^2 and G^2 require the definition of a significance level α to decide the rejection threshold of the null hypothesis. Remind that the α value must be chosen carefully since a high value leads to a dense skeleton (with many edges), else to a skeleton with too few edges to accurately represent the conditional independences of distribution P.

Algorithm 1: PC Algorithm

```
   Input: a dataset $\mathcal{D}$, significance level $\alpha$
   Output: a CPDAG $\mathcal{G}$
1  Start with a complete undirected graph $\mathcal{G}$
2  Let $d = 0$
3  //Skeleton learning step
4  repeat
5      while $\exists X - Y$ with $|\mathbf{Adj}(X) \setminus \{Y\}| \geq d$ do
6          repeat
7              Select a new set $\mathbf{Z} \subseteq \mathbf{Adj}(X)$, where $|\mathbf{Z}| = d$
8              if $\{X\} \perp_P \{Y\}|\mathbf{Z}$ then
9                  Delete edge $X - Y$ from $\mathcal{G}$
10                 $SepSet_{XY} = SepSet_{XY} \cup \mathbf{Z}$
11                 break
12         until All $\mathbf{Z}$ of size $d$ have been tested;
13     $d = d + 1$
14 until $\forall X \in \mathcal{X}$, $|\mathbf{Adj}(X)| \leq d$;
15 //Orientation step
16 foreach unshielded triple $\langle X, Z, Y\rangle$ do
17     if $Z \notin SepSet_{XY}$ then
18         $R1$: substitute in $\mathcal{G}$ edges $X - Z$ and $Z - Y$ by arcs $X \to Z$ and $Z \leftarrow Y$
19 repeat
20     $\forall\{X, Y\} \in \mathcal{X}^2$
21     if $\neg(X - Y)$, $\forall Z \in \mathcal{X} \setminus \{X, Y\}$, where $X \to Z$ and $Z - Y$ then
22         $R2$ : substitute in $\mathcal{G}$ edge $Z - Y$ by arc $Z \to Y$
23     if $X - Y$ and $X \to ... \to Y$ then
24         $R3$ : substitute in $\mathcal{G}$ edge $X - Y$ by arc $X \to Y$
25 until No further edge can be oriented;
26 return CPDAG $\mathcal{G}$
```

3.2.3.2 PC and IC algorithms

BN learning structure with the PC algorithm (Algorithm 1) [SGS01] is performed in two steps. The first one is called the *adjacency search*. It consists in learning the undirected skeleton. During the second step, edges are oriented through the identification of *v-structures* and the use of a set of deterministic rules [Mee95]. As defined in [Mee95], these rules rely on the learnt skeleton and on the conditioning separating sets discovered during the first phase of the algorithm. In the following, we provide further details on each of these two steps.

1. Adjacency search (lines 4-13 of Algorithm 1)

This step starts with a complete undirected graph, i.e., there is an edge between every pair of nodes. Edges are removed using statistical test $\{X\} \perp_P \{Y\} | \mathbf{Z}$ (using a statistical criterion like χ^2, G^2, etc.) where the size of the conditioning set $\mathbf{Z}$ is gradually increased until reaching a maximal fixed size called the *depth* of the search and denoted by d in the algorithm. To avoid the problem of checking all combinations of conditioning sets $\mathbf{Z}$ when testing the independence between any pair of variables X and Y, the PC algorithm considers only variables $\mathbf{Z}$ that are adjacent to either X or Y. The correctness of such restriction can be easily explained by the fact that if X and Y are *d-separated* by any subset of $\mathcal{X} \setminus \{X, Y\}$ in $\mathcal{G}$, then they are also *d-separated* either by $\mathbf{Pa}(X)$ or by $\mathbf{Pa}(Y)$ (and both of them are adjacent to either X or Y) [Pea88].

Let us now explain how the PC algorithm proceeds to learn the skeleton. First, it begins by checking the marginal independence between each pair of nodes X and Y (with a *depth* $d = 0$). If $\{X\} \perp_P \{Y\}$ holds, the corresponding edge is removed and $SepSet_{XY} = \varnothing$ is recorded. After checking all *marginal independences*, the *depth* of the search is increased by one, and the next step tries to check all conditional independences with a conditioning set whose size is equal to $d = 1$. It should be noted that this adjacency step requires that at least one of the tested nodes X or Y has a set of adjacent nodes whose size is greater than d. Actually, conditioning set $\mathbf{Z}$ is such that $\mathbf{Z} \subseteq \mathbf{Adj}(X) \setminus \{Y\}$ or $\mathbf{Z} \subseteq \mathbf{Adj}(Y) \setminus \{X\}$. The PC algorithm performs iteratively the independence tests with sets $\mathbf{Z}$, adjacent to each X or Y, of increasing sizes until all adjacency sets become smaller than or equal to d[3].

For a better understanding of the different steps while executing the PC algorithm, we consider the example of learning the BN depicted by Figure 3.2. The results of the learning steps of PC algorithm are shown in Figure 3.3. As discussed earlier, the algorithm starts with a completely connected undirected graph (a). At the first step, the algorithm removes edge $X - Y$ since both nodes are marginally independent (b). In the next step, the conditioning set composed by $\{Z\}$ leads to the deletion of edges $X - W$, $Y - V$, $Y - W$, $X - V$ and $W - V$ (d)(e)(f)(g). The first step of the PC algorithm ends with an undirected graph (skeleton) in which all adjacent nodes have a direct probabilistic dependence (g).

[3]We add "equal" to the inequality since when we test the conditional independence between any variable X with $Y \in \mathbf{Adj}(X)$, we consider as conditioning set variables in $\mathbf{Adj}(X) \setminus \{Y\}$.

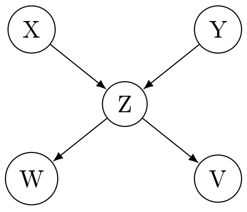


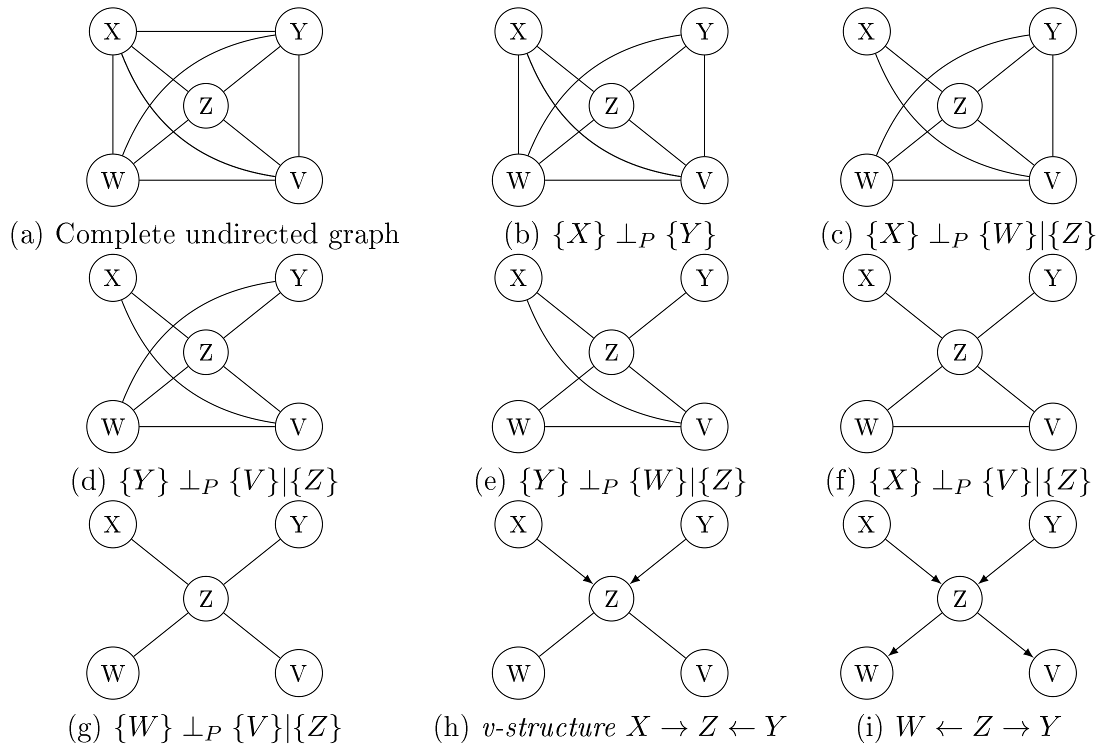
Figure 3.2: Structure we want to learn

2. Orientation step (lines 16-25 of Algorithm 1)

In this step, the different edges composing the skeleton (learnt during the previous step) are oriented by means of the following deterministic rules ($R1$, $R2$ and $R3$) [Mee95]:

- An unshielded triple $\langle X, Z, Y\rangle$ is a triple of nodes such that there exist edges $X - Z$ and $Y - Z$ in $\mathcal{G}$ but not edge $X - Y$.
  $R1$: for each unshielded triple $\langle X, Z, Y\rangle$ such that $Z \notin SepSet_{XY}$, orient $X - Y - Z$ as $X \rightarrow Z \leftarrow Y$. In other words, this orientation rule is based on the identification of possible *v-structures*, for which two nodes are independent, but become dependent conditionally on the third node Z. An example of *v-structure* identification can be seen in Figure 3.3.(h). In this example, edge $X - Y$ has been deleted at step (b) of the PC algorithm due to the marginal independence of these nodes. Since X and Y are adjacent to Z and $Z \notin SepSet_{XY}$, in that case, both X and Y are the causes of Z and thus the edges should be oriented toward the latter.
- $R2$: given $X \rightarrow Z$, if Z and W are adjacent, and X and W are not adjacent then convert $Z - W$ into $Z \rightarrow W$, since an opposite orientation would result in a *v-structure*. Figure 3.3.(i) gives an example of orientation using such a rule. In fact, the only possible orientation of $Z - W$ and $Z - V$ relations are $Z \rightarrow W$, $Z \rightarrow V$ since the opposite orientations would lead to *v-structures* $X \rightarrow Z \leftarrow W$ and $X \rightarrow Z \leftarrow V$ which are not confirmed by the independences shown at steps (c) and (f) of Figure 3.3. The resulting BN (after using $R1$ and $R2$) corresponds exactly to the original model shown at Figure 3.2.
- $R3$: if there is a directed path from X to Y, i.e., $X \rightarrow Z \rightarrow ... \rightarrow Y$, and an edge between X and Y, then orient $X - Y$ to $X \rightarrow Y$. Otherwise, this would introduce a directed cycle, which is forbidden in a DAG.

Note that the absence of sufficiently many *v-structures* in the first orientation step (with *R1*) can lead to the impossibility of orienting some edges since there is not enough information. In such a case, the orientations of some edges remain undecided. This results in a PDAG representing the *Markov equivalence class* of the final BN graph that is representative for the data in $\mathcal{D}$.



(a) Complete undirected graph (b) $\{X\} \perp_P \{Y\}$ (c) $\{X\} \perp_P \{W\}|\{Z\}$

(d) $\{Y\} \perp_P \{V\}|\{Z\}$ (e) $\{Y\} \perp_P \{W\}|\{Z\}$ (f) $\{X\} \perp_P \{V\}|\{Z\}$

(g) $\{W\} \perp_P \{V\}|\{Z\}$ (h) *v-structure* $X \to Z \leftarrow Y$ (i) $W \leftarrow Z \to Y$

FIGURE 3.3: An example of execution of the PC algorithm

Another constraint-based algorithm called *IC* (*Inductive Causation*) has been proposed in [PV91a]. Instead of starting from a complete undirected graph (as in the PC algorithm), the IC algorithm starts from an empty graph and tries to add edges between each pair of variables for which a conditional dependence given every set of variables $\mathbf{Z} \subseteq \mathcal{X} \setminus \{X, Y\}$ can be proven. Therefore, at the end of its execution, the IC algorithm obtains a minimal *D-map* whereas the PC algorithm looks for a minimal *I-map*. The orientation step of the IC algorithm is also based on the set of deterministic rules ($R1$, $R2$, and $R3$) presented earlier. It must be emphasized that the correctness of the PC and the IC algorithms relies on the correctness of conditional independences discovered during the learning process. However, as stated in [TBA06, SGS01, DD02], the reliability of constraint-based algorithms is very sensitive to the size of the dataset. In addition to the problem of the strong dependence between the resulting graph and the chosen significance level α, the power of the hypothesis test when learning the structure is also sensitive to the size of the conditioning set; as conditioning sizes grow, the reliability decreases [AGM06][4]. Hence, statistical tests may return results that erroneously suggest adding or removing edges from the skeleton, especially when the dataset size is small. Another problem concerning the classical constraint-based algorithms is related to the order in which the random variables are considered. This order, as we shall see, can lead to a wide range of different results, especially in high dimensional spaces (those containing many variables) [FHJ08].

[4]In the rest of this section, this problem will be referred to as an error of type II.

Several approaches have been proposed in the literature to address some problems of the classical constraint-based algorithms (PC, IC, *etc.*) [vDVDGT03b, TBA06, TASS03, AGM06, CGOM08, XG08, FHJ08, DD02, Mar03]. Some algorithms aim to enhance the reliability of constraint-based algorithms by reducing as much as possible the size of the conditioning sets used in the statistical tests. Most of the skeleton learning-based algorithms are variable ordering-dependent, i.e., the result they produce depends on the order in which they processed the random variables. So, some variants have also been proposed to overcome this problem in order to obtain a result that remains stable for all given orders [CM14]. Before providing the details about the most common extensions of constraint-based algorithms, we start this part by explaining how variable ordering may impact the skeleton learning process.

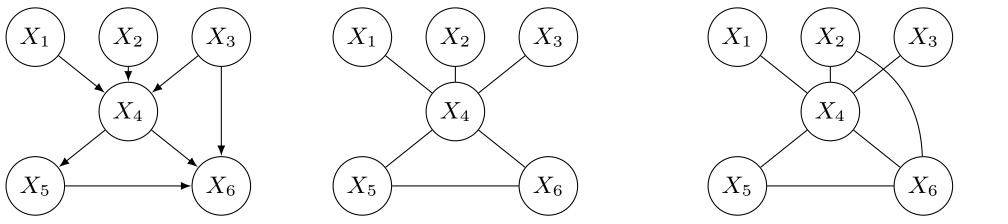
3.2.3.3 Problem of variable ordering dependence

In the following, we will consider a concrete example (inspired from [CM14]) to show the impact of variable ordering on the skeleton estimation process with the PC algorithm.

Example 3.2. *Suppose we have a distribution P defined on a set of variables $\mathcal{X} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$. Assume that distribution P is* faithful *to the DAG shown in Figure 3.4.(a). In this example, the set of independences encoded by the graph are: $\{X_1\} \perp_{\mathcal{G}} \{X_2\}$, $\{X_1\} \perp_{\mathcal{G}} \{X_3\}$, $\{X_3\} \perp_{\mathcal{G}} \{X_2\}$, $\{X_1\} \perp_{\mathcal{G}} \{X_5\}|\{X_4\}$, $\{X_2\} \perp_{\mathcal{G}} \{X_5\}|\{X_4\}$, $\{X_3\} \perp_{\mathcal{G}} \{X_5\}|\{X_4\}$, $\{X_1\} \perp_{\mathcal{G}} \{X_6\}|\{X_4, X_3\}$, $\{X_2\} \perp_{\mathcal{G}} \{X_6\}|\{X_4, X_3\}$. We assume that the only wrong independence statement returned by the algorithm (for a given α) is: $\{X_3\} \perp_P \{X_6\}|\{X_1, X_4\}$. Let us now explain how the first step of the PC algorithm determines the skeleton given two different orders: $\prec_1 = \{X_1 \prec X_3 \prec X_2 \prec X_5 \prec X_4 \prec X_6\}$ and $\prec_2 = \{X_1 \prec X_6 \prec X_3 \prec X_4 \prec X_5 \prec X_2\}$. The resulting skeletons from $\prec_1$ and $\prec_2$ are shown respectively in Figures 3.4.(b) 3.4.(c). To explain the resulting skeletons, we go through the first step of the PC algorithm for each given order to see what happened.*

Let us start with $\prec_1$. As defined in Algorithm 1, the starting point of PC is a complete undirected graph. When $d = 0$, the marginal independence tests lead to the following edge removals: $X_1 - X_2$, $X_1 - X_3$, $X_3 - X_2$. When $d = 1$, edges removed by conditional independence tests are: $X_1 - X_5$, $X_3 - X_5$ and $X_2 - X_5$. When $d = 2$, the following edges are removed: $X_1 - X_6$, $X_2 - X_6$ and also $X_3 - X_6$ since this latter is erroneously deleted due to the wrong decision $\{X_3\} \perp_P \{X_6\}|\{X_1, X_4\}$.

For $\prec_2$, the deletion process is performed as follows: when $d = 0$ and $d = 1$ the same deletions as those made for $\prec_1$ are performed. When $d = 2$, only $X_1 - X_6$ and $X_6 - X_3$ are deleted. Actually, edge $X_2 - X_6$ cannot be deleted given this order, since the neighbor set of X_6 has been affected by the previous deletions and therefore no longer contains X_3, which is necessary to delete this edge (see Line 5 of Algorithm 1). Therefore, independence test "($\{X_2\} \perp_P \{X_6\}|\{X_3, X_4\}$)?" is never performed and edge $X_2 - X_6$ cannot be deleted.



(a) True DAG (b) Skeleton learned for $\prec_1$ (c) Skeleton learned for $\prec_2$

FIGURE 3.4: The impact of variable ordering when learning the skeleton.

As can be seen in Example 3.2, at each search depth d, the prior $\prec$ determines the order in which pairs of adjacent nodes are considered when computing the set of conditional independences. For each edge removal, the skeleton $\mathcal{G}$ is updated, hence, the adjacency sets of the concerned nodes typically change within one level of d. Given a level d, when the adjacencies of some nodes are updated, the other conditional independences that have to be checked at level $d+1$ are also affected. Since the optimal order is not necessarily known *a priori*, variable order dependence becomes very problematic, especially for high-dimensional spaces since they can lead to highly unstable results, i.e., they can produce skeletons varying drastically from one order to the other.[5].

3.2.3.4 PC-stable

In our discussion, we focus here on the first phase of the PC-stable algorithm, which deals with the order dependence issue when learning the skeleton. As can be seen in Algorithm 2, the main difference between PC-stable and the classic PC algorithm is highlighted in lines 4 and 7 of Algorithm 2. At each level d, in a first stage, the algorithm saves the current neighbor set of all the variables (Line 4). Then it searches all the separating sets between variables, considering neighbors in the saved sets and, when determining conditional independences, it removes edges but does not update the saved sets. As such, the modifications in the graph are only taken into account when considering search depth $d+1$. This significantly reduces the instability problem resulting from variable ordering (as seen in Example 3.2). Actually, wrong decisions that may occur when using a statistical test at level d have no longer any influence on the other tests performed at the same level. In addition, PC-stable allows parallelizing conditional independence tests within each level d since they do not depend on each other anymore. It should be noted that the added instructions (shown by lines 4 and 7) in Algorithm 2 do not rule out the soundness and completeness of the PC-stable algorithm if the exact list of conditional independence relationships is known from the beginning [CM14].

Theorem 3.1. *Let P be a distribution* faithful *to a DAG $\mathcal{G}$, and assume that we have an exact list of all the conditional independence statements that hold between every pair*

[5]If a perfect conditional independence list is known *a priori*, the variable order-dependent algorithms such as PC or IC give the same result for all orders. But, unfortunately, in practice, such a list is not available.

of random variables, then the output of the PC-stable algorithm is exactly the CPDAG that represents $\mathcal{G}$.

If we apply PC-stable on Example 3.2, we obtain for both $\prec_1$ and $\prec_2$ the same skeleton given in Figure 3.4.(b). The result for $\prec_2$ can be explained as follows: despite the deletion of edge $X_3 - X_6$, PC-stable performs the deletion of the edge between $X_2 - X_6$ ($\{X_2\} \perp_P \{X_6\}|\{X_4, X_3\}$) since the neighborhood of X_6 remains unchanged when $d = 2$ (see Line 4 of Algorithm 2). For more details about the proof and the orientation step of the PC-stable algorithm, readers should refer to[CM14].

Algorithm 2: Learning the skeleton of the BN

```
Input: Data $\mathcal{D}$, significance level $\alpha$
Output: Skeleton $\mathcal{G}$
1  Start with a complete undirected graph $\mathcal{G}$
2  Let $d = 0$
3  repeat
4      $\forall X \in \mathcal{X}$, $adj_X = \mathbf{Adj}(X)$
5      repeat
6          Select a new set $\mathbf{Z} \subseteq adj_X$, where $|\mathbf{Z}| = d$
7          if $\{X\} \perp_P \{Y\}|\mathbf{Z}$ then
8              Delete edge $X - Y$
9              $SepSet_{XY} = SepSet_{XY} \cup \mathbf{Z}$
10             break
11     until All $\mathbf{Z}$ of size $d$ have been tested;
12     $d = d + 1$
13 until $\forall X \in \mathcal{X}$, $|\mathbf{Adj}(X)| \leq d$;
14 return undirected graph $\mathcal{G} = (\mathcal{X}, \mathbf{A})$
```

3.2.3.5 Variations on the PC Algorithm

Many extensions of the original PC algorithm have been advocated in [AGM06]. Among them, we can mention:

- the use of min cutset algorithms to minimize the sizes of the conditioning sets used in the conditional independence tests.
- the handling of the problem of link ambiguities when a triangle situation holds, i.e., when three variables X, Y and Z form a complete graph (3 links) and each pair of variables are dependent but are also conditionally independent given the third one.

Minimum *cut set*: as discussed earlier, when learning from small datasets, the use of conditional independence tests with a large conditioning set may lead to many testing

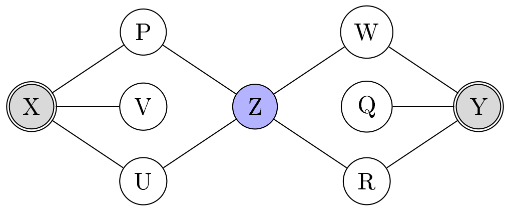


FIGURE 3.5: Minimum cutset between X and Y: $\mathbf{Cut}_{XY} = \{Z\}$.

errors. This can reduce the reliability of the skeleton recovery process. To alleviate this problem, authors in [ST99] proposed to reduce the size of set $\mathbf{Z}$ when testing the conditional independence between X and Y given $\mathbf{Z}$. This reduction method consists in restricting the set of conditioning variables to only those that appear on a path linking X and Y in $\mathcal{G}$. Another variant has been proposed in [AGM06], which aims to find a minimum cutset $\mathbf{Cut}_{XY}$, i.e., a minimal set of nodes that, when removed from $\mathcal{G}$, makes X and Y belong to two different connected components. This can be easily done by computing a min cutset in a max-flow problem [AC96]. At each iteration of the PC algorithm, if the size of $\mathbf{Cut}_{XY}$ is smaller than $\mathbf{Z}$ (incremented after each iteration), we just perform $\chi^2(\{X\} \perp_P \{Y\}|\mathbf{Cut}_{XY})$. This results in the construction of smaller contingency tables and, as such, this increases the accuracy of the statistical tests. Moreover, the time overhead required to compute $\mathbf{Cut}_{XY}$ can be balanced by the number of χ^2 tests it allows to avoid. Figure 3.5 depicts an example of computation of a min cutset when testing the conditional independence between X and Y. The first step consists in removing the link between X and Y and then in determining the smallest set of variables that blocks all the undirected paths between these two nodes. In this example, we can clearly see that the min cutset is $\mathbf{Cut}_{XY} = \{Z\}$. Instead of conditioning on a set of three variables $\mathbf{Z} = \{V, U, P\}$ or $\mathbf{Z} = \{W, Q, R\}$, which increases the possibility of having a statistical error, we consider only one variable in the conditioning sets. By using the min cutset option, the following instruction should replace instruction 7 of Algorithm 1: *"while $\exists\ X - Y$, find $\mathbf{Z} = \boldsymbol{Cut}_{XY}$ in $\mathcal{G}$".*

Triangle resolution: Figure 3.6 depicts an example of a fully connected undirected graph composed by 3 variables: X, Y and Z. In this example, in distribution P, each pair of variables are marginally dependent but they are also conditionally independent given the third variable [AGM06]: $\{X\} \not\perp_P \{Y\}$, $\{Y\} \not\perp_P \{Z\}$, $\{X\} \not\perp_P \{Z\}$, $\{X\} \perp_P \{Y\}|\{Z\}$, $\{Y\} \perp_P \{Z\}|\{X\}$ and $\{X\} \perp_P \{Z\}|\{Y\}$.

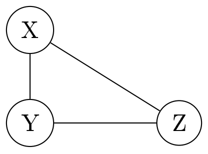


FIGURE 3.6: An example of triangle structure.

Hence, when executing the classical PC learning algorithm, the order in which variables are considered leads to the skeletons displayed in Figure 3.7.

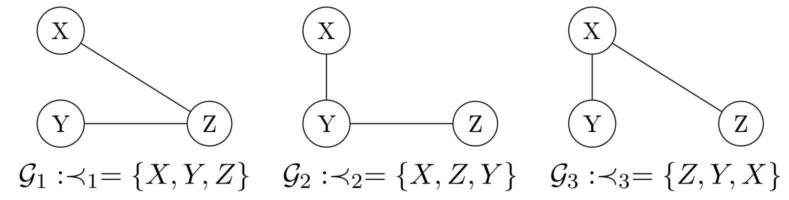

X
Y
Z
$\mathcal{G}_1 : \prec_1 = \{X, Y, Z\}$
X
Y
Z
$\mathcal{G}_2 : \prec_2 = \{X, Z, Y\}$
X
Y
Z
$\mathcal{G}_3 : \prec_3 = \{Z, Y, X\}$

FIGURE 3.7: The set of resulting skeletons given the orders $\prec_1$, $\prec_2$ and $\prec_3$.

This ambiguous situation has been tackled by Abellan et al. [AGM06] through the deletion of the link that corresponds to the weakest dependence among the three possible edge deletions. As mentioned in [AGM06], the dependence strength of each link is calculated by means of a statistical test such as χ^2, G^2, *etc.*

3.2.3.6 MMPC approach

Max-Min Parents Children (MMPC) [TAS03] is an order-dependent heuristics that determines the skeleton over a high-dimensional space. Given a target variable denoted by T and a dataset $\mathcal{D}$, MMPC allows to perform a local discovery of the set of parents and children of T denoted by $\mathbf{PC}_{\mathcal{G}}^{T}$[6]. Unlike the PC algorithm, MMPC operates in a more depth-first search manner, considering all the possible parents-children candidates for a given node T before moving to the next one. The set of all these candidates is denoted by **CPC** (where $\mathbf{CPC} \subseteq \mathcal{X} \setminus \{T\}$). It should be noted that, for each node T, set $\mathbf{PC}_{\mathcal{G}}^{T}$ is unique for all the BNs that are *faithful* to the same distribution [PV+91b, BHKL91].

As a constraint based algorithm, MMPC relies on the use of G^2 statistical tests to decide whether pairs of variables T and X are conditionally independent given a set $\mathbf{S} \subseteq \mathbf{CPC}$. In this context, conditional independence assertions are used to quantify the strength of the association between T and each $X \in \mathcal{X} \setminus (\mathbf{CPC} \cup \{T\})$ given $\mathbf{S}$, denoted by $Assoc(T, X - \mathbf{S})$[7], i.e., if $\{T\} \perp_P \{X\} | \mathbf{S} \iff Assoc(T, X - \mathbf{S}) = 0$. Function $Assoc$ uses the *p-value* returned by G^2 test: the smaller the *p-value*, the higher the association between the tested variables. To check whether variable X can be added to the set of **CPC** of T, the following test has to be performed:

$$\mathbf{CPC} \leftarrow \begin{cases} \mathbf{CPC} & \text{if } \exists \mathbf{S} \subseteq \mathbf{CPC}, s.t. \{X\} \perp_P \{T\} | \mathbf{S} \\ \mathbf{CPC} \cup \{X\} & otherwise \end{cases} \tag{3.25}$$

In other words, a node X can be added to the set of **CPC** of a target T if and only if the minimum association (see Equation (3.26)) between X and T given a set of conditioning

[6]This notation shall not be confused with the PC algorithm.
[7]$S \subseteq \mathbf{CPC}$ as defined earlier.

variables $\mathbf{S} \subseteq \mathbf{CPC}$ does not entail an independence between them.

$$MinAssoc(T, X|\mathbf{CPC}) = \min_{\mathbf{S} \subseteq \mathbf{CPC}} Assoc(T, X|\mathbf{S}). \tag{3.26}$$

As shown in Algorithm 4, MMPC is composed of two phases. In the first phase, called the "forward" phase, the algorithm starts with an empty set of **CPC** for the target variable T. Then, at each iteration, a variable X is added to the list of **CPC** following the test shown in Equation (3.25). The first phase stops when the minimum association of the remaining variables reaches zero (lines 2-7). The second phase, called the "backward" phase, consists in removing from the **CPC** the false positive parents-children that may have been detected in the previous step. By false positive, we mean each variable $X \in \mathbf{CPC}$ that is independent from T given a subset $\mathbf{S} \subseteq (\mathbf{CPC} \setminus \{X\})$ (lines 9–13). Note that if there exists a DAG $\mathcal{G}$ which is *faithful* to P, the MMPC algorithm guarantees the learning of a correct skeleton. As explained in [TBA06], the $\mathbf{PC}_{\mathcal{G}}^{T}$ returned for each target variable T are used afterward to build the skeleton of the BN as follows: X and Y are connected by an edge in the skeleton $\mathcal{G}$ if and only if $Y \in \mathbf{PC}_{\mathcal{G}}^{X}$ AND $X \in \mathbf{PC}_{\mathcal{G}}^{Y}$ as shown in Algorithm 4.

To complete the orientation of the learned skeleton using the information provided by set $\{\mathbf{PC}_{\mathcal{G}}^{X_i}\}_{i=1}^{n}$, the algorithm performs a *greedy search* over the possible DAGs to find an optimal orientation.

Algorithm 3: $\overline{MMPC}$

```
Input: Target variable $T$, dataset $\mathcal{D}$
Output: list of CPC
1  //Forward step
2  CPC ← $\varnothing$
3  repeat
4      if $\exists X,\ s.t.\ \forall \mathbf{S} \subseteq \mathbf{CPC}, \{X\} \not\perp_P \{T\}|\mathbf{S}$ then
5          CPC ← $\mathbf{CPC} \cup \{X\}$
6      end
7  until CPC does not change;
8  //Backward
9  foreach $X \in \mathbf{CPC}$ do
10     if $\exists \mathbf{S} \subseteq \mathbf{CPC},\ s.t.\ \{X\} \perp_P \{T\}|\mathbf{S}$ then
11         CPC ← $\mathbf{CPC} \setminus \{X\}$
12     end
13 end
14 return list of CPC for $T$
```

Algorithm 4: MMPC Algorithm

```
Input: Target variable $T$, dataset $\mathcal{D}$
Output: list of CPC
1 CPC ← $\overline{MMPC}(T, \mathcal{D})$
2 foreach $X \in$ CPC do
3     if $T \notin \overline{MMPC}(X, \mathcal{D})$ then
4         CPC ← CPC $\setminus \{X\}$
5     end
6 end
7 return list of final CPC
```

3.2.3.7 Fast-IAMB approach

Fast-IAMB [YM05] is a constraint-based algorithm for BN skeleton learning relying on the identification of *Markov blankets*. Instead of learning for each target variable T its $\mathbf{PC}_{\mathcal{G}}^{T}$ (as the MMPC algorithm does), the Fast-IAMB algorithm aims to identify only the Markov blanket of node T, i.e., the set $\mathbf{MB}(T)$ such that T is guaranteed to be unaffected by any instantiation of the sets of nodes $\mathbf{Z} \subseteq \mathcal{X} \setminus (\mathbf{MB}(T) \cup \{T\})$. The Markov blanket is the optimal set of nodes that allows to predict the values taken by T (as explained in [KS96]). Before giving the details of the Fast-IAMB algorithm, it should be noted that this algorithm relies on the existence of a *faithful* BN for the underlying probability distribution P. This assumption implies that only one *Markov blanket* exists for each target variable T in the network. The Fast-IAMB relies on the use of G^2 statistical tests to identify the set of conditional independence assertions among the variables. Similarly to the MMPC algorithm, the Fast-IAMB algorithm performs the skeleton learning in two phases: the *growing* and *shrinking* steps. The *growing* step starts by computing, using G^2 tests, an order over the set of candidate variables $\mathbf{S}$ of $\mathbf{MB}(T)$ (sorted by decreasing dependence w.r.t. T). For a given target variable T, the set of candidate variables $\mathbf{S}$ are obtained as follows:

$$\mathbf{S} \leftarrow \{X \in \mathcal{X} \setminus \{T\} \ s.t. \ \{X\} \not\perp_P \{T\}\}. \tag{3.27}$$

The purpose of making such an ordering is to enhance the reliability of the algorithm by reducing the number of false positive nodes that may be added to $\mathbf{MB}(T)$. As shown in Algorithm 5, before adding any variable to the list $\mathbf{MB}(T)$, Fast-IAMB performs an additional instruction in order to choose only variables (lines 8) that guarantee the reliability of the statistical independence test $\{X\} \perp_P \{T\}|\mathbf{MB}(T)$, i.e., those that satisfy:

$$\frac{N}{r_T \cdot r_X \cdot r_{\mathbf{MB}(T)}} \geq k \tag{3.28}$$

where the value of k is generally fixed to 5 and r_T, r_X, $r_{\mathbf{MB}(T)}$ denote respectively the domain sizes of variables T, X and $\mathbf{MB}(T)$. In other words, the aim of Equation (3.28) is to ensure that the amount of data in each cell of the contingency tables of the statistical tests is sufficient to perform the conditional independence test $\{X\} \perp_P \{T\}|\mathbf{MB}(T)$[8]. The second phase of the algorithm (called *"shrinking"*) attempts to remove false positive (or irrelevant) variables that might be added to $\mathbf{MB}(T)$ during the *"growing"* step. It corresponds to computing the following set:

$$\mathbf{MB}(T) \leftarrow \begin{cases} \mathbf{MB}(T) \setminus \{X\} & \text{if } \exists \mathbf{S} \subseteq \mathbf{MB}(T), s.t. \{X\} \perp_P \{T\}|\mathbf{S} \\ \mathbf{MB}(T) & otherwise \end{cases} \tag{3.29}$$

The pseudocode of Fast-IAMB is given by Algorithm 5.

Algorithm 5: Fast-IAMB Algorithm

```
Input: Target variable $T$, data $\mathcal{D}$, threshold $k$
Output: $\mathbf{MB}(T)$
1  $\mathbf{S} \leftarrow \{X \in \mathcal{X}, s.t. \{X\} \not\perp_P \{T\}\}$
2  //Growing step
3  $\mathbf{MB}(T) \leftarrow \varnothing$
4  repeat
5      $\mathbf{S} \leftarrow sort(\mathbf{S})$ according to their dependences w.r.t. $T$
6      sufficient$_\mathcal{D} \leftarrow True$
7      foreach $X \in \mathbf{S}$ do
8          if $\frac{N}{r_T \cdot r_X \cdot r_{\mathbf{MB}(T)}} \geq k$ then
9              $\mathbf{MB}(T) \leftarrow \mathbf{MB}(T) \cup \{X\}$
10         else
11             sufficient$_\mathcal{D} \leftarrow False$
12             go to line 15
13         end
14     end
15     //Shrinking phase
16     $remove \leftarrow False$
17     foreach $X \in \mathbf{MB}(T)$ do
18         if $\{X\} \perp_P \{T\}|(\mathbf{MB}(T) \setminus \{X\})$ then
19             $\mathbf{MB}(T) \leftarrow (\mathbf{MB}(T) \setminus \{X\})$
20             $remove \leftarrow True$
21         end
22     end
23     if $remove = False$ AND $sufficient_\mathcal{D} = False$ then
24         break
25     else
26         $\mathbf{S} \leftarrow \{X | X \in \mathcal{X} \setminus \{T, \mathbf{MB}(T)\}$ AND $\{X\} \not\perp_P \{T\}|\mathbf{MB}(T)\}$
27     end
28 until $\mathbf{S} \neq \varnothing$;
29 return $\mathbf{MB}(T)$
```

Like the MMPC approach, the Fast-IAMB algorithm is completed by an edge orientation step relying, for instance, on the use of a *greedy search* optimization approach.

[8]This instruction allows to alleviate the problem of errors of type II (false negative edges) since the lack of data leads automatically to incorrect independences.

3.2.4 Score-based approaches

Score-based algorithms perform an optimization search through the set of possible DAGs in order to find the best structure according to an evaluation criterion called the "scoring function". These algorithms assign to each DAG a score which represents a trade-off between the precision of the model and its complexity [Hec95, Aka70, S+78]. As discussed earlier, when the number of variables is large, an exhaustive consideration of all the possible structures existing in the search space is computationally unfeasible. To make this task tractable, the only solution consists in using an approximation algorithm. Known in the literature as the heuristic search, this class of approximation algorithms tries to find a solution which may not be optimal but which, hopefully, may not be too far away from the optimum.

It should be emphasized that heuristic algorithms require the definition of a *search space* (from which the DAGs will be examined), the definition of a neighborhood (the algorithm iterating moves from one graph to one of its neighbors) and a scoring function to evaluate the goodness of fit of each DAG candidate as a representation of the input data $\mathcal{D}$.

3.2.4.1 Possible search spaces

DAG-based search space For BN structures, the natural search space is the space of all the DAGs defined over n nodes/variables. Given the DAGs space, learning algorithms try to explore the possible structures in order to find the one that has the highest value of the scoring function. Given a current structure $\mathcal{G}$, the set of operators that must be used to move throughout the DAGs in the neighborhood space are:

- If two nodes X and Y are not adjacent in $\mathcal{G}$, add an arc between them in either direction.
- If arc $X \to Y$ exists in $\mathcal{G}$, remove it.
- If arc $X \to Y$ exists in $\mathcal{G}$, reverse it.

Note that the above operators are sufficient to navigate throughout the search space, i.e., from any DAG $\mathcal{G}_i$, we can reach any other DAG $\mathcal{G}_j$ in the search space applying a sequence of the above operations. It should be noted that all transformations obtained are subject to the constraint that the resulting graph $\mathcal{G}$ should be acyclic (it cannot contain any directed cycle). In theory, arc reversal operators are not absolutely necessary since they are equivalent to a sequence of an arc deletion followed by another arc addition. In practice, this operator is needed to avoid being stuck in local optima: it may actually be the case that removing arc $X \to Y$ decreases the score of the graph whereas reversing it may increase the latter. If arc reversal is considered as a sequence of two operations (arc deletion + arc addition), some search algorithms may consider not performing it since the first operation (deletion) would decrease the score of the graph.

***Markov Equivalence classes* search space:** Instead of trying to determine the best structure by searching over the DAGs space, Chickering [Chi95a, Chi02b] has suggested searching over the space of *Markov equivalence classes*. In this context, the algorithm iterate moves from one CPDAG to another in its neighborhood. Once an optimal CPDAG is determined, it is converted into a DAG to obtain the optimal structure $\mathcal{G}^*$. As discussed in [Chi95b], searching over *Markov equivalence classes* allows to alleviate many problems encountered when searching over DAGs.

The first problem is related to the convergence speed of the search process. Since several well-known scoring functions[9] (e.g., AIC, BDe, BIC) used for BN structure learning assign the same score to *equivalent structures*, searching over the DAGs space can waste significant time moving from one DAG to another in the same Markov equivalence class, hence performing moves that do not produce better DAGs. This is especially true for large networks. Figure 3.9 depicts an example of useless move: moving from $\mathcal{G}_1$ to $\mathcal{G}_2$ is not useful since both graphs belong to the same Markov equivalence class; the same holds for $\mathcal{G}_3$ and $\mathcal{G}_4$. As shown in [GP01], the space of DAGs is of 3.7 times larger than the space of *Markov equivalence classes* when the number of nodes n is equal to 10.

The second problem concerns the prior assigned implicitly to the DAGs. By assigning equal priors to all DAGs when searching over the DAGs space, it turns out that the *Markov equivalence class* with the largest number of DAGs receives the highest prior probability. For example, complete CPDAGs, which do not represent any conditional independence, contain $n!$ DAGs (the highest prior), whereas CPDAGs without any edge contain only one DAG (the lowest prior).

The third problem concerns the efficiency of the search algorithm. By using equivalent scores, search algorithms usually get stuck in local maxima.

Chickering *et al.* [Chi02a] define a set of six operators to perform local transformations when navigating through the space of *Markov equivalence classes*. Those are: **InsertU, DeleteU, InsertD, DeleteD, ReverseD and MakeV**. The first two operators respectively increase and reduce the number of undirected edges in the CPDAG. The third and fourth ones respectively increase and decrease the number of directed edges. The fifth operator maintains the same number of arcs and edges in the current CPDAG since it only reverses the direction of an arc. The last one transforms unshielded triples $X - Z - Y$ into *v-structures* $X \rightarrow Z \leftarrow Y$. After each modification of the CPDAG, an operation called PDAG-to-DAG is performed in order to guarantee the validity of the modification,

[9]Definitions about scoring functions will be provided in section 3.2.4.2.

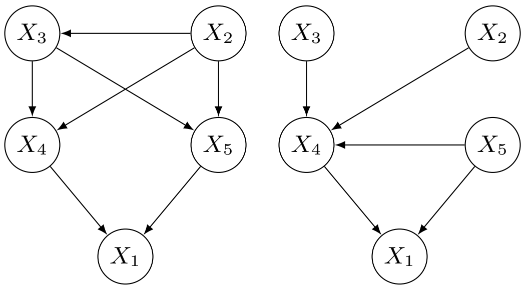


FIGURE 3.8: Structures consistent with $(X_2 \prec X_3 \prec X_5 \prec X_4 \prec X_1)$.

i.e., if the applied operator fails to return a valid DAG structure, then the modification cannot be carried out.

Ordering-based search space: Unlike the majority of algorithms that perform a search over the DAGs space, the order-based approach [TK05] intends to find the best structure by searching over a (topological) ordering space $\mathcal{O}$ over the random variables. For each order "$\prec$", it determines the best structure consistent with this order. Given an order $\prec$, if we bound the indegree of each node X_i to k parents, the set of possible parents for each node in the network is then restricted. More formally, the possible set of parent nodes of X_i for a given $\prec$ and indegree k is:

$$\mathbf{Pa}(X_i)_{\prec} = \{\mathbf{X} \subset \mathcal{X} : |\mathbf{X}| \leq k \text{ and } X_j \prec X_i \text{ for all } X_j \in \mathbf{X}\}. \tag{3.30}$$

Figure 3.8 depicts an example of two structures compatible with the following order $\prec = (X_2 \prec X_3 \prec X_5 \prec X_4 \prec X_1)$. By exploiting the *decomposability* property of the scoring function[10], the optimal structure of the BN, which is denoted by $\mathcal{G}^*_{\prec}$, is learnt by constructing for each node X_i the optimal family $\mathbf{Pa}(X_i)$ consistent with $\prec$:

$$\forall X_i \in \mathcal{X}, \mathbf{Pa}(X_i) = \underset{\mathbf{Pa}(X_i) \subseteq \mathbf{Pa}(X_i)_{\prec}}{\text{Argmax}} \; score(X_i|\mathbf{Pa}(X_i)). \tag{3.31}$$

Structure learning in then performed by conducting a search over ordering space $\mathcal{O}$ by assigning to each order the score of the best network $\mathcal{G}^*_{\prec}$ consistent with it. Therefore, the optimal structure is then obtained from the optimal order $\prec^*$, which is determined as follows:

$$\prec^* = \underset{\prec}{\text{Argmax}} \, score(\mathcal{G}^*_{\prec}|D). \tag{3.32}$$

As shown in [TK05], searching over $\prec$ has many attractive properties. First, the search space is significantly smaller ($2^{O(n \log n)}$) than the DAGs space. Second, for a given order, searching the optimal structure can be addressed effectively since the set of possible parents is restricted by $\prec$. Therefore, it is easier to avoid being stuck in local maxima. Finally, the acyclicity test of each evaluated structure (which can be a costly operation,

[10]Properties of scoring functions will be given in section 3.2.4.2.

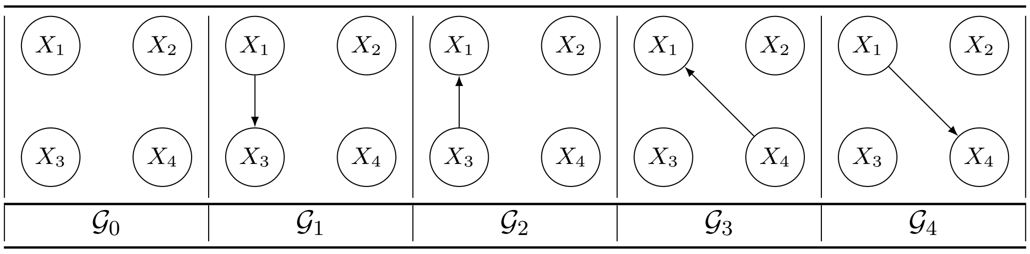


FIGURE 3.9: The four neighborhood candidates of graph $\mathcal{G}_0$

especially for large networks) is not an issue given $\prec$, since a variable X_i cannot be both a predecessor and a successor for a variable $X_j \in \mathcal{X} \setminus \{X_i\}$ in a given order $\prec$. Hence no directed cycle can exist.

Navigating over ordering space $\mathcal{O}$ can be performed throughout several sets of operators: among them we can cite the simplest swap operator which consists in modifying the position of two variables in order $\prec$:

$$(X_i \prec X_{i+1} \prec ... \prec X_j \prec X_{j+1}, ...) \to (X_i \prec X_{i+1} \prec ... \prec X_{j+1} \prec X_j, ...). \tag{3.33}$$

Similarly to the DAG-search space, for each order $\prec$, all $n-1$ swaps are generated, the best network for each swap is computed and the order with the highest score is considered as the new current order $\prec$. The searching process continues until a local maximum is reached.

3.2.4.2 The scoring functions

Throughout the computation of $P(\mathcal{G}|\mathcal{D})$ a scoring function is used to evaluate how well structure $\mathcal{G}$ matches the data $\mathcal{D}$. More precisely, by assuming a uniform prior on all structure $P(\mathcal{G})$, we have that:

$$P(\mathcal{G}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})} \propto P(\mathcal{D}|\mathcal{G}) = \int_{\mathbf{\Theta}} P(\mathcal{D}|\mathcal{G}, \mathbf{\Theta})\pi(\mathbf{\Theta}|\mathcal{G})d\mathbf{\Theta}. \tag{3.34}$$

In Equation (3.34), $P(\mathcal{D}|\mathcal{G})$ denotes the *likelihood* of data $\mathcal{D}$ given $\mathcal{G}$, π represents the prior on the parameters $\mathbf{\Theta}$ of a discrete BN with structure $\mathcal{G}$, and $P(\mathcal{D})$ represents the prior on the data $\mathcal{D}$. By assuming a uniform prior on $\mathcal{G}$, it turns out that searching for the optimal structure maximizing the posterior probability $P(\mathcal{G}|\mathcal{D})$ is equivalent to finding the one that maximizes the likelihood $P(\mathcal{D}|\mathcal{G})$. Several scoring functions for learning Bayesian networks exist in the literature. Essentially, they result from different hypotheses made on prior π and on $\mathbf{\Theta}$. Before giving the details related to the most common used scores, we start by talking about two of their main properties: *decomposition* and *Markov equivalence*. As we shall see in the next section, search methods heavily rely on them when learning the BN structures.

Definition 3.1. ***(Score decomposition)*** *A scoring function $score(\mathcal{G}|\mathcal{D})$ is said to be decomposable if and only if it can be expressed as the sum over each node in $\mathcal{G}$ of a local score depending only on the node and its parents:*

$$score(\mathcal{G}|\mathcal{D}) = \sum_{i=1}^{n} score(X_i|\mathbf{Pa}(X_i), \mathcal{D}). \tag{3.35}$$

By abuse of notation, since the local scores obviously depend unambiguously on dataset $\mathcal{D}$, they are often denoted as $score(X_i|\mathbf{Pa}(X_i))$. Exploiting the decomposition property, after each modification of structure $\mathcal{G}$ (by adding, deleting or reversing an arc), instead of recomputing the overall score of the corresponding structure, we only need to compute the local score of the family concerned by the modification and sum it to the other precalculated (or unchanged) local scores. All the scoring functions that we will mention in the following are decomposable.

Definition 3.2. ***(Score equivalence)*** *A scoring function is said to be score equivalent if and only if it assigns the same score to all the networks that belong to the same Markov equivalence class.*

Now, let us delve into the details of the different scoring functions. These scores are classified into two main categories: Bayesian and information-theoretic scores. We start by discussing the Bayesian scores. They consist in computing the posterior probability distribution $P(\mathcal{G}|\mathcal{D})$, starting with a prior distribution $P(\mathcal{G})$ over the possible networks. In the following, we provide details about the most common Bayesian scores in the literature such as *Bayesian Dirichlet* [CH92b] (BD), *Bayesian Dirichlet Equivalence* [HGC95] (BDe) and *Bayesian Dirichlet equivalence uniform* [Bun91](BDeu).

***Bayesian Dirichlet* (BD):** Proposed by [CH92b], the BD score relies on a Bayesian perspective, i.e., the uncertainties over structures, parameters and possible databases are probabilized. The Bayesian scoring function is derived as follows:

$$\begin{aligned} Score_{BD}(\mathcal{G}, \mathcal{D}) &= P(\mathcal{G}|\mathcal{D}) \propto P(\mathcal{D}, \mathcal{G}) \\ &= \int_{\mathbf{\Theta}} P(\mathcal{D}, \mathcal{G}, \mathbf{\Theta}) d\mathbf{\Theta} \\ &= \int_{\mathbf{\Theta}} P(\mathcal{D}|\mathcal{G}, \mathbf{\Theta}) \pi(\mathbf{\Theta}|\mathcal{G}) \pi(\mathcal{G}) d\mathbf{\Theta} \\ &= \pi(\mathcal{G}) \int_{\mathbf{\Theta}} P(\mathcal{D}|\mathcal{G}, \mathbf{\Theta}) \pi(\mathbf{\Theta}|\mathcal{G}) d\mathbf{\Theta} \end{aligned}$$

Assuming that $\pi(\mathbf{\Theta}|\mathcal{G})$ is a Dirichlet prior with hyperparameters $\alpha_{i,j,k}$, Chickering and Heckerman [CH96] have proved that the above equation is equivalent to the following one:

$$Score_{BD}(\mathcal{G}|\mathcal{D}) = \pi(\mathcal{G}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{i,j})}{\Gamma(N_{i,j} + \alpha_{i,j})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{i,j,k} + \alpha_{i,j,k})}{\Gamma(\alpha_{i,j,k})} \tag{3.36}$$

where $\alpha_{i,j} = \sum_{k=1}^{r_i} \alpha_{i,j,k}$ and $\Gamma(\cdot)$ is the usual Gamma function.

***Bayesian Dirichlet Equivalent* (BDe):** Unfortunately, it can be shown that the BD score is not score-equivalent in general. Heckerman *et al.* [HGC95] addressed this problem by proposing a variant called the *Bayesian Dirichlet Equivalent* (BDe) score. They proved that this new score is both decomposable and score-equivalent. The only difference between the BD and BDe scores is the way by which each $\alpha_{i,j,k}$ value is specified. By introducing equivalent sample sizes $N^{'}$, the BDe score defines:

$$\alpha_{i,j,k} = N^{'} P(X_i = x_k, \mathbf{Pa}(X_i) = \mathbf{x}_j | \mathcal{G}) \tag{3.37}$$

where $N^{'}$ represents to the strength of our belief assigned to the prior distribution.

***Bayesian Dirichlet equivalence uniform* (BDeu):** The major problem of BDe is the computation of the probabilities $P(X_i = x_k, \mathbf{Pa}(X_i) = \mathbf{x}_j | \mathcal{G})$ which do not depend on any particular value of parameter set $\boldsymbol{\Theta}$. This makes these probabilities hard to estimate in practice. To overcome this problem, in [Bun91], a new score called BDeu has been proposed that, under some additional assumptions, specifies the values of the probabilities:

$$\alpha_{i,j,k} = N^{'} P(X_i = x_k, \mathbf{Pa}(X_i) = \mathbf{x}_j | \mathcal{G}) = \frac{N^{'}}{r_i q_i} \tag{3.38}$$

where r_i and q_i are the domain sizes of X_i and $\mathbf{Pa}(X_i)$ respectively. Note that the BDeu score is very sensitive to the sample size $N^{'}$ (which is the only needed parameter): large values tend to *oversmooth* the data $\mathcal{D}$ and thus lead to complex DAGs with many edges. The resulting BDeu score (obtained by replacing $\alpha_{i,j,k}$ of Equation (3.36) by its value shown in Equation (3.38)) is decomposable, as well as score equivalent.

The K2 score: Like BDeu, the K2 score [CH92a] can be thought of as the combination of a Bayesian Dirichlet (BD) score with a particular *a priori*. In the case of K2, this *a priori* is a Dirichlet distribution with all hyperparameters $\alpha_{i,j,k}$ equal to 1. As it is well-known that $\Gamma(1) = 1$ and that, for any positive integer value n, we have that $\Gamma(n+1) = n!$, Equation (3.36) can be rewritten as:

$$Score_{K2}(\mathcal{G}|\mathcal{D}) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{i,j} + r_i - 1)!} \prod_{k=1}^{r_i} N_{i,j,k}!$$

The second category of scoring functions is information-theoretic. The idea is to exploit information-theoretic notions to find a model that corresponds as much as possible to the observed dataset $\mathcal{D}$. Of course, the complete graph $\mathcal{G}$ certainly accounts for all the information stored in $\mathcal{D}$. So, information-theoretic approaches also exploit the *Occam's Razor* principle to keep the graph structure as sparse as possible. As such, these scores are thus divided into two parts: the likelihood $L(\mathcal{D}|\mathcal{G})$, which gives us an idea about how well model $\mathcal{G}$ fits to the data $\mathcal{D}$, and a second term which penalizes complex models. The complexity is usually expressed in terms of the number of parameters $\theta_{i,j,k}$ needed to represent the BN. Given a node X_i and its parents $\mathbf{Pa}(X_i)$ in $\mathcal{G}$ with respective domain sizes r_i and q_i, the number of parameters needed to represent distribution $P(X_i|\mathbf{Pa}(X_i))$, which is denoted by $dim(X_i, \mathcal{G})$, is obtained as follows:

$$dim(X_i, \mathcal{G}) = (r_i - 1)q_i. \tag{3.39}$$

Therefore, the overall number of parameters of a given BN is:

$$dim(\mathcal{G}) = \sum_{i=1}^{n}(r_i - 1)q_i. \tag{3.40}$$

There exist several information-theoretic scoring functions. As for the Bayesian scores, their differences result from different hypotheses made. Among these scores, we can mention: the *likelihood* score , the *Bayesian information criterion (BIC)* and the *Akaike information criterion (AIC)*.

Likelihood score: As we have seen before, the likelihood of data $\mathcal{D}$ given a network $\mathcal{G}$ is computed by the following equation:

$$L(\mathcal{D}|\mathcal{G}) = \prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}(\theta_{i,j,k})^{N_{i,j,k}}. \tag{3.41}$$

By taking the log of the previous equation, this lead to the $LL(\mathcal{D}|\mathcal{G})$ scoring function, defined as follows:

$$LL(\mathcal{D}|\mathcal{G}) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{i,j,k}\log\theta_{i,j,k} \tag{3.42}$$

As shown in Equation (3.42), the LL score does not take into account the complexity of the model, therefore it tends to overfit the data by favoring the learning of graphs $\mathcal{G}$ with (too) many arcs.

AIC/BIC scores: The Akaike Information Criterion and the Bayesian Information Criterion [Aka70, S+78] (respectively denoted by AIC and BIC) scoring functions intend to alleviate the problem of overfitting by favoring simple networks over complex ones. To do so, they add to the likelihood function some penalization term. Equations for the

AIC and BIC scores are given as follows:

$$score_{AIC}(\mathcal{G}|\mathcal{D}) = LL(D|\mathcal{G}) - dim(\mathcal{G}). \tag{3.43}$$

$$score_{BIC}(\mathcal{G}|\mathcal{D}) = LL(D|\mathcal{G}) - \frac{1}{2}dim(\mathcal{G})logN. \tag{3.44}$$

Independently, the same formula as BIC was derived using a Minimum Description Length (MDL) justification [LB94] and is therefore called the MDL score.

3.2.4.3 Heuristics search approaches

***Greedy search*:** This kind of algorithm can exploit any of the above scores. For this algorithm, the considered search space is composed by all the DAGs defined over n variables, and the set of operators used to perform the search are those mentioned in Paragraph 3.2.4.1. Basically, the principle of this algorithm is the following: it starts with a structure $\mathcal{G}_0$ (often empty). Then, at each step, it looks in the neighborhood of the current structure (see Figure 3.9) for another structure $\mathcal{G}$ with a higher score than $\mathcal{G}_0$ given observations $\mathcal{D}$. By exploiting the decomposability property of the scoring functions, for each modified arc (insertion, deletion and reversing), we need only reevaluate the score of the family of node X_i concerned by the modification.

Note that the trajectory generated by *greedy search* allows to map the initial solution $\mathcal{G}_0$ to a local optimum, where the search process is stuck (no candidate in the neighborhood improves the score). To alleviate this problem, the *greedy search* algorithm can be iterated several times, each time starting with a new graph resulting from a random perturbation of $\mathcal{G}_0$. After several iterations, the algorithm chooses the structure leading to the best improvement of the scoring function. Another variant for escaping local maxima consists in performing many *greedy search* with random dataset perturbation [ENFS02] at each time. In this approach, at each *greedy search* restart, the dataset is perturbed through the addition or the deletion of a random number of instances in $\mathcal{D}$. At the end, the best structure over all the perturbed data is chosen. A third approach called *greedy search* with a *tabu* list has been proposed by [Glo90]. This extension prevents the algorithm from reversing recent moves, e.g., it does not allow an edge deletion at iteration $t+1$ knowing that this edge has been added at iteration t. Note that other extensions of the classical *greedy search* have been applied on ordering search spaces ($\mathcal{O}$) in [TK05] and on *Markov equivalence class* search spaces [Chi02b] instead of the DAGs one.

Algorithm *K2*: This algorithm has been proposed by [CH92a]. *K2* is a *greedy search* algorithm which requires an order over $\mathcal{X}$. It tries to approximate the optimal graph $\mathcal{G}$ by searching for each node X_i the set of $\mathbf{Pa}(X_i)$ that locally maximizes the scoring function $score(X_i|\mathbf{Pa}(X_i))$. Therefore, the only operator needed by this algorithm is the addition of an arc between X_i and the set of its candidate parents (represented by its predecessors in the given order $\prec$). The *K2* algorithm proceeds as follows: it starts by setting the parents of each node X_i to the empty set ($\mathbf{Pa}(X_i) = \varnothing$) and by calculating the corresponding score ($score(X_i|\mathbf{Pa}(X_i))$). Next, at each iteration, it tries to choose the best parent X_j to add to X_i, with $X_j \prec X_i$, i.e., the parent that increases the most the score. This process is repeated for each node X_i until no additional parent can enhance the scoring function $score(X_i|\mathbf{Pa}(X_i))$. The pseudocode for *K2* is given in Algorithm 6.

Algorithm 6: K2 Algorithm

```
Input: Data $\mathcal{D}$, variable ordering $\prec$, upper bound $k$
Output: DAG $\mathcal{G}$
1  $\mathbf{A} \leftarrow \varnothing$, $\mathcal{G} \leftarrow \mathcal{G}(\mathcal{X}, \mathbf{A})$
2  $s_{new} \leftarrow score(\mathcal{G}|\mathcal{D})$
3  foreach $X_i \in \mathcal{X}$ do
4      repeat
5          $s_{old} \leftarrow s_{new}$
6          $X_j \leftarrow \underset{X_j \prec X_i \text{ AND } (X_j \to X_i) \notin \mathbf{A}}{\text{Argmax}} score(X_i|\mathbf{Pa}(X_i) \cup \{X_j\})$
7          $s_{new} \leftarrow score(X_i|\mathbf{Pa}(X_i) \cup \{X_j\})$
8          if $s_{new} > s_{old}$ then
9              $\mathcal{G} \leftarrow \mathcal{G}(\mathcal{X}, (\mathbf{A} \cup \{(X_j \to X_i)\})$
10         end
11     until $s_{new} < s_{old}$ OR $|\mathbf{Pa}(X_i)| \geq k$;
12 end
13 return $DAG\ \mathcal{G} = (\mathcal{X}, \mathbf{A})$
```

***Greedy equivalence search (GES)*:** Searching over *equivalence classes* can be considered as an extension of the classical *greedy search* over the DAGs space. As seen earlier, to be able to perform a *greedy search* over *equivalence classes*, a score function allowing the evaluation of partially directed graphs is needed. A common way to tackle this task is to build for each CPDAG one corresponding DAG and, then, to resort to a classical scoring function for DAGs, as presented previously. In order to move over the CPDAG space, Chickering *et al.* [Chi02a] define a set of operators (mentioned in paragraph 3.2.4.1) allowing to perform local graph modifications. Unfortunately, these search operators are very complex and the algorithm needs many operations between CPDAG and DAG spaces to calculate the score. Based on "Meek's Conjecture" [Mee95], the problem of multitude operations has been efficiently addressed [Chi02a] throughout the proposition of an alternative approach that uses only the classical *addition* and *deletion* operators. As proven in [Chi02b], these guarantee to change of *equivalence class* at each arc modification. The proposed algorithm is called *greedy equivalence search* (*GES*). As defined in [Chi02b], it consists of two steps: i) it starts with a CPDAG representing the *equivalence class* of the empty DAG and it performs in the next iterations a set of edge additions until a locally maximal solution is reached; and ii) it then performs the reverse operation, by deleting edges from the previous CPDAG until no new edge deletion improves the score. The *GES* algorithm guarantees to learn the correct *Markov equivalence class* under the following conditions:

- the employed scoring function must be consistent, i.e., if the amount of data $N \to +\infty$, then the structure $\mathcal{G}^*$ that represents exactly the set of independence assertions in $\mathcal{D}$ (*P-map*) shall obtain the highest score.
- the distribution P is *faithful* w.r.t. graph $\mathcal{G}^*$ (see section 3.2.1),
- the observations in dataset $\mathcal{D}$ are *i.i.d*,
- a large dataset is needed.

3.2.5 Hybrid approaches

As mentioned earlier, the purpose of hybrid methods is to extend the advantages and reduce the limitations in both score and constraint-based approaches [TBA06, vDvdGT03a, WL04, FJ09]. Constraint-based approaches are relatively fast and allow performing the learning task in a deterministic way. However, the results returned by the statistical tests rely on an arbitrary significance level α to decide whether two variables can be considered independent. This can make the learning process unstable in the sense that an erroneous decision can trigger the accumulation of compensatory errors in the final structure. Unlike constraint-based approaches, score-based can address the structure learning task even from small datasets. Unfortunately, such methods are slow to converge and the complexity of the search space often prevents them from finding the optimal BN structure. Inspired from both previous approaches, hybrid algorithms begin by constructing the skeleton of the BN (in order to restrict the search space) using a constraint-based approach and, then, they transform it into a BN and they refine it through a score-based search algorithm.

The specificities of each step and the way by which information resulting from the skeleton learning are used in the second step differ between the approaches. Tsamardinos *et al.* [TBA06] proposed the Min-Max Hill Climbing (MMHC), which is one of the most popular BN structure learning in the literature. This algorithm has been extensively evaluated on both synthetic and real-world datasets. It has proved to be very effective in terms of computation times and has shown very accurate results compared to many state-of-the-art approaches such as PC [SGS01], *Three Phase Dependence* [CGK$^+$02], *Greedy Equivalence Search* [Chi02b], *greedy search*, *etc.* As an hybrid approach, MMHC is composed of two steps. In the first step, the algorithm tries to estimate the set of parents and children of each node X_i using the MMPC Algorithm [TAS03] (see section 3.2.3.6). In the second step, MMHC performs a restricted *greedy search* with a *tabu* list using the

Bayesian scoring function BDeu [HGC95]. It should be noted that the search step is constrained to only consider adding an arc $X \rightarrow Y$ if and only if it was learnt by MMPC in the first phase, i.e., $X \in \mathbf{PC}_{\mathcal{G}}^{Y}$ and $Y \in \mathbf{PC}_{\mathcal{G}}^{X}$. The pseudocode of MMHC is given at Algorithm 7.

Algorithm 7: MMHC Algorithm

```
Input: Data $\mathcal{D}$
Output: DAG $\mathcal{G}$
1 //Restrict
2 foreach $X \in \mathcal{X}$ do
3     $\mathbf{PC}_{\mathcal{G}}^{X} \leftarrow MMPC(X, \mathcal{D})$
4 end
5 // Greedy search
6 Perform a greedy search
7 Add $X \rightarrow Y$ iff $X \in \mathbf{PC}_{\mathcal{G}}^{Y}$
8 return DAG $\mathcal{G} = (\mathcal{X}, \mathbf{A})$
```

In [vDVDGT03b], a skeleton-based learning algorithm has been proposed. This approach starts by constructing the skeleton of the BN using only zero and first-order independence tests. Those are then used to restrict the DAGs space in the second step. It should be emphasized that the first step does not consists in learning the correct skeleton but it aims at finding a skeleton that is already close to the true one[11]. By performing zero and first order tests, the algorithm tries on one hand to avoid missing edges (errors of type II) since they cannot be recovered in the next step. On the other hand, it aims to minimize the number of additional edges (errors of type I), since these unnecessary edges increase the size of the search space when performing the score-based search. In the second phase, the algorithm uses a *greedy search* algorithm to navigate through the restricted DAGs space. The same principle holds as in [TBA06]: the algorithm adds, orients and removes only edges from the skeleton to produce the final structure $\mathcal{G}$.

In [WL04], the authors use the same learning skeleton principle as in [vDvdGT03a] together with an evolutionary algorithm at the second step which also works on a restricted search space.

In [FJ09], after learning the skeleton of the BN, the algorithm performs a local *greedy search* to refine and find the optimal orientation of the structure by satisfying as much as possible the set of constraints returned from the skeleton identification phase. The second step is performed as follows: the algorithm starts by constructing a random ordering over the variables and then it returns the optimal orientation consistent with it. The resulting structure is then considered as a starting point of a *greedy search*, which relies on a new operator called *Toggle-Collider* (shown in Figure 3.10), to generate the successor candidates w.r.t. each current structure. Given a triple of variables $X - Z - Y$ in $\mathcal{G}$, the

[11]It should be noted that zero and first order tests do not suffice for identifying all neighbors for all nodes.

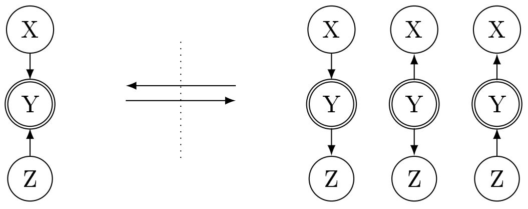


FIGURE 3.10: Example of *Toggle-Collider* operator.

Toggle-Collider either transforms the triple into a *v-structure* (when it is not already the case) or it breaks an existing *v-structure* into serial or divergent connections [FJ09]. It then chooses the transformation that resulted in the highest BDeu score increase. The orientation process continues until no further modification increases the score.

3.3 Conclusion

In this chapter, we have discussed the most common state-of-the-art Bayesian network parameters and structures learning algorithms. We have shown the various details related to each of them. Parameters learning algorithms can be divided into two main groups: *likelihood* and *Bayesian*-based approaches. The former relies on the use of the *likelihood* function as an evaluation criterion to identify the best set of parameter values. The latter relies on the specification of a prior over parameters and tries to estimate the parameters throughout the optimization of the posterior probability of the graph given the dataset.

Concerning the structure learning algorithms, they have been classified into three groups: constraint-based, score-based and hybrid approaches. The main difference between these methods appears in the way in which the set of conditional independences between variables are identified from the dataset. The constraint-based approaches rely on the use of statistical tests to determine the skeleton of the BN, which is then oriented using Meek's deterministic rules. The score-based algorithms can be characterized by the following three features: the considered search space (DAGs, orderings, *Markov equivalence classes*, *etc.*), the set of operators used to move from a given structure to another one, and the heuristics used to find the structure that optimizes the scoring function. Finally, the hybrid approaches try to exploit the advantages and reduce the limitations of both constraint and score-based approaches.

It should be noted that until now Bayesian network parameter and structure learning algorithms essentially work only on discrete datasets (multinomial and binomial) and rely on a list of assumptions under which they are capable to perform an efficient estimations. However, in many real-world applications like in nuclear safety, medicine and robotics, these conditions are not necessarily met. For instance, in these domains, the datasets are often composed of a mixture of continuous and discrete variables and they may also contain some parameters configurations that can lead to the violation of some of the previous assumptions, especially *faithfulness* (as we shall see in the next chapter). In such situations, classical Bayesian network learning algorithms are doomed to be ineffective, which limits their scope, notably by ruling them out of many interesting real application domains.

In the next chapters, we will discuss how the Bayesian network structure learning task has been tackled when the *faithfulness* assumption is ruled out. In this context, we will highlight the different causes responsible for the *unfaithfulness* of the probability distribution and we will present the most common state-of-the-art algorithms dedicated to this problem. In addition, we will discuss how the mixture of continuous and discrete variables in real-world application datasets has been managed in the literature in order to make the task of Bayesian network learning both tractable and efficient.

Chapter 4

Bayes Net structure learning in the absence of *faithfulness*

As shown in the preceding chapters, the DAG of a BN is a graphical representation of the conditional independences among the BN's random variables. Those are captured from the graph by means of the *d-separation* criterion. Earlier works done in [Pea00, SGS01, LMCL12, Luo06, RdMAC08] have shown that learning the Bayesian network structure from observational data heavily relies on the assumption that *DAG-faithfulness* is satisfied. Remind that *faithfulness* means that independences holding in distribution P are equivalent to those entailed by the graph, i.e., $\{X\} \perp_{\mathcal{G}} \{Y\}|\mathbf{Z} \Longleftrightarrow \{X\} \perp_P \{Y\}|\mathbf{Z}$ (see Section 3.2.1). Actually, a BN is an I-map, which means that its semantics is the following: $\{X\} \perp_{\mathcal{G}} \{Y\}|\mathbf{Z} \Longrightarrow \{X\} \perp_P \{Y\}|\mathbf{Z}$. But, when learning the BN from data, the algorithms can only determine conditional independences of the type $\{X\} \perp_P \{Y\}|\mathbf{Z}$ and those are converted into graphical independences, i.e., into $\{X\} \perp_{\mathcal{G}} \{Y\}|\mathbf{Z}$. As a consequence, what the algorithms learn is a D-map, not an I-map. Under *DAG-faithfulness*, those are equivalent, so the algorithms provide a BN that correctly represents the distribution P that generated the observed data.

Unfortunately, *DAG-faithfulness* does not always hold. Through the Hammersley-clifford theorem, it is known to hold for strictly positive probability distributions P but, otherwise, in general, the property is not always satisfied. As an example, when $\mathcal{X}$ contains some random variables having deterministic relations with others, e.g., when $X = f(\mathbf{Z})$ with f a deterministic function (which implies that P is not strictly positive), *DAG-faithfulness* is never satisfied. It should be noted that in many domains, there exist random variables with some deterministic relationships with other variables (e.g., in nuclear safety, medicine, robotics, *etc.*). Other sources of *unfaithfulness* such as *information equivalences* or *equivalent partitions* as mentioned in [LMCL12]. In such cases, all the BN learning algorithms mentioned in the previous chapter, whatever their type (score-based, constraint-based or hybrid), fail to construct a BN that represents correctly the dependences between the random variables. In other words, the BN they construct does not represent the probability distribution P that generated the data. The problem is the twofold: i) under unfaithfulness, some sets of conditional independences cannot be represented by any DAG and the algorithms are misled in the sets of arcs they remove (which can be too large); ii) some statistical conditional independence tests may indicate an independence when, actually, there is none because they are misled by some *statistical ambiguity* property.

This chapter studies some BN learning algorithms that try to overcome the issues raised by unfaithfulness. It is organized as follows: we start by highlighting the most common factors leading to the violation of the probability distribution *faithfulness*. Then, we explain the impact of the absence of *faithfulness* when learning the BN structures (in both skeleton learning and orientation phases). Finally, we present the most common state-of-the-art learning algorithms dedicated to *unfaithful* distributions.

4.1 The causes of unfaithfulness

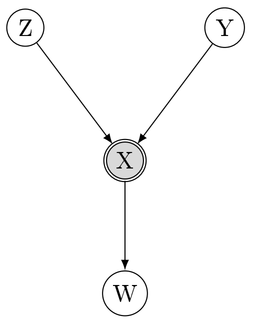
In this section we shall focus our discussion on the three most common factors responsible for the *unfaithfulness* of the distribution, namely the deterministic relationships between random variables, the *information equivalence* and *equivalent partitions* properties.

4.1.1 Deterministic relationships

In the context of BNs, deterministic relationships lead to the notion of *deterministic nodes*, which are defined formally as follows:

Definition 4.1. ***(Deterministic node)*** *Let $\mathcal{X}$ be a set of random variables and let P be a probability distribution over them. A random variable X is said to be deterministic if there exists a subset of variables $\mathbf{Z} \subseteq \mathcal{X} \setminus \{X\}$ and a deterministic function f such that $X = f(\mathbf{Z})$. This relation entails that $P(X = x, \mathbf{Z} = \mathbf{z}) = 0$ whenever $x \neq f(\mathbf{z})$ and 1 otherwise.*

It may be noted that, when distribution P contains some deterministic nodes, it cannot be strictly positive. As a matter of fact, the conditional distribution of a deterministic node given its parents in the network $\mathcal{G}$ will be composed by only 1 and 0 probabilities. As an example, Figure 4.1 shows an example of a CPT of a deterministic node representing the functional relation $X = f(Z, Y)$, where all variables are assumed to be binary. In this example, the relation $X = f(Z, Y)$ tells us that Y and Z contain all the information needed to characterize without uncertainty the value of X. As a consequence, given the deterministic node $X = f(Z, Y)$ in $\mathcal{G}$, arc $X \to W$ can be substituted by arcs $Y \to W$



		$P(X\|Z,Y)$	
		X	
Z	Y	t	f
f	f	0	1
f	t	1	0
t	f	1	0
t	t	0	1

FIGURE 4.1: An example of deterministic CPT: $X = f(Z, Y)$.

and $Z \rightarrow W$ in $\mathcal{G}$ without altering the semantics of the original network. Therefore, we can say that the BN shown in Figure 4.2 is *equivalent* to the one given in Figure 4.1. However, recall that two BNs represent the same set of independences if they have the same skeleton and the same set of v-structures (see the *Markov equivalence* property in Section 2.3 of Chapter 2). As a consequence, from the definition of *Markov equivalence*, both BNs shall not be equivalent. This seeming incoherence precisely results from the unfaithfulness of distribution P and calls for another definition of the equivalence of two BNs. In [GVP90], an extended definition of the concept of *d-separation* criterion, called *D-separation*, has been proposed for this purpose:

Definition 4.2. ***(D-separation)***

Let $\mathcal{G} = (\mathcal{X}, \mathbf{A})$ be a directed acyclic graph (DAG). Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ be three disjoint sets of nodes in $\mathcal{G}$. $\mathbf{X}$ and $\mathbf{Y}$ are said to be D-separated by $\mathbf{Z}$, which is denoted by $\mathbf{X} \perp\!\!\!\perp_{\mathcal{G}} \mathbf{Y} | \mathbf{Z}$ if all the trails between each node in $\mathbf{X}$ and each node in $\mathbf{Y}$ are blocked by either $\mathbf{Z}$ or $\mathbf{W} \in \mathcal{X} \setminus (\mathbf{Z} \cup \mathbf{X} \cup \mathbf{Y})$ such that $\mathbf{W} = f(\mathbf{Z})$.

As can be seen, the definition of the *D-separation* criterion is similar to *d-separation* except that it contains an additional condition dealing with the presence of deterministic nodes in $\mathcal{G}$. Therefore, the *D-separation* guarantees the detection of conditional independences revealed by classical *d-separation* but also those resulting from the presence of

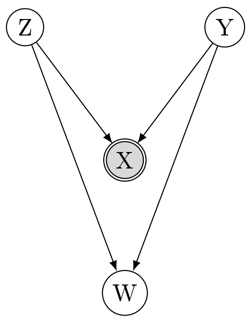


FIGURE 4.2: An equivalent representation of the BN of Figure 4.1.

deterministic nodes in the network. To facilitate the understanding of the independences induced by deterministic nodes in $\mathcal{G}$, we consider the example of the BN structure shown in Figure 4.3 and we assume that $Y = f(W)$ is the only deterministic node in $\mathcal{G}$.

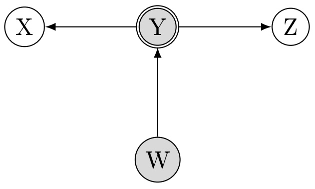


FIGURE 4.3: D-separation between X and Y by W where $Y = f(W)$

By *D-separation*, the conditional independence $\{X\} \perp\!\!\!\perp_{\mathcal{G}} \{Z\}|\{W\}$ holds in this example. This independence can be explained by the fact that, once the value of node W is known, Y does not bring any additional information on X.

4.1.2 Information equivalence

Authors in [LMCL12] provide the definition of a new class of *unfaithfulness* called "information equivalence" that represents a broader class of relations than just deterministic ones.

Definition 4.3. *Two disjoint sets of variables* $\mathbf{X}$ *and* $\mathbf{Y}$ *are considered as information equivalent w.r.t. to a variable* Z *(which forms the reference variable) if and only if there exists a subset* $\mathbf{W} \subseteq \mathcal{X} \setminus (\mathbf{X} \cup \mathbf{Y} \cup \{Z\})$ *for which the following conditions hold in distribution* P*:*

- $(\mathbf{X} \not\perp_P \{Z\}|\mathbf{W}) \wedge (\mathbf{Y} \not\perp_P \{Z\}|\mathbf{W})$
- $(\mathbf{X} \perp_P \{Z\}|(\mathbf{W} \cup \mathbf{Y}))$
- $(\mathbf{Y} \perp_P \{Z\}|(\mathbf{W} \cup \mathbf{X}))$

Figure 4.4 illustrates the information equivalence of $\{X\}$ and $\{Y\}$ w.r.t. reference variable Z. In this example, $\mathbf{W} = \emptyset$ and $\{X\} \perp_P \{Z\}|\{Y\}$ and $\{Y\} \perp_P \{Z\}|\{X\}$.

Given Definition 4.3, we can conclude that equivalence information [LMCL12] can result from different *unfaithful* configurations, among which we can mention:

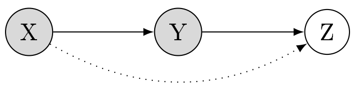


FIGURE 4.4: $\{X\}$ and $\{Y\}$ are information equivalent for Z.

- **Triangle situation**: as discussed in Section 3.2.3.5, this situation holds for each triple of variables X, Y, Z in which each pair is marginally dependent but conditionally independent given the third variable. As it is defined, triangle situation can be seen as a kind of equivalence information where the knowledge given by each pair of variables are equivalent from the viewpoint of the third one.

- **Linear one-to-one relation**: two variables X and Y are related by a one-to-one relation if and only if $f(X) = Y$ and $f^{-1}(Y) = X$. If a one-to-one relation holds between two variables X and Y, this means that any variable Z being dependent on X also depends on Y (and conversely). Both variables thus contain the same knowledge about any other variable in network $\mathcal{G}$ (and in the same form). Therefore, we can say that these variables are indistinguishable, hence they are not both essential for the independence model because they express redundant information.

- **Deterministic relation**: the only difference between information equivalence and deterministic relations is that the former does not entail necessarily a functional relation between the considered variables. Hence, deterministic nodes can be considered as a subset of the information equivalence class. Given a network $\mathcal{G}$, a deterministic relation $X = f(\mathbf{W})$ and a child node Z of X in the network, then X and $\mathbf{W}$ form an information equivalence w.r.t. Z. Therefore, when learning the BN structure, deleting the edge $X - Z$ or $\mathbf{W} - Z$ is equivalent from the point of view of Z.

4.1.3 Equivalent partitions

Equivalent partitions have been discussed in details in [LMCL12]. In the following we will recall the most common notions related to this *unfaithful* parameters' configuration in P, and we will show how it may have an impact on the set of independence tests performed when learning the BN structure.

As discussed earlier, two disjoint sets of variables $\mathbf{X}$ and $\mathbf{Z}$ are dependent if and only if $P(\mathbf{Z}|\mathbf{X}) \neq P(\mathbf{Z})$. That is, there exists at least two different values/tuples $\mathbf{x}_i$, $\mathbf{x}_j$ in the domain of $\mathbf{X}$, hereafter denoted by $val(\mathbf{X})$, such that $P(\mathbf{Z}|\mathbf{x}_i) \neq P(\mathbf{Z}|\mathbf{x}_j)$. However, despite the dependence between $\mathbf{X}$ and $\mathbf{Z}$, it may sometimes occur that a subset of values $\mathbf{v}_\mathbf{X} \subseteq val(\mathbf{X})$ lead to the same conditional distribution of $\mathbf{Z}$, i.e., $P(\mathbf{Z}|\mathbf{x}_i)$ is the same for all $\mathbf{x}_i \in \mathbf{v}_\mathbf{X}$. In such a case, we say that all $\mathbf{x}_i \in \mathbf{v}_\mathbf{X}$ contain the same information about the reference variable $\mathbf{Z}$. Given the previous situation, $val(\mathbf{X})$ can be decomposed into a set of disjoint subsets of values denoted by $val(\mathbf{X})^i$ where distribution $P(\mathbf{Z}|\mathbf{x}_i)$ is the same for all $\mathbf{x}_i \in val(\mathbf{X})^i$. We call this decomposition of $val(\mathbf{X})$ the $\mathbf{Z}$-*partition* of $val(\mathbf{X})$. Let $\mathcal{K}_\mathbf{Z}(\mathbf{X})$ denote the index of a given set $val(\mathbf{X})^i$ in this partition. Then the conditional distribution of $\mathbf{Z}$ depends only on the index of the $\mathbf{Z}$-partition and therefore:

$$P(\mathbf{Z}|\mathbf{X}) = P(\mathbf{Z}|\mathcal{K}_{\mathbf{Z}}(\mathbf{X})) \Longrightarrow \mathbf{X} \perp_P \mathbf{Z}|\mathcal{K}_{\mathbf{Z}}(\mathbf{X}). \tag{4.1}$$

Before explaining the dependences/independences resulting from equivalent partitions, we start by defining the notion of a binary relation between two sets of random variables $\mathbf{X}$ and $\mathbf{Y}$. By abuse of notation, a binary relation $\mathcal{R}$ between $\mathbf{X}$ and $\mathbf{Y}$ is a binary relation over their domains, i.e., it is a subset of $val(\mathbf{X}) \times val(\mathbf{Y})$. For each pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{R}$, we use the classical notation $\mathbf{x}\mathcal{R}\mathbf{y}$.

Definition 4.4. ***(Equivalent partitions)*** *Let $\mathbf{X}$ and $\mathbf{Y}$ be two sets of random variables. Let $V_{\mathbf{X}} = \{val(\mathbf{X})^1, \dots, val(\mathbf{X})^{r_{\mathbf{X}}}\}$ be a partition of $val(\mathbf{X})$. A binary relation $\mathcal{R}$ over $\mathbf{X}$ and $\mathbf{Y}$ is an equivalent partition in $val(\mathbf{Y})$ to partition $V_{\mathbf{X}}$ if and only if the following conditions are satisfied:*

- *$\forall \mathbf{x}_i \in val(\mathbf{X})^i, \mathbf{x}_j \in val(\mathbf{X})^j$ where $i \neq j$ and $\forall \mathbf{y}_i \in val(\mathbf{Y})$ with $\mathbf{x}_i\mathcal{R}\mathbf{y}_i$, then $\neg(\mathbf{x}_j\mathcal{R}\mathbf{y}_i)$ holds.*
- *$\forall i \in \{1, \dots, r_{\mathbf{X}}\}$, $\exists \mathbf{x}_i \in val(\mathbf{X})^i$, $\exists \mathbf{y}_j \in val(\mathbf{Y})$ such that $\mathbf{x}_i\mathcal{R}\mathbf{y}_j$.*

From definition 4.4, we can deduce that each partition $val(\mathbf{X})^i$ is associated with another one $val(\mathbf{Y})^j$ for $\mathbf{Y}$ (first condition) which is non-empty (second condition). Similarly to the equivalence information and deterministic relations (in terms of resulting independences), the equivalent partitions lead to the following theorem:

Theorem 4.1. ***(Equivalent partitions)*** *Let $\mathbf{X}$ and $\mathbf{Y}$ be two sets of random variables and let Z be another variable. The following property:*

$$(\{Z\} \not\perp_P \mathbf{X} \wedge \{Z\} \perp_P \mathbf{Y}|\mathbf{X}) \Longrightarrow \{Z\} \perp_P \mathbf{X}|\mathbf{Y}$$

is equivalent to the fact that the binary relation $\mathcal{R}$ over $\mathbf{X}$ and $\mathbf{Y}$ defined by $\mathbf{x}_i\mathcal{R}\mathbf{y}_j \Longleftrightarrow P(\mathbf{x}_i, \mathbf{y}_j) > 0$ is an equivalent partition in $val(\mathbf{Y})$ to the Z-partition of $val(\mathbf{X})$.

The reader can also refer to [LMCL12] for additional details about equivalent partitions.

4.2 Absence of faithfulness when learning the Bayes Net structure

4.2.1 When learning the skeleton

Remind that learning the skeleton of a BN consists in learning from data $\mathcal{D}$ the set of direct dependences (represented by means of non-oriented edges in $\mathcal{G}$) between random variables. When performing the learning task, we would like to have $\{X\} \perp_{\mathcal{G}} \{Y\}|\mathbf{Z} \iff \{X\} \perp_P \{Y\}|\mathbf{Z}$ because it would imply that $\mathcal{G}$ contains an arc between X and Y if and only if X and Y are probabilistically dependent given any set $\mathbf{Z}$ [Pea88]. The previous assumption is called the *adjacency faithfulness*. As shown before, it guarantees the correctness of learnt edges between nodes in the skeleton $\mathcal{G}$. Unfortunately, as we have seen earlier, this equivalence does not hold when the distribution P contains some *unfaithful* parameters configurations. Recall that one of the main crucial conditions for the *faithfulness* assumption is the Intersection axiom (see Section 2.2 of Chapter 2) which is defined as follows:

$$(\mathbf{X} \perp_P \mathbf{Y}|(\mathbf{Z} \cup \mathbf{W})) \wedge (\mathbf{X} \perp_P \mathbf{W}|(\mathbf{Z} \cup \mathbf{Y})) \Rightarrow (\mathbf{X} \perp_P (\mathbf{Y} \cup \mathbf{W})|\mathbf{Z}). \tag{4.2}$$

The intersection property is ruled out when the distribution contains zero probability. To facilitate the understanding of this problem, we consider the example of BN shown in Figure 4.5, where $\mathcal{X} = \{X, Y, Z, W_1, W_2\}$ and $Y = f(W_1, Z, W_2)$ is the only deterministic node in $\mathcal{G}$. By *D-separation*, the graph represents the following conditional independences:

- $\{X\} \perp\!\!\!\perp_{\mathcal{G}} \{Y\}|\{Z, W_1, W_2\}$ because, once the values of Z, W_1, W_2 are known, Y does not bring anymore information on X since $Y = f(W_1, Z, W_2)$.
- $\{X\} \perp\!\!\!\perp_{\mathcal{G}} \{W_1, W_2\}|\{Z, Y\}$ by d-separation.
- $\{X\} \not\perp\!\!\!\perp_{\mathcal{G}} \{Y, W_1, W_2\}|\{Z\}$ by d-separation.

By substituting $\mathbf{X} = \{X\}$, $\mathbf{Y} = \{Y\}$, $\mathbf{Z} = \{Z\}$ and $\mathbf{W} = \{W_1, W_2\}$ in Equation (4.2), we can observe that the above three independences do not satisfy the intersection axiom.

The same violation of the intersection axiom occurs also for information equivalence, equivalent partitions and all *unfaithful* parameters configurations. In general, whenever a deterministic node —or any node for which the conditions of information equivalence or equivalent partition hold— has a child in $\mathcal{G}$, then the *adjacency-faithfulness* is lost, i.e., $X = f(\mathbf{Y})$ makes X conditionally independent from the rest of the network given $\mathbf{Y}$, even of its immediate children in the network (if they exist). As an example, in the BN of Figure 4.5, where $Y = f(W_1, Z, W_2)$, we have that $\{X\} \not\perp\!\!\!\perp_{\mathcal{G}} \{Y\}|\{W_1, Z, W_2\}$. However, $\{X\} \perp_P \{Y\}|\{W_1, Z, W_2\}$ because the values of W_1, Z, W_2 determine the value of Y, hence inducing that $P(X|Y, W_1, Z, W_2) = P(X|W_1, Z, W_2)$. Therefore, most BN structure learning algorithms would delete edge $Y - X$ from the skeleton $\mathcal{G}$. Remind that this problem also holds for the other sources of *unfaithfulness* mentioned in Definitions 4.3 and 4.4.

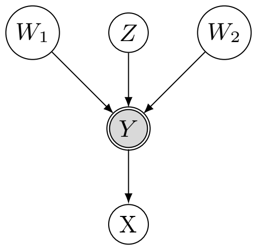


FIGURE 4.5: A BN with one deterministic relation $Y = f(W_1, Z, W_2)$.

4.2.2 When orienting the skeleton

By assuming adjacency faithfulness, the authors in [RZS06] explore the consequences of the probability distribution *unfaithfulness* when performing the orientation of the BN skeleton. It should be noted that the adjacency *faithfulness* states that if two variables are adjacent in $\mathcal{G}$ they remain conditionally dependent given any subset of variables. This assumption is necessary to recover the correct skeleton of the true BN. In the following, we will give the definition the *orientation faithfulness.*

Definition 4.5. ***(Orientation faithfulness)*** *For a given DAG $\mathcal{G}$, the orientation faithfulness for each unshielded triple $\langle X, Z, Y\rangle$ must verifies the following conditions:*

- *if $\langle X, Z, Y\rangle$ forms a* v-structure *($X \to Z \leftarrow Y$), then X and Y are conditionally dependent given any subset of variables $\mathbf{W} \subseteq \mathcal{X} \setminus \{X, Y\}$ that contains Z,*
- *otherwise, X and Y are conditionally dependent given any subset $\mathbf{W} \subseteq \mathcal{X} \setminus \{X, Y\}$ that does not contain Z.*

The orientation faithfulness assumption allows to ensure the correctness of the *collider and non-collider* edges identifications [1]. Recall that an unshielded triple $\langle X, Z, Y\rangle$ is considered as a *v-structure* if and only if Z does not belong to the separating sets of X and Y ($SepSet_{XY}$). As shown in [RZS06], it is possible to find probability distributions that satisfy the adjacency faithfulness w.r.t. to the true DAG but not orientation-faithfulness. In other words, *adjacency faithfulness* does not necessarily imply *orientation faithfulness.* In the following example, borrowed from [RZS06], we consider the structure shown in Figure 4.6. Here, the independences $\{X\} \perp_P \{Y\}$ and $\{X\} \perp_P \{Y\}|\{Z\}$ that can be obtained from the distribution P satisfy the *adjacency faithfulness* assumption since both of them lead to the deletion of edge $X - Y$, hence leading to the skeleton of the BN shown in Figure 4.6. Suppose that learning the BN skeleton with the PC algorithm actually leads to the deletion of edge $X - Y$. Then, when performing the skeleton orientations during the second phase of PC, the unshielded triple $\langle X, Z, Y\rangle$ will be converted into a *v-structure* since $SepSet_{XY} = \varnothing$. This orientation leads to a contradiction since

[1]If the orientation step fails to identify the correct BN, Meek's rules used to perform the rest of the orientation output an incorrect orientation.

$\{X\} \perp_P \{Y\}|\{Z\}$ cannot be represented by the *v-structure* $X \to Z \leftarrow Y$. Moreover, if another orientation were chosen, this would lead to a serial or divergent connection which is not confirmed by the marginal independence $\{X\} \perp_P \{Y\}$.

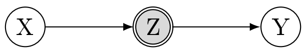


FIGURE 4.6: A simple BN with serial connection between X, Y and Z.

4.3 Related work

Several approaches have been proposed in the literature to address the problem of structure identification when some random variables are deterministic. For example, as mentioned in Definition 4.2 [GVP90] the definition of the d-separation criterion has been adapted to cope with deterministic variables, hence resulting in D-separation. As discussed earlier, the latter adds to Definition 2.5 a condition that handles the conditional independences resulting from deterministic relations. Unfortunately, applying this definition instead of that of d-separation is not sufficient to decrease the difficulty of learning the BN structure in the presence of deterministic random variables.

In Spirtes *et al.* [SSM+96][2], it is considered that deterministic variables are not essential for the independence model since they only express redundant information. Therefore, it is proposed to filter them out from data before learning the structure of the BN. However, it may be pointed out that, with datasets of limited sizes, such an approach can fail to learn correctly the structure of the BN. For instance, assume that the graphical structure $\mathcal{G}$ of a BN is such that $\mathcal{X} = \{X, W, Y_1, \dots, Y_k, Z_1, \dots, Z_r\}$ and that all these variables are Boolean. Assume in addition that X is deterministically defined as the exclusive OR of $Y_1, \dots, Y_k$ and that W depends stochastically on $X, Z_1, \dots, Z_r$. Then, by removing X, W depends on $Y_1, \dots, Y_k, Z_1, \dots, Z_r$, which, to be detected, requires an independence test over a contingency table of 2^{k+r+1} cells instead of 2^{r+2} cells for the dependence test of W with $X, Z_1, \dots, Z_r$. As another example, consider a BN whose variables are X, Y, Z, T, with $X = f(Y, Z)$, and whose arcs are $Y \to X$, $Z \to X$, $X \to T$. Then $\{X\}$ *d-separates* $\{Y, Z\}$ from T whereas, by removing X, this conditional independence cannot be taken into account in the BN. Moreover, as shown in [J.L07], deterministic variables play an essential role by providing an efficient insight in the underlying studied system when performing causal analysis.

Rodrigues de Morais *et al.* [RdMAC08] addressed structure learning through an original determination of the Markov blanket (**MB**) of each node X of $\mathcal{X}$, i.e., the minimal set of nodes $\mathbf{Z}$ such that, given $\mathbf{Z}$, X is independent from the rest of the network (in a BN, the **MB** of X is the set of its parents, children and the other parents of the

[2]Note that this approach supposes that deterministic variables are known a priori, which is not always the case in many real application domains.

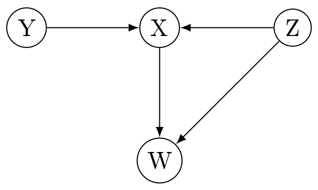


FIGURE 4.7: An example where $W \in \mathbf{PC}_{\mathcal{G}}^{Y}$ and $Y \notin \mathbf{PC}_{\mathcal{G}}^{W}$.

children). The idea relies on the identification of only the set of parents and children ($\mathbf{PC}_{\mathcal{G}}^{\mathbf{X}}$) of the nodes X, as specified in [TBA06], i.e., $\mathbf{PC}_{\mathcal{G}}^{X}$ is equal to $\mathbf{MB}(X)$ minus the parents of the children of X. As discussed in the previous chapter (Section 3.2.3.6), in Tsamardinos *et al.* [TBA06], $\mathbf{PC}_{\mathcal{G}}^{X}$ is constructed incrementally, starting from an empty set and adding into it one by one new nodes Y such that $\{X\} \not\perp_P \{Y\}|\mathbf{Z}$ for all sets $\mathbf{Z}$ included in the current set $\mathbf{PC}_{\mathcal{G}}^{X}$. In a second step, for each variable $Y \in \mathbf{PC}_{\mathcal{G}}^{X}$, if there exists a set $\mathbf{Z} \subseteq \mathbf{PC}_{\mathcal{G}}^{X} \setminus \{Y\}$ such that $\{X\} \perp_P \{Y\}|\mathbf{Z}$, then Y is removed from the candidates parents-children $\mathbf{PC}_{\mathcal{G}}^{X}$. In a DAG-faithful context, two different BNs representing the same independence model have precisely the same $\{\mathbf{PC}_{\mathcal{G}}^{X_i}\}_{i=1}^{n}$ sets and it is shown in [TBA06] that a BN should contain an arc between nodes X and Y if and only if $Y \in \mathbf{PC}_{\mathcal{G}}^{X}$ *and* $X \in \mathbf{PC}_{\mathcal{G}}^{Y}$. The *and* condition as shown in [TBA06] is necessary to avoid false positive problems. For a better understanding of this problem, we consider the example of BN depicted in Figure 4.7. Here, node Z cannot be added to $\mathbf{PC}_{\mathcal{G}}^{Y}$ because, if the BN is faithful, $\{Y\} \perp_P \{Z\}$. Node X is not d-separated of Y given $\emptyset$ and given $\{W\}$, so $X \in \mathbf{PC}_{\mathcal{G}}^{Y}$. Similarly, $W \in \mathbf{PC}_{\mathcal{G}}^{Y}$ because W and Y are not *d-separated* given $\emptyset$ and $\{X\}$. Conversely, $Y \notin \mathbf{PC}_{\mathcal{G}}^{W}$ because $X, Z \in \mathbf{PC}_{\mathcal{G}}^{W}$ and $\{Y\} \perp_P \{W\}|\{X, Z\}$. The "and" condition is thus compulsory to avoid adding arc $Y \rightarrow W$ in the graph. However, in a DAG-unfaithful context, this condition fails to produce the correct BN. For instance, if X is a deterministic node in Figure. 4.7, i.e., $X = f(Y, Z)$, then $Y, Z \in \mathbf{PC}_{\mathcal{G}}^{X}$ but this will rule out W belonging to $\mathbf{PC}_{\mathcal{G}}^{X}$ because $\{X\} \perp_P \{W\}|\{Y, Z\}$ since X does not bring any more information to W when Y and Z are already known. Therefore the "and" condition will prevent the existence of arc $X \rightarrow W$ in the learned structure. To cope with this problem, it is suggested in [RdMAC08] to substitute the "and" operator by an "or" operator. This solution allows to address the problem entailed by deterministic nodes, but as shown previously, this "or" operator will add arc $Y \rightarrow W$, which is not necessary for the other parts of structure where *faithfulness* holds.

In Luo [Luo06], association rules miners are used to detect deterministic relations. Those are used in a constraint based algorithm (inductive causation (IC)) to build the BN as follows:

1. X and Y are connected by an arc if and only if $\{X\} \not\perp_P \{Y\}|\mathbf{Z}$ for every set $\mathbf{Z} \subseteq \mathcal{X} \setminus \{X, Y\}$ such that neither X nor Y are determined by $\mathbf{Z}$;

2. the unshielded triple $(X - Z - Y)$ is considered as a *v-structure* if and only if there exists a set $\mathbf{W} \not\supseteq \{Z\}$ such that $\{X\} \perp_P \{Y\}|\mathbf{W}$ and neither X nor Y are determined by $\mathbf{W}$.

This method, while quite efficient, is unable to remove some arcs. For instance, as shown in Figure 4.8, if $\mathcal{X} = \{X, Y, Z, W, U\}$, with $X = f(Y, Z)$, and if the BN has a diamond shape with X at the bottom, W at the top and Y, Z on the middle, then rule 1 above prevents discarding arc $X \to W$ or $X \leftarrow W$.

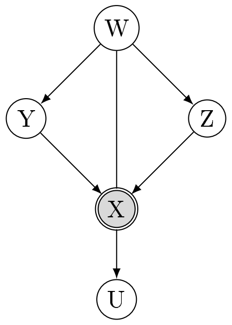


FIGURE 4.8: An example of problem of statistical indistinguishably between X and W.

Lemeire *et al.* [LMCL12] provides the definition of a new class of *unfaithfulness* called “information equivalence” that follows from a broader class of relations than just deterministic ones as shown in Definition 4.3. In addition to the latter, this class also contains equivalence partitions discussed in Section 4.4 where each one is a source of *unfaithfulness*. The algorithm developed in [LMCL12] consists in testing, for each independence returned by the statistical test $\{X\} \perp_P \{Y\}|\mathbf{Z}$, whether an equivalence information can be found by testing additionally whether $\{X\} \perp_P \mathbf{Z}|(\{Y\} \cup \mathbf{W})$ and $\{Y\} \perp_P \mathbf{Z}|(\{X\} \cup \mathbf{W})$ hold and whether the conditions of Definition 4.3 are satisfied. If this is the case, the arc between X and Y is not removed from $\mathcal{G}$. Information equivalence encompasses dependences due to deterministic relations but, in practice, we observed that independence tests often fail to reveal true information equivalences.

Finally, to cope with this problem, Ramsey *et al.* [RZS06] have proposed an extended version of the PC algorithm called the conservative PC (CPC)[3] which deals with the problem of *unfaithfulness orientation*. The CPC algorithm replaces the *v-structure* identification step in the PC algorithm by the following instructions: for each unshielded triple $\langle X, Z, Y\rangle$, all potential parent subsets (adjacent nodes) of X and Y are checked:

(a) if Z is not part of the subsets of variables that renders X and Y independent, orient the triple as follows: $X \to Z \leftarrow Y$,

(b) if Z is in all the subsets of variables that render X and Y independent, do not orient the triple, i.e., it is not a *v-structure*,

[3]Not to be confused with **CPC** of algorithm MMPC mentioned in Chapter 3.

(c) if none of the above conditions hold, mark the triple as *"unfaithful"*.

For more details about CPC algorithm, the reader should refer to [RZS06].

4.4 Conclusion

In this chapter we have discussed the most common sources leading to the *unfaithfulness* of the probability distribution and we have shown their impacts on the BN learning task. In the adjacency search step, *unfaithful* parameters configurations such as deterministic nodes, information equivalence or equivalent partitions may lead to a set of conditional independences that cannot be represented *faithfully* by a BN structure. The orientation step and more particularly the identification of the collider connections (or *v-structure*) cannot be carried out since many marginal independences from the distribution lead to either erroneous or ambiguous edge orientations.

We have also discussed the most common state-of-the-art approaches dedicated to BN learning task when the *faithfulness* of the distribution is ruled out. These algorithms, as we have seen earlier, try to deal with deterministic relations and other sources of *unfaithfulness* but all of them have serious shortcomings. In those methods, *unfaithful* parameters configurations are perceived as sources of problems, hence they try to perform additional statistical tests in order to cope with unfaithfulness. However, as we shall see in following chapters, deterministic relations as well as the Bayesian network structure properties (see Chapter 2) bring valuable information that cannot be exploited by classical algorithms, thereby preventing them to be fully effective.

In the next chapter, we will develop a new approach with an opposite view: we try to exploit deterministic relations together with the BN properties as new opportunities to help the learning algorithm to add, remove and orient edges in a principled manner.

Part II

Contributions

Chapter 5

An efficient Bayes Net structure learning in the presence of deterministic relations

As discussed in the previous chapters, in many real-world applications, the set of assumptions on which rely the majority of the structure learning algorithms is not fully satisfied. As a consequence, those algorithms are no more guaranteed to provide the BNs that the user looks for. In addition, when they converge, there is usually no criterion to assess the "distance" between the returned BN and the "true" one. The different exceptions that may occur in the probability distribution that lead to the violation of the aforementioned assumptions have led researchers to propose tailored learning algorithms that could overcome these issues. For instance, some algorithms take into account the existence of deterministic relations [RdMAC08, Luo06], information equivalence, equivalence partitions [LMCL12], *etc.*

However, for important critical applications, such algorithms can be ineffective and provide very unsatisfactory solutions. Such a situation, which was actually our starting point, arises in nuclear safety, notably in problems of nuclear accident scenario reconstruction from sensors' partial observations. In this domain, BNs and their inference engines seem well-suited for helping Decision Makers make the best decisions in order to limit as much as possible the consequences of the accident. The difficulty of the BN learning task in this application domain arises from the presence of deterministic dependences between random variables. Those essentially represent equations modeling the various physical and chemical phenomena occurring in a damaged nuclear power plant during a severe accident. Remind that classical structure learning algorithms rely on the *faithfulness* assumption which assumes that independences holding in distribution P are equivalent to those entailed by the graph, i.e., there exists no conditioning set of variables that can make directly related variables in $\mathcal{G}$ become conditionally independent.

Based on this property, the algorithms determine the conditioning sets in the distribution based on the dataset and each dependence/independence is converted into its graphical counterpart in $\mathcal{G}$. As such, these algorithms learn a D-map instead of learning the I-map which is a BN. Under *faithfulness*, D-maps and I-maps are equivalent. Unfortunately, the *faithfulness* property does not hold in the nuclear safety domain due to the presence of deterministic relations among some variables in the distribution. In this case, even state-of-the-art algorithms dedicated to structure learning with deterministic variables [Luo06, LMCL12, RdMAC08, GVP90] prove to be ineffective to discover many dependences/independences. In our opinion, this is due to the fact that they strongly rely on statistical tests that often fail to provide the right answers when *faithfulness* occurs. In addition, they do not necessarily exploit valuable information that may come from deterministic relations. The problem, e.g., in nuclear safety, is that such a failure is a serious issue especially if the results returned by the BN are to be taken into account in the reactor handling process.

In our case, we advocate to exploit the properties of both deterministic relations and those entailed by Bayesian networks when learning the structure. More precisely, we propose a new hybrid algorithm, combining a constraint-based approach with a greedy search, that includes specific rules dedicated to deterministic nodes that significantly reduce the incorrect learning. It consists in constructing at a first step the skeleton of the BN, i.e., its graphical structure in which the arcs are substituted by edges, and, then to convert it into a directed graph which is subsequently refined by a greedy search. This kind of technique is known to be very effective [TBA06]. The originality of our algorithm lies in the rules applied in its three steps, that exploit the features of the deterministic relations:

1. to discard arcs that should not belong to the BN but that cannot be detected by scoring or independence tests;
2. to orient optimally the arcs adjacent to deterministic nodes.

In addition, our algorithm uses an original way to compute the conditional independences it needs to construct the BN's graphical structure.

The rest of the chapter is organized as follows: we start by presenting a concrete example related to the nuclear safety area for which the *faithfulness* property is ruled out. Then, we present the set of assumptions under which our proposed approach works. Next, we provide the details of our approach and justify its correctness. Finally, some concluding remarks are given in the conclusion. Some experimentations highlighting the effectiveness and efficiency of the method are provided in the next chapter.

5.1 A concrete case of *unfaithfulness* in the nuclear field

In the following, we discuss a concrete phenomenology that may occur in severe accident situations and for which the *faithfulness* property is ruled out. This example results from the ideal gas law.

The ideal gas law is a thermodynamic model used to determine the behavior of real gases in the containment at low pressure. When the gas molecules collide with the wall of the container (e.g., the containment of the reactor), they are exerting a force over it. This increases the pressure inside the container. Ideal gas low expresses the relationships between pressure, volume, temperature and the moles of the gas through the following deterministic relation:

$$PV = n_{mol}.R.T \tag{5.1}$$

where :

- P is the pressure of the gas (in Pa);
- V is the volume of the container (in m^3)
- n_{mol} is the number of moles (in mol)
- R is a universal constant fixed to $R =$ 8,3144621 $J.K^{-1}.mol^{-1}$
- T is the temperature (in K) ;

From Equation (5.1), we can conclude that each element composing the ideal gas law model can be deduced from the other terms via a deterministic relation as can be seen in Table 5.1.

$P = \frac{n_{mol}RT}{V}$	Pressure
$V = \frac{n_{mol}RT}{P}$	Volume
$n_{mol} = \frac{PV}{RT}$	Mole
$T = \frac{PV}{n_{mol}R}$	Temperature

TABLE 5.1: Equations derived from the Ideal Gas Law

To facilitate the understanding of the ideal gas law model, we consider the example of a cylinder with a moving piston as shown in Figure 5.1[1]. This example illustrates the effect of gas molecules (represented by balloons) which exert a small force (Fr) each time they collide with the walls of the container (or the cylinder), hence increasing the pressure

[1]Picture taken from http://www.eoht.info/page/Social+ideal+gas+law

inside it. The more frequently the gas molecules collide with the walls, the greater the pressure exerting on the walls of the container. Suppose that the gas injected into the enclosed volume 2 is the Air. By forcing more and more Air moles into this small volume, the pressure in the container increases more and more, hence this exerts a force on the moving piston. This force ($Fr = P \cdot Ar$, where Ar is the area of the piston base and P is the pressure of gas) is sufficient to move the piston toward the top part of the container with a distance ΔL, which therefore leads to the increase of the volume of the cylinder to ΔV. The increasing volume phenomena (volume 2) shown in Figure 5.1 can be expressed by the fact that injection of more and more Air into the cylinder makes the gas molecules in volume 2 get more and more compressed. As a consequence, the proportion of the total volume taken by the molecules gets higher and higher. When the volume available around the molecules converges to zero (molecules touching each other), the gas needs more and more space. For this reason, it pushes the stamp toward the top part of the cylinder. As can be noticed in Figure 5.1, the compression of the molecules of the gas is not the same in both volumes. It should be noted that the relation between volume (V) and pressure (P) can easily be deduced from the first equation shown in Table 5.1. As the volume of the containment increases (denominator), the pressure decreases.

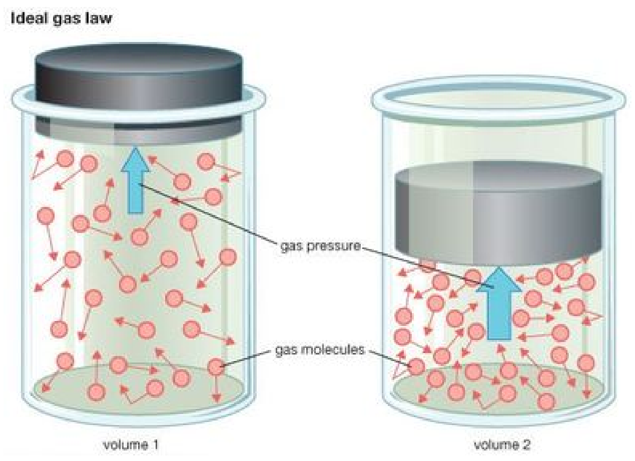


FIGURE 5.1: Ideal gas law with a piston and closed cylinder example

Figure 5.2.(a) depicts the BN structure that represents the ideal gas law related data and its impact on the failure of the containment (depicted by the variable CF). In this example, the deterministic variable "pressure (P)" is represented with a shaded node with double-borders. As can be seen in Table 5.1, Pressure (P) is determined by $n_{mol}(= n_{H_2}+ n_{H_2O}+ n_{Air})$, T, and V, which form its parents in $\mathcal{G}$. Note that term R of the ideal gas law equation cannot be taken into account in the BN since its value remains always the same (it is a constant, not a random variable). The set of conditional independences encoded by the structure of Figure 5.2.(a) do not rule out the rest of the deterministic relations derived from the ideal gas law equation (see Table 5.1). To understand how the other deterministic relations induced by the ideal law gas model are represented by the BN of Figure 5.2, let us delve into the details of the conditional independences semantics

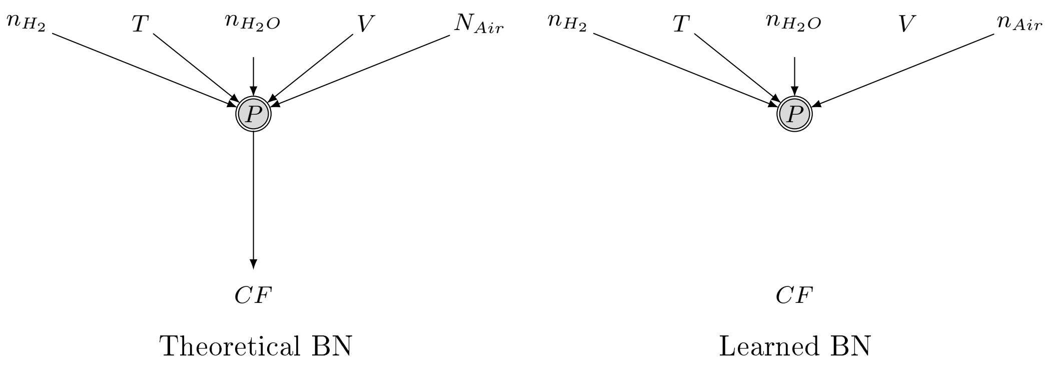


FIGURE 5.2: BNs resulting from the ideal gas low example

encoded by the model. In this example, the considered variables n_{mol}, V and T of the gas perfect law are marginally independent, since the state of each one of them does not have any effect on the values taken by the others. Such independences can be easily represented by means of a *v-structure* (which corresponds to the family of node P) as shown in the BN of Figure 5.2.(a). Given the definition of *v-structures*, it turns out that, by conditioning on the value of P, the marginal independences between parent nodes are transformed into conditional dependencies, i.e., each value(s) taken by a subset of these variables (parents nodes of P) changes our belief on the other variables when P is instantiated. The set of connections that can be established after conditioning on P are represented by red arcs as shown in Figure 5.3.

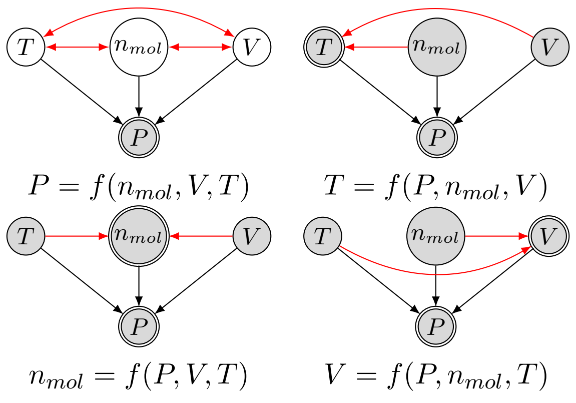


FIGURE 5.3: Communication flow between the ideal gas law variables

By *D-separation*, the model of Figure 5.2 entails the following conditional independences:

$$(\{n_{mol}, T, V\} \perp_{\mathcal{G}} \{CF\}|\{P\}) \wedge (\{n_{mol}\} \perp_{\mathcal{G}} \{T\}) \wedge (\{n_{mol}\} \perp_{\mathcal{G}} \{V\}) \wedge (\{V\} \perp_{\mathcal{G}} \{T\}).$$

However, by functional relation, we know that nodes V, n_{mol}, and T contains all the relevant information about P. Hence the latter becomes independent from the rest of the variables given its parents in $\mathcal{G}$. This leads to:

$$\{P\} \perp_P \{CF\}|\{n_{mol}, T, V\} \tag{5.2}$$

Both independences $(\{P\} \perp_P \{CF\}|\{n_{mol}, T, V\})$ and $(\{n_{mol}, T, V\} \perp_P \{CF\}|\{P\})$ cannot be represented *faithfully* in the network because the deletion of their corresponding arcs leads to an erroneous BN structure. In addition, maintaining both marginal dependencies with CF would represent a redundant information and, thus, would rule out the minimality condition of the BN structure. The structure resulting from classical learning algorithms with ideal gas law data is shown in Figure 5.2.(b), where the arc between P and CF is erroneously deleted by the conditional independence shown in Equation (5.2).

5.2 A new learning algorithm suited for deterministic relations

In the following, we will introduce a new algorithm to overcome the issues raised in the preceding section. This one is designed for learning the BN structure when some deterministic nodes exist in dataset $\mathcal{D}$. Like several other BN learning algorithms, ours is an hybrid which is composed by two phases. In the first phase, a skeleton of the BN is determined (or at least an approximation). This skeleton serves in a first phase to approximately describe the set of dependencies between random variables, hence it can be used as a prior about possible existing arcs in the final BN. In the second phase, the skeleton is transformed into a BN, which is refined through a score-based search algorithm. It must be emphasized that in all the steps (skeleton and orientation), we dedicate special rules to address the *unfaithfulness* induced by deterministic nodes.

Before giving the details of our approach, we start this section by introducing the set of assumptions under which our approach is supposed to work effectively. Next, we will provide the details related to a well known information theoretic metric called the *Shannon entropy* function, since it will be used in our approach to help determining the *deterministic relations* when learning the BN structure from dataset $\mathcal{D}$.

5.2.1 Assumptions

In the sequel, we will assume that DAG-*unfaithfulness* results only from deterministic nodes and that, whenever $X = f(\mathbf{Z})$, all $Z \in \mathbf{Z}$ are automatically parents (or causes) of X in the BN's graphical structure $\mathcal{G}$. One-to-one relationships are not considered in our scope, i.e., if we detect two deterministic nodes that are in a one-to-one mapping in distribution P, we simply discard one of them to avoid redundancies in the data. In addition, we assume that the problem of *unfaithfulness* is local and does not propagate to the rest of the network when performing the BN structure learning, i.e., the *unfaithfulness* concerns only the adjacency around each deterministic node. In other words, whenever there is no deterministic node in some parts of the searched structure $\mathcal{G}$, then the *faithfulness* assumption holds and classical learning algorithm identify correctly these latter. An example of *unfaithfulness* boundaries in $\mathcal{G}$ is represented in Figure 5.4, where the impact of deterministic node $Z = f(X, Y)$ during the BN learning concerns only the identification of arcs $Z - U$ and $Z - W$.

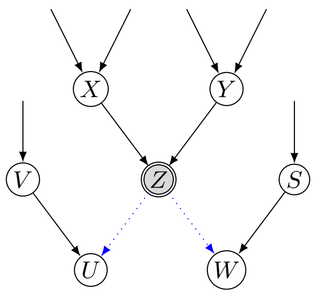


FIGURE 5.4: Boundaries of *unfaithfulness* in a BN with $Z = f(X, Y)$.

Finally, we suppose that the training dataset is large and fully observed. This guarantees the correctness of the scoring/statistical functions used to compute the set of conditional independences between random variables.

5.2.2 Deterministic nodes detection with an entropy function

In general, the (Shannon) entropy function is used to compute the expected uncertainty that characterizes a random variable X, i.e., it tells us the amount of information stored into X. The entropy of X, hereafter denoted by $H(X)$, is calculated by the following equation:

$$H(X) = -\sum_{i=1}^{r_x} p(x_i) \log_2 p(x_i). \tag{5.3}$$

It must be noted that the entropy function does not depend on the values taken by the random variable, but only on its probability distribution. The "log" function used in Equation (5.3) is in base 2, hence the unit of information uncertainty is expressed in bits. For instance, we take the following sequence of length 20 bits which is composed by 1 and 0 values: 00000010101000100001. From the previous sequence, we can deduce that the probabilities of 0 and 1 outcomes are respectively 0.75 and 0.25[2]. By referring to Equation (5.3), there exists an encoding of the sequence that uses only (on average) $H(X) = 0.188$ bits for each bit of the sequence. The latter can thus be fully encoded (without any loss) in 3.76 bits instead of 20 bits.

It should be noted that when the distribution is uniform $X \sim \mathcal{U}(\{1, .., N\})$, the uncertainty reaches its maximum, with an entropy value equal to $\log N$[3]. To facilitate the

[2]The probabilities of 0 and 1 values are obtained by simply counting the occurrences of each value and dividing them by $N = 20$.

[3]The general rule of the entropy function says that if a given value is more probable than the others, the uncertainty of X decreases, i.e., as knowledge grows, the entropy value decreases.

understanding of the maximum value that $H(X)$ may take, let p_1 and $p_2 = 1 - p_1$ denote respectively the probabilities of 0 and 1 values in the previous sequence. We should keep in mind that searching the optimal value of p_1 can be considered as a constrained optimization problem:

$$\begin{aligned} &\text{Argmax}_{p_i}\, H(X) \\ &\text{such that } \sum_{i=1}^{2} p_i = 1 \end{aligned} \tag{5.4}$$

Such an optimization problem can be addressed using the approach of Lagrange multipliers. To maximize Equation (5.4), it suffices to maximize the following unconstrained equation:

$$\begin{aligned} \mathcal{L} &= H(X) + \lambda g(X) \\ &= \left(-\sum_{i=1}^{2} p_i \log p_i\right) + \lambda\left(\sum_{i=1}^{2} p_i - 1\right) \end{aligned}$$

where λ represents the Lagrange Multiplier. Finding the optimal value p_i consist in maximizing the previous equation, which is done by solving the system of equations:

$$\frac{\partial \mathcal{L}}{\partial p_i} = 0, \frac{\partial \mathcal{L}}{\partial \lambda} = 0$$

This leads to the following optimal solutions:

$$p_1 = 0.5, p_2 = (1 - p_1) = 0.5 \Longleftrightarrow X \sim \mathcal{U}(\{1, 2\}) \text{ or } X \sim \mathcal{B}(0.5)$$

where $\mathcal{B}$ and $\mathcal{U}$ denote respectively the binomial and the uniform distributions.

The joint entropy of two variables X and Y is calculated by the same formula as that of Equation (5.3), where $p(x_i)$ is replaced by $p(x_i, y_j)$:

$$H(X, Y) = -\sum_{i=1}^{r_x}\sum_{j=1}^{r_y} p(x_i, y_j) \log p(x_i, y_j). \tag{5.5}$$

The chain rule property can be used to decompose the equation of the joint entropy as follows:

$$H(X_1, X_2, \ldots, X_n) = -\sum_{i=1}^{n} H(X_i|\mathbf{X}^{i-1})$$

where $\mathbf{X}^{i-1} = \{X_1, X_2, ..., X_{i-1}\}$ and $H(X_i|\mathbf{X}^{i-1})$ denotes the conditional entropy. The conditional entropy is used generally to represent the uncertainty over a random variable X remaining after the values of a set $\mathbf{Z}$ are known. Conditional entropy formula $H(X|\mathbf{Z})$ is given as follows:

$$H(X|\mathbf{Z}) = -\sum_{i=1}^{r_x}\sum_{j=1}^{r_{\mathbf{z}}} p(x_i, \mathbf{z}_j) \log p(x_i|\mathbf{z}_j). \tag{5.6}$$

By referring to the previous equation, the joint entropy expression given in Equation (5.5) can be rewritten as follows

$$H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X). \tag{5.7}$$

Note that $H(X|Y) \neq H(Y|X)$ except when X and Y are in a one-to-one mapping.

If we go back to the Subsection 4.1 of Chapter 4 and make the relation with the notion of conditional entropy, we can conclude that equivalence $X = f(\mathbf{Z}) \Longleftrightarrow H(X|\mathbf{Z}) = 0$ holds, because $\mathbf{Z}$ contains all the information about X (there is no uncertainty over X once the values of $\mathbf{Z}$ are known). The previous equivalence can be of great explanatory importance in our case, since it can be exploited in the identification of possible deterministic relations between a given set of variables, allowing therefore the identification of deterministic nodes $\mathbf{W}$ in $\mathcal{G}$ as follows:

$$\mathbf{W} = \{X_i \in \mathcal{G} | H(X_i|\mathbf{Pa}(X_i)) = 0 : \mathbf{Pa}(X_i) \neq \varnothing, i \in \{1, ..., n\}\}. \tag{5.8}$$

Note that deterministic root nodes $\mathbf{S}$ are not considered in our scope. Actually, $H(X_i) = 0$ means that X takes only one value in the dataset, hence X_i is not a random variable and can therefore be discarded from dataset $\mathcal{D}$. Such nodes are automatically independent from the rest of model variables:

$$\forall S \in \mathbf{S}, (\{S\} \perp_P (\mathcal{X} \setminus \{S\})).$$

To summarize, the relationship between deterministic nodes and conditional entropy is a valuable information that can be exploited when learning the BN structure. However, it may be pointed out that the exclusive use of conditional entropy to address the problem of *unfaithfulness*, as we shall see hereafter, is not sufficient because it may lead to another kind of problem called "statistical indistinguishability" [Luo06]. For this reason, the information given by conditional entropy must be merged with BN properties in the form of novel learning rules dedicated to deterministic nodes and to the learning process in general (in both adjacency and orientation steps).

5.2.3 First phase: learning the BN's skeleton

In the first phase of our approach, we try to determine the skeleton of the final BN from the available data in order to restrict the search space when moving to the second phase. Recall that the skeleton of a BN corresponds to its graphical structure $\mathcal{G}$ in which the arcs have been transformed into (undirected) edges. As a consequence, there exists an arc $X \rightarrow Y$ in the BN if and only if there exists an edge $X - Y$ in the skeleton.

Learning the skeleton prior to refine it into a BN is computationally attractive because the skeleton's undirected nature makes it faster to learn than learning BNs.

As discussed in Chapter 3, there exist several algorithms for discovering the skeleton of a BN, for instance the PC algorithm [SGS01], Max-Min Parents Children [TBA06] or Three-Phase Dependency Analysis [CGK$^+$02]. For the first step of our method, we designed a variant of PC adapted to exploit deterministic relations. We choose PC because it is simple and efficient. In addition, the way by which statistical independence tests are performed within the different iterations of the PC algorithm is well suited for the deterministic nodes identification during the skeleton learning phase.

To test whether X and Y are independent given a set of variable $\mathbf{Z}$ ($X \perp_P Y|\mathbf{Z}$), we have chosen to compute a G^2-statistical test, i.e.,

$$G^2(X,Y|\mathbf{Z}) = 2\sum_{i=1}^{r_x}\sum_{j=1}^{r_y}\sum_{k=1}^{r_{\mathbf{z}}} N_{i,j,k} \ln \frac{N_{i,j,k}N_k}{N_{i,k}N_{j,k}}, \tag{5.9}$$

where $N_{i,j,k}, N_{i,k}, N_{j,k}, N_k$ represent the number of occurrences of each tuple $(x_i, y_j, \mathbf{z}_k)$, $(x_i, \mathbf{z}_k)$, $(y_j, \mathbf{z}_k)$, $\mathbf{z}_k$ in an N-sized dataset respectively. The G^2 test is known to provide more robust independence tests than χ^2, especially for large datasets. Moreover, the likelihood ratio G^2 computation proves to be more tractable for large dimensional datasets than χ^2, since it can be neatly decomposed into smaller components (by parsing database), allowing therefore to speed-up its computation. This option cannot be done exactly with χ^2 statistical test.

As discussed earlier, the idea of PC consists in starting with a complete undirected graph $\mathcal{G}$. Then, the algorithm iterates the computations of independence tests between pairs of variables (X, Y) given sets $\mathbf{Z}$ that are adjacent to X or Y in $\mathcal{G}$. Whenever it finds that $\{X\} \perp_P \{Y\}|\mathbf{Z}$, edge $X - Y$ is removed from $\mathcal{G}$ and $\mathbf{Z}$ is added to the $SepSet_{XY}$. However, as shown in the preceding section, when $X = f(\mathbf{Z})$, the G^2-test always indicates that $\{X\} \perp_P \{Y\}|\mathbf{Z}$, for any Y, even if Y strongly depends on X (e.g., Y is a child of X in the searched DAG). See Figure 4.5. Therefore, a G^2-test should never be used when X or Y is a deterministic node depending on $\mathbf{Z}$, since it leads automatically[4] to an erroneous result (error of type II). Unfortunately, deterministic nodes are not known from the beginning of the learning algorithm. As statistical tests cannot distinguish between conditional independences resulting from *d-separation* and those entailed by deterministic relations, this calls for a method to detect deterministic nodes before computing the statistical tests. For this purpose, we exploit the conditional entropy $H(X|\mathbf{Z})$, and more precisely the equivalence property between the latter and the deterministic relation as discussed earlier. As we shall see, this will be sufficient to avoid numerous erroneous edge deletions.

[4]Except in the case of statistical indistinguishability as we shall see later.

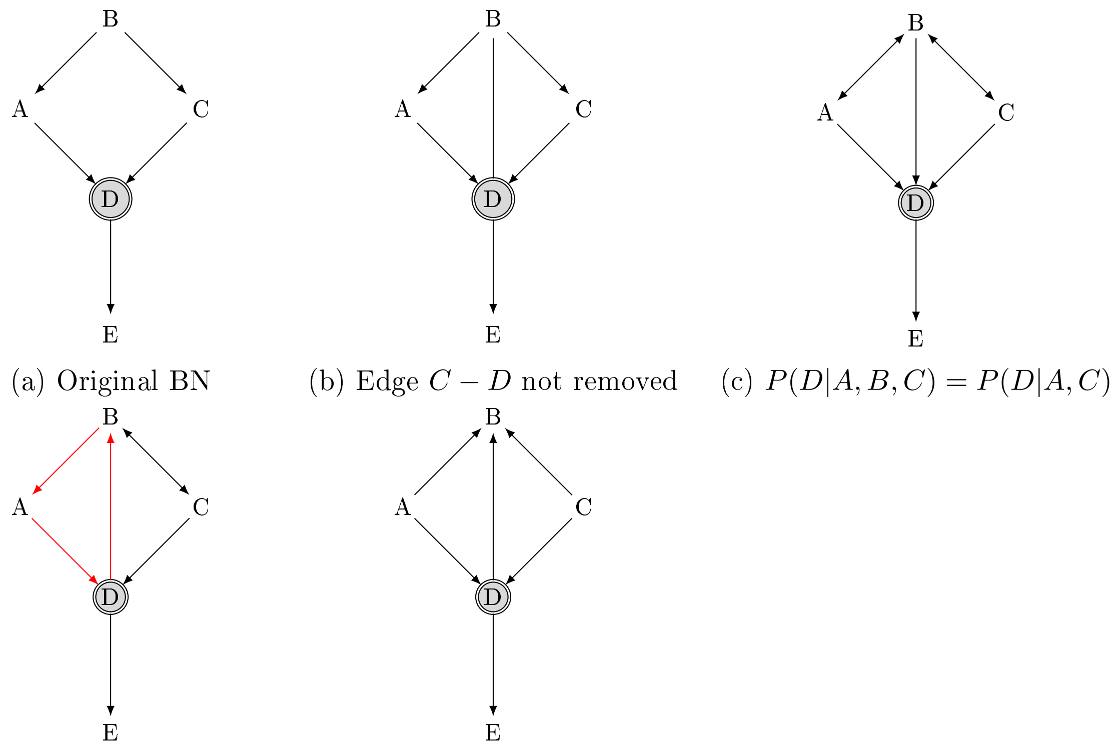
Given a dataset $\mathcal{D}$, the conditional entropy of a variable X given $\mathbf{Z}$ (represented by $\mathbf{Pa}(X)$ in $\mathcal{G}$) can be estimated by the following equation:

$$H(X|\mathbf{Z}) = -\frac{1}{N}\sum_{i=1}^{r_X}\sum_{j=1}^{r_\mathbf{Z}} N_{i,j}\log\frac{N_{i,j}}{N_j} \tag{5.10}$$

where N_{ij} is the number of occurrences in the dataset in which X and $\mathbf{Z}$ have taken their ith and jth values respectively, N_j is the number of occurrences in the dataset in which $\mathbf{Z}$ has taken its jth value, and r_X and $r_\mathbf{Z}$ are the domain sizes of X and $\mathbf{Z}$ respectively.

As shown above, X is a deterministic node defined by $X = f(\mathbf{Z})$ if and only if $H(X|\mathbf{Z}) = 0$. Therefore our first adaptation of PC consists in computing the conditional entropies $H(X|\mathbf{Z})$ and $H(Y|\mathbf{Z})$ and, if one of them is equal to 0 (or is below a threshold), we just do not use the G^2-test over $(X,Y|\mathbf{Z})$ to remove edge $X - Y$.

For instance, in the BN depicted by Figure 5.5.(a), the test $\{D\} \perp_P \{E\}|\{A,C\}$ should not be carried out by the PC algorithm, since $D = f(A,C)$. It must be noted here that computing these conditional entropies only marginally increases the overall learning's computation time. Actually, the most time consuming task is the parsing of the dataset in order to evaluate observation counts $N_{i,j,k}, N_{i,k}, N_{j,k}, N_k$, which are also needed to compute traditional scores like BDeu, BIC or K2. The computations of Equations (5.9) and (5.10) are significantly faster than this task.



(a) Original BN (b) Edge $C - D$ not removed (c) $P(D|A,B,C) = P(D|A,C)$

(d) problem of acyclicity (e) $P(B|A,C,D) = P(B|A,C)$

Figure 5.5: A BN with both deterministic relation $D = f(A,C)$ and the problem of statistical indistinguishability.

5.2.3.1 Edges deletion with deterministic nodes

As discussed earlier, the use of conditional entropy test allows to overcome the problem of *unfaithfulness* by detecting the set of deterministic nodes during the BN learning task. However, the exclusive use of conditional entropy when performing conditional independence tests can raise another kind of problem which is called "statistical indistinguishability" [Luo06]: there exist cases where conditional independences can be explained by both *d-separation* and by deterministic relations. For instance, in Figure 5.5, the conditional independence between B and D given $\{A, C\}$ can result from B and D being *d-separated* by $\{A, C\}$ but also from D being a deterministic node defined by $D = f(A, C)$. By taking into account the results returned by conditional entropy, our algorithm first checks whether $H(D|\{A, C\}) = 0$ or $H(B|\{A, C\}) = 0$. As this is the case for node D since $D = f(A, C)$, edge $B - D$ will never be removed from $\mathcal{G}$ due to a G^2 test involving D given $\{A, C\}$ (since the rule we provided in the preceding subsection forbids G^2 testing in this case). But it might be the case that the "true" BN we look for contains neither arc $B \to D$ nor $D \to B$ due to *d-separation*, hence edge $B - D$ should have been removed. Note that the statistical indistinguishability problem can be a serious issue if it is not handled during BN learning, especially for large networks involving many deterministic nodes since:

- It prevents the algorithm from deleting many false positive edges (error of type I) from $\mathcal{G}$, hence leading to the learning of a complex BN structure (with a lot of arcs).
- It can also be the cause of false negative edge deletions (error of type II especially for small dataset), since it contributes to the increase of the adjacency set around deterministic variables in $\mathcal{G}$, leading therefore to the increase of the sizes of the contingency tables used for the conditional independence tests[5].

If we come back to the example of Figure 5.5, the approach of [Luo06] and the current version of our approach keep this edge since there is no rule yet allowing to remove it from the skeleton.

However, as will be shown in Proposition 5.1, edge $B - D$ can always be safely removed from $\mathcal{G}$ without requiring the computation of G^2 independence test, i.e., the BN should not contain arcs $B \to D$ or $D \to B$.

Proposition 5.1. *Let X be a deterministic node defined by $X = f(\mathbf{Z})$ and let Y be a node such that, in the skeleton $\mathcal{G}$ learnt by the algorithm, $Y \notin \mathbf{Z}$ and $\mathbf{Z} \subseteq \mathbf{Adj}(Y)$, where $\mathbf{Adj}(Y)$ refers to the set of neighbors of Y in $\mathcal{G}$. Then edge $X - Y$ can be removed from $\mathcal{G}$.*

[5]We should keep in mind that conditional independence tests rely on the adjacency of each tested nodes as a set of conditioning variables

Proof. Our goal is to learn a skeleton $\mathcal{G}$. Hence to any of its edges must correspond an arc in the BN. Three cases can obtain:

- assume that, in the BN, there exists an arc $X \to Y$ and that all the nodes, say Z, of $\mathbf{Z}$ are such that $Z \to Y$. Then $\{X\} \cup \mathbf{Z} \subseteq \mathbf{Pa}(Y)$. Define $\mathbf{K}$ as $\mathbf{Pa}(Y) \setminus (\{X\} \cup \mathbf{Z})$. Then $P(Y|\mathbf{Pa}(Y)) = P(Y|\{X\} \cup \mathbf{Z} \cup \mathbf{K}) = P(Y|\mathbf{Z} \cup \mathbf{K})$ because X does not bring any additional information once $\mathbf{Z}$ is known. Therefore arc $X \to Y$ can be removed from the BN as well as edge $X - Y$ from $\mathcal{G}$.
- assume that, in the BN, there exists an arc $X \to Y$ and that there exists at least one node $Z \in \mathbf{Z}$ such that $Y \to Z$. Then X, Y, Z forms a directed cycle since $Z \in \mathbf{Z}$ is also a parent of X as $X = f(\mathbf{Z})$. This is impossible since BNs are DAGs.
- assume that, in the BN, there exists an arc $Y \to X$. Then $\{Y\} \cup \mathbf{Z} \subseteq \mathbf{Pa}(X)$. Define $\mathbf{K}$ as $\mathbf{Pa}(X) \setminus (\{Y\} \cup \mathbf{Z})$. Then $P(X|\mathbf{Pa}(X)) = P(X|\{Y\} \cup \mathbf{Z} \cup \mathbf{K}) = P(X|\mathbf{Z})$ because, once $\mathbf{Z}$ is known, X is determined. Hence arc $Y \to X$ can be removed from the BN as well as edge $X - Y$ from $\mathcal{G}$.

So, in all cases, the BN should contain neither arc $B \to D$ nor arc $D \to B$. The skeleton should therefore not contain edge $B - D$. □

To facilitate the understanding of Proposition 5.1, a schematization example of the different parts (or the orientation cases) of the previous proof are shown by Figure 5.5.(b).(c).(d).(e).

Through the above proposition, we can see that the originality of our proposal is to solve the problem of statistical indistinguishability by using both conditional entropy (to discover deterministic nodes) and by exploiting properties of Bayesian networks (conditional independences, acyclicity, *etc.*).

Note that the way by which Proposition 5.1 is used during the skeleton learning phase will be as follows: at the end of the first phase of our algorithm, for each deterministic node X, all edges $X - Y$ for nodes Y satisfying the conditions of Proposition 5.1 are discarded. Remind that the previous proposition is applied only once at the end of our first skeleton phase. This can be explained by the fact that PC algorithm starts with a fully connected graph $\mathcal{G}$, hence many edges may satisfy the deletion conditions mentioned in Proposition 5.1 at the beginning steps of the algorithm, leading therefore to many erroneous edge deletions from $\mathcal{G}$. However, at the end of the skeleton's construction, this proposition is meaningful and is coherent with the proof of Proposition 5.1.

A second variant we introduce also concerns edges deletions **during the skeleton learning** using deterministic variables: the proposition below shows that some edges can be removed from $\mathcal{G}$ without requiring the computation of conditional independences using G^2 test:

Proposition 5.2. *Let X and Y be two deterministic nodes, i.e., $X = f(\mathbf{W})$ and $Y = f(\mathbf{Z})$. Assume that $X \notin \mathbf{Z}$ and that $Y \notin \mathbf{W}$. Then edge $X - Y$ can be safely removed from $\mathcal{G}$.*

Proof. Let us assume that the BN we learn contains arc $X \to Y$. Then, $\{X\} \cup \mathbf{Z} \subseteq \mathbf{Pa}(Y)$. But then, $P(Y|\mathbf{Pa}(Y)) = P(Y|\mathbf{Z})$ since $Y = f(\mathbf{Z})$. Therefore, removing from the BN all the arcs in $\mathbf{Pa}(Y)\backslash\mathbf{Z}$ results in a new BN that represents the same distribution. Therefore, it is safe to discard arc $X \to Y$. By symmetry, it would also be safe to discard arc $Y \to X$. $\mathcal{G}$ being a skeleton, it is thus safe to remove edge $X - Y$ from $\mathcal{G}$. $\square$

An example of edge deletion using Proposition 5.2 is mentioned in the network example of Figure 5.6. As can be seen, in this example the BN contains two deterministic nodes $Z = f(X, Y)$ and $W = f(S)$. Knowing that each edge in the skeleton corresponds to an arc in the final BN structure, the two possible orientation cases of edge $Z - W$ in $\mathcal{G}$ lead to the following results:

- $Z \to W$:

$$\begin{aligned} \text{if } f(s) = w \quad &\Longrightarrow P(W = w|S = s, Z = z) \approx \frac{N_{w,s,z}}{N_{s,z}} = \frac{N_{w,s}}{N_s} = 1 \\ &\Longrightarrow P(W|S, Z) = P(W|S) \end{aligned}$$

- $Z \leftarrow W$:

$$\begin{aligned} \text{if } f(x, y) = z \quad &\Longrightarrow P(Z = z|X = x, Y = y, W = w) = \frac{N_{z,x,y,z}}{N_{x,y}} = \frac{N_{x,y,z}}{N_{x,y}} = 1 \\ &\Longrightarrow P(Z|X, Y, W) = P(Z|X, Y) \end{aligned}$$

As a result, we can clearly see that both edge $Z - W$ orientations do not entail any modification on both conditional probability distributions of Z and W when W and Z are respectively instantiated. The only impact that edge $Z - W$ may exerts on the BN representation concerns the sizes of the CPTs of Z and W, the latter being obtained by simply duplicating the CPTs of the deterministic nodes $P(Z|X, Y)$ and $P(W|S)$ by the number of modalities taken by W and Z respectively. Due to these reasons, edge $Z - W$ must be deleted from $\mathcal{G}$ without being tested.

To summarize, each time our algorithm discovers a new deterministic node, it applies Proposition 5.2 to remove all the edges between this node and the previously discovered deterministic nodes satisfying the conditions.

Overall, our algorithm for computing the BN's skeleton (using the previous propositions) when there exist deterministic nodes is described in Algorithm 8.

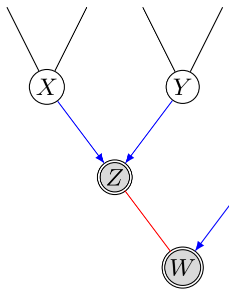


FIGURE 5.6: Example of edge $Z - W$ deletion using deterministic nodes $Z = f(X, Y)$ and $W = f(S)$.

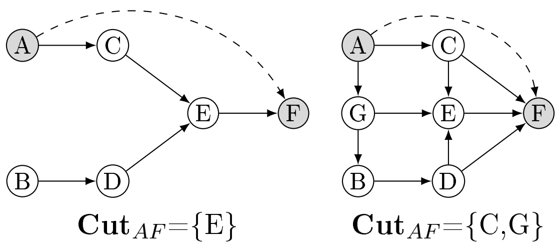


FIGURE 5.7: Example of two scenarios of conditioning variables computations.

5.2.3.2 Reducing the sizes of the conditioning sets during G^2 tests

Another variant with the original PC algorithm that was already advocated in [SGS01] and [AGM06] is the use of min cutsets to determine the conditioning sets in our independence tests in order to avoid missing edges (error of type II): PC usually tests the independence between X and Y first by computing $G^2(X, Y|\emptyset)$ then, if this test fails (i.e., if it does not provide evidence that X is independent of Y), conditional tests $G^2(X, Y|\mathbf{Z})$ are performed iteratively with sets $\mathbf{Z}$, adjacent to X or Y, of increasing sizes until either a conditional independence is proved or the size of the database does not allow any more meaningful statistical test.

In our algorithm, we not only perform these tests with $\mathbf{Z}$ of increasing sizes, we also compute a minimal cutset $\mathbf{Cut}_{XY}$ in $\mathcal{G}$ separating X and Y, i.e., a set of nodes $\mathbf{Cut}_{XY}$ which, when removed from $\mathcal{G}$, makes X and Y belong to two different connected components. This can be easily done by computing a min cut in a max-flow problem. If the size of $\mathbf{Cut}_{XY}$ is smaller than that of sets $\mathbf{Z}$ PC would have used, we just perform $G^2(X, Y|\mathbf{Cut}_{XY})$. This results in the construction of smaller contingency tables and, as such, this increases the accuracy of the statistical test. Moreover, the additional time required to compute $\mathbf{Cut}_{XY}$ can be balanced by the number of G^2-tests it allows to avoid. An example of conditioning variables selection can be shown in Figure 5.7. In this example, we assume that we wish to test the conditional independence between A and F. The application of a max-flow algorithm (to construct the min-cut) on the left graph returns a conditioning set equal to $\{E\}$, which also corresponds to what the PC algorithm would have chosen. On the right graph, however, PC would be unable to find a conditioning set of size 2 in the neighborhood of F that would make A and F independent whereas conditioning on $\mathbf{Cut}_{AF} = \{C, G\}$ would conclude that edge $A - F$ can be removed from the skeleton.

Of course, for large graphs, computing min cutsets can be expensive. In this case, we resort to a technique advocated in [GJ06]: basically, graph $\mathcal{G}$ is incrementally triangulated, hence resulting in a junction tree incrementally constructed [GJ06]. Then, some cliques $\mathbf{C}_X$ and $\mathbf{C}_Y$ containing X and Y are identified as well as the minimal-size separator $\mathbf{S}_{XY}$ on the trail between $\mathbf{C}_X$ and $\mathbf{C}_Y$. Set $\mathbf{S}_{XY}$ is a cutset and, thus, to test the independence between X and Y, it is sufficient to compute $G^2(X, Y|\mathbf{S}_{XY})$. In practice, incremental triangulations are very fast to perform and sets $\mathbf{S}_{XY}$ are relatively small.

Algorithm 8: Learning the skeleton of the BN

```
Input: a dataset $\mathcal{D}$
Output: a skeleton $\mathcal{G}$
Start with a complete undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{A})$
DR $\leftarrow \emptyset$   // the set of deterministic relations found so far
SepSet $\leftarrow \emptyset$   // the set of independences $\{X\} \perp_P \{Y\}|\mathbf{Z}$ found so far
$i \leftarrow 0$   // the size of the conditioning sets tested
repeat
    foreach edge $X - Y$ in $\mathcal{G}$ do
        if there exists a set $\mathbf{Z} = \mathbf{Cut}_{XY}$ or $\mathbf{S}_{XY}$ of size $\leq i$, or $\mathbf{Z}$ adjacent to $X$ or
          to $Y$ of size $i$, s.t. $\{X\} \perp_P \{Y\}|\mathbf{Z}$ then
            if $\nexists$ $\mathbf{S}, \mathbf{T} \subseteq \mathbf{Z}$ s.t. $(X|\mathbf{S}) \in DR$ or $(Y|\mathbf{T}) \in DR$ then
                NewDeterministicRelation $\leftarrow$ false
                if $H(X|\mathbf{Z}) = 0$ then
                    find the smallest $\mathbf{S} \subseteq \mathbf{Z}$ s.t. $H(X|\mathbf{S}) = 0$
                    DR $\leftarrow$ DR $\cup \{(X|\mathbf{S})\}$
                    apply Proposition 5.2 to remove edges
                    NewDeterministicRelation $\leftarrow$ true
                if $H(Y|\mathbf{Z}) = 0$ then
                    find the smallest $\mathbf{T} \subseteq \mathbf{Z}$ s.t. $H(Y|\mathbf{T}) = 0$
                    DR $\leftarrow$ DR $\cup \{(Y|\mathbf{T})\}$
                    apply Proposition 5.2 to remove edges
                    NewDeterministicRelation $\leftarrow$ true
                if NewDeterministicRelation = false then
                    $\mathbf{A} \leftarrow \mathbf{A} \setminus \{X - Y\}$
                    SepSet $\leftarrow$ SepSet $\cup \{(X, Y|\mathbf{Z})\}$
until all nodes in $\mathbf{V}$ have at most $i$ neighbors;
use Proposition 5.1 to remove unnecessary edges
return undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{A})$
```

5.2.4 Second phase: orientation and refinement

The next step of our learning algorithm consists in orienting the edges in order to get an initial BN that will subsequently be refined. In [SGS01], PC first identifies the BN's v-structures using the computed "*SepSet*", i.e., triples of nodes (X, Y, Z) that are connected as $X - Y - Z$ and such that $\{X\} \not\perp_P \{Z\}|\mathbf{Y}$ and $\{X\} \perp_P \{Z\}|\mathbf{W}$ for some $\mathbf{W} \not\supseteq \{Y\}$. For these triples, the BN should contain the following arcs: $X \rightarrow Y \leftarrow Z$. A BN is fully characterized by its skeleton and the set of its *v-structures*. When deterministic nodes exist, it is suggested in [Luo06] to add an extra condition on sets $\mathbf{W}$: if X or Y is a deterministic node defined by some set $\mathbf{S}$, then $\mathbf{W} \not\supseteq \mathbf{S}$. Our algorithm follows the same rule. In our framework, we add, prior to the v-structures orientation, the following "causal" constraint: for each deterministic node $X = f(\mathbf{W})$, all the edges $W - X$, for $W \in \mathbf{W}$, are converted into arcs $W \rightarrow X$. But deterministic relations can be further exploited; examples of such exploitation can be found, e.g., in [JHS10] and [DJM+10]. Here, we propose to leverage deterministic relation features to deduce some orientations that are necessary. As an illustration, consider skeleton $\mathcal{G}$ of Figure 5.8.a where $X = f(U, Y, Z)$ is a deterministic node. Our "causal" constraint imposes the orientation of Figure 5.8.b. But, as shown in the proposition below, the only reasonable orientation for the remaining edges adjacent to X are those given in Figure 5.8.c.

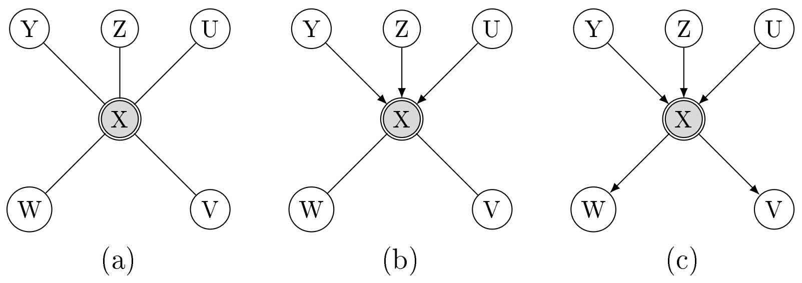


FIGURE 5.8: A skeleton $\mathcal{G}$ with a deterministic node $X = f(Y, Z, U)$

Proposition 5.3. *Let $\mathcal{G}$ be a skeleton and let X be a deterministic node defined by $X = f(\mathbf{W})$. Let $\mathbf{Z} = \mathbf{Adj}(X) \setminus \mathbf{W}$ in $\mathcal{G}$. Then for each $Z \in \mathbf{Z}$, edge $X - Z$ must be oriented as $X \rightarrow Z$.*

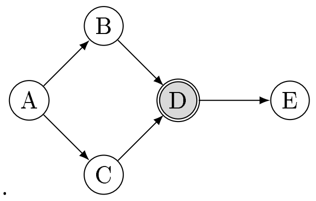
Proof. Assume that the BN contains arc $Z \rightarrow X$. In the BN, let $\mathbf{K} = \mathbf{Pa}(X) \setminus (\mathbf{W} \cup \{Z\})$. Then the joint probability model by the BN is $P(\mathbf{V}) = P(X|\mathbf{Pa}(X)) \times \prod_{V \in \mathbf{V} \setminus \{X\}} P(V|\mathbf{Pa}(V)) = P(X|\mathbf{W}) \times \prod_{V \in \mathbf{V} \setminus \{X\}} P(V|\mathbf{Pa}(V))$ because, once $\mathbf{W}$ is known, $\mathbf{K}$ and Z do not bring any additional information on X. Therefore arc $Z \rightarrow X$ can be removed from the BN without altering the probability distribution. But this is impossible because edge $X - Z$ belonging to skeleton $\mathcal{G}$ implies that X and Z are conditionally dependent given any other set of nodes. Hence, the arc between X and Z in the BN is necessarily $X \rightarrow Z$. $\square$

An example of orientation using Proposition 5.3 is mentioned in Figure 5.9.(a). Remind that there exists an arc $D \leftarrow E$ or $E \rightarrow D$ in the BN structure if and only if there exists an edge $D - E$ in its skeleton. As a matter of fact, the edge linking D and E in the example of Figure 5.9 has to be oriented at the end of the learning process. By assuming that all variables represented in the BN are binary and the deterministic relation $D = f(B, C)$ is defined as follows:

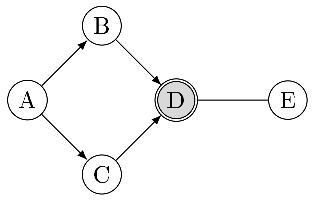
$$f(b,c) = \begin{cases} f & \text{if} \quad (b = f) \wedge (c = f), \\ t & otherwise \end{cases}$$

The resulting CPTs of node D given the two possible orientations of $D - E$ are shown in Figure 5.10.

As can be seen in Figure 5.10, we notice that the CPT representing the distribution $P(D|B, C, E)$ appears as a duplication ($\times$ 2 times) of $P(D|B, C)$ (which corresponds to the deterministic relation $D = f(B, C)$). Given the previous assertion, we can deduce that the semantic of the CPT of node D where $D \leftarrow E$ is inconsistent with that of the edge $D - E$ learnt in the CPDAG as shown in Figure 5.9.(b): remind that the $D - E$ asserts that $D \not\perp_P E|\mathbf{W}$ in $\mathcal{P}$ for all $\mathbf{W} \subseteq \mathcal{X} \setminus \{B, D, E\}$ i.e., learning new information on a given variable influences our belief in the other. Given the previous property,



(a) Original BN structure with $D = f(B, C)$



(b) CPDAG of BN

FIGURE 5.9: Example of BN and CPDAG to be learned

		$P(D\|B,C)$	
		D	
B	C	t	f
f	f	0	1
f	t	1	0
t	f	1	0
t	t	1	0

(a) CPT with $D \rightarrow E$

			$P(D\|E,B,C)$	
			D	
E	B	C	t	f
f	f	f	0	1
f	f	t	1	0
f	t	f	1	0
f	t	t	1	0
t	f	f	0	1
t	f	t	1	0
t	t	f	1	0
t	t	t	1	0

(b) CPT with $D \leftarrow E$

FIGURE 5.10: Resulting CPT orientation of edge $D - E$

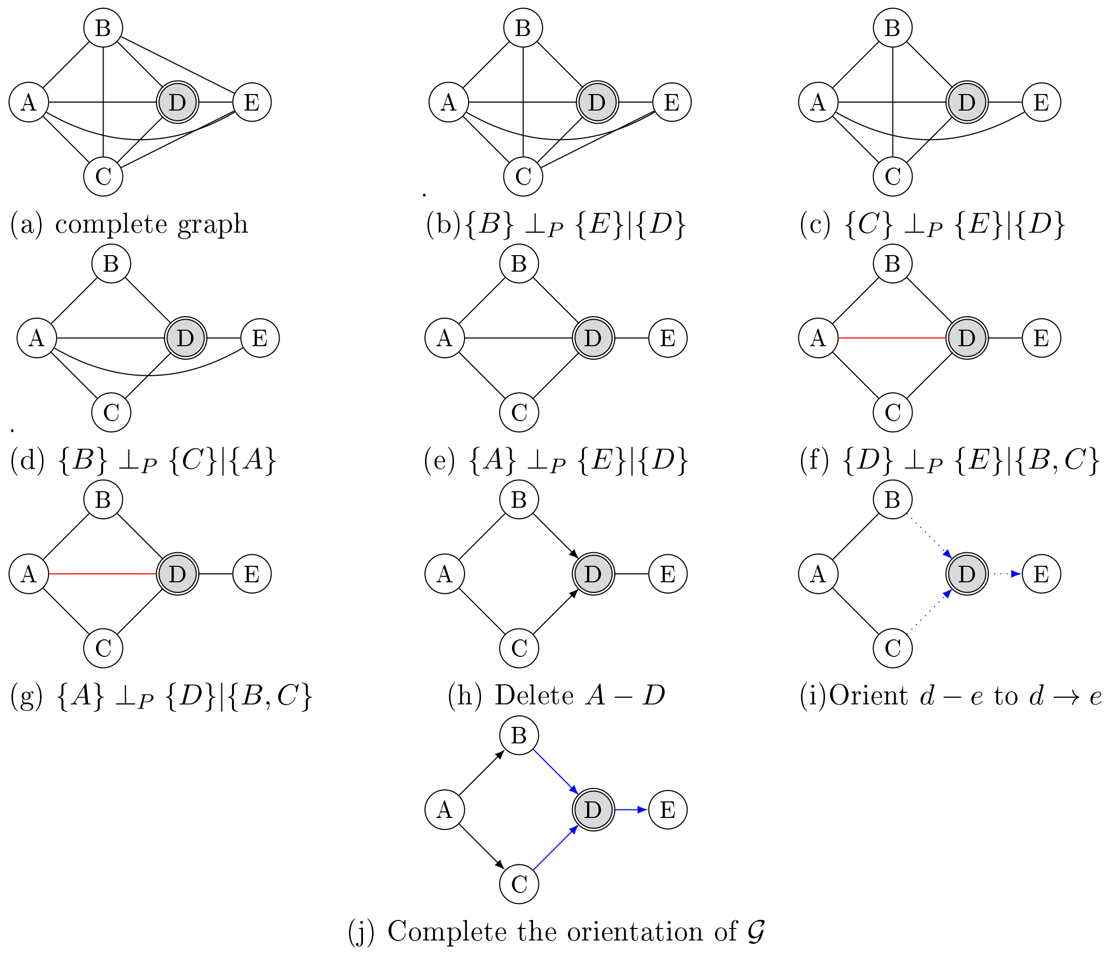
the orientation $D \leftarrow E$ (which gives $P(D|B,C,E) = P(D|B,C)$) must be replaced by $D \rightarrow E$ in $\mathcal{G}$ in order to represent the dependency between D and E.

With all those rules, our algorithm converts the skeleton into a Completed Partially Directed Acyclic Graph (CPDAG). The remaining edges are then converted in a similar fashion as PC does (when PC is unable to orient edges, we select arbitrarily an orientation). At this point, our algorithm has constructed an initial BN. This one is then refined using any search algorithm [TBA06]. The only constraint we add on this algorithm is that it never modifies the arcs adjacent to the deterministic nodes nor add new ones. As a matter of fact, due to the *unfaithfulness* induced by determinism, classical algorithms tend to erroneously modify the neighborhood of the deterministic nodes.

To summarize the different steps of our learning algorithm, we take the example of data that has been sampled from the model of Figure 5.9.(a). The results of the different phases while executing our proposed algorithm are depicted in Figure 5.11. As can be seen, the algorithm starts with a completely connected undirected graph (a). At first, the checking phase of the set of marginal independences between variables ends without any edge removed since the model of Figure 5.9 does not entail any marginal independence. In the second step, the conditional independence of each pair of adjacent nodes in $\mathcal{G}$ given one single conditioning variable is tested. The removals of edges $B-E$ and $C-E$ result from such conditional independences given $\{D\}$ (see figures 5.11.(b) and 5.11.(c)). Then, edge $B-C$ can be removed by testing $\{B\} \perp_P \{C\}|\{A\}$ (Figure 5.11.(d)). At this point, the min-cutset algorithm can identify that independence tests between A and E can be performed by conditioning only on $\mathbf{Cut}_{AE} = \{D\}$ (Figure 5.11.(e)). Recall that the goal of the min cutset algorithm is to minimize the set of conditioning variables, this is the reason why D will be used instead of, e.g., $\{B,C\}$ (which would have been selected by PC). In the same step, we can remark in (f) and (g) that, despite the independences $\{D\} \perp_P \{E\}|\{B,C\}$ and $\{D\} \perp_P \{A\}|\{B,C\}$ resulting from the statistical tests, the edges $D-E$ and $A-D$ are not removed from $\mathcal{G}$ since $H(D|\{B,C\}) = 0$ and the deterministic nature of $D = f(B,C)$ is recorded[6]. Given the resulting skeletons in (f) and (g), we can clearly identify that edge $A-B$ satisfies the set of deletion conditions defined in Proposition 5.1[7] since D is identified as a deterministic node in $\mathcal{G}$ and the adjacent nodes of A are also the determinant variables of D. As a matter of fact, the edge $A-B$ should never exists in $\mathcal{G}$ (it is thus removed).

During the skeleton's orientation phase, the deterministic relation $D = f(B,C)$ leads to the orientation of edges $B-D$ and $C-D$ towards D. In addition, the deterministic nature of node D is also exploited to orient the edge $D-E$ as $D \rightarrow E$ by referring to

[6]The deterministic node D and the set of *SepSet* together are used to orient the skeleton of the BN.
[7]Remind that this edge cannot be removed from $\mathcal{G}$ due to the problem of statistical indistinguishability.



(a) complete graph
(b)$\{B\} \perp_P \{E\}|\{D\}$
(c) $\{C\} \perp_P \{E\}|\{D\}$
(d) $\{B\} \perp_P \{C\}|\{A\}$
(e) $\{A\} \perp_P \{E\}|\{D\}$
(f) $\{D\} \perp_P \{E\}|\{B, C\}$
(g) $\{A\} \perp_P \{D\}|\{B, C\}$
(h) Delete $A - D$
(i)Orient $d - e$ to $d \to e$
(j) Complete the orientation of $\mathcal{G}$

FIGURE 5.11: Execution of our algorithm when learning the model of Figure 5.9.(a)

our proposed rule shown in Proposition 5.3[8]. Note that at step (i), information about *SepSet* and meek's rules cannot allow the orientation of the rest of the CPDAG, notably the edges $B - A$ and $C - A$ given that they can be oriented in both directions. In such a case, we select an arbitrary orientation of the remaining edges and we obtain a prior DAG that will be the starting point of a refined process using the *TABU* search algorithm. Recall that during this search phase, the adjacency of node D ($B \to D$, $C \to D$ and $D \to E$) should never been modified[9].

5.3 Conclusion

In this chapter, we have proposed a new algorithm for learning the structure of BNs when some variables have deterministic relations with others. This algorithm relies on very effective dedicated rules whose mathematical correctness has been proved. Unlike popular BN learning algorithms dedicated to deterministic relations, ours does not rely

[8]In addition to Proposition 5.3, an opposite orientation of $D \to E$ leads to a *v-structure* which is inconsistent with the independences shown in steps (b,c) of Figure 5.11.

[9]Note that we cannot rely on score and statistical tests when the *faithfulness* assumption is ruled out.

on the exclusive use of statistical tests to address the problem of erroneous edge deletions. Its originality lies in the combination of valuable information coming from deterministic nodes with other useful properties entailed by BN in an efficient way. The principle of our approach can be described as follows:

i) it consists in learning at a first time the skeleton of the BN using an extension of the PC algorithm (cutset and entropy) together with our first proposed rule which attempts to discard arcs that should not belong to the BN;

ii) statistical indistinguishability that may result from the use of the entropy function during the previous step is handled by means of a second proposed rule which defines a set of conditions (available only at the end of the first phase) under which false positive edges in $\mathcal{G}$ are detected and then removed;

iii) finally, our algorithm uses an original way to orient the skeleton of the BN by exploiting first the discovered deterministic nodes and *SepSet* variables to obtain a prior orientation. Second, it consists in refining the prior orientation through a score-based search algorithm under a set of rules (or constraints) deduced from the deterministic nodes.

To summarize, thanks to our proposed approach we can theoretically handle the problem of *unfaithfulness* when learning the BN from data containing deterministic nodes. In the next chapter, we provide experiments to show the effectiveness and efficiency of our method.

Chapter 6

Evaluation on Benchmark Bayesian networks

6.1 Principles of the benchmark evaluation

In this chapter, we highlight the effectiveness of our method by comparing it with MMHC [TBA06] (which is not suited to cope with deterministic nodes), "OR+Inter.IAMB" [RdMAC08] and the algorithm of Luo [Luo06], substituting its association rule miners by conditional entropy tests to detect deterministic nodes, in order to improve its effectiveness.

The comparisons of the studied algorithms have been carried out by evaluating the structure returned by each of these algorithms. To do this, we used synthetic datasets generated from discrete BNs following the guidelines given in [ICR04]. We thus generated randomly BNs containing from 10 to 50 nodes and from 12 to 170 arcs. Nodes had at most 6 parents and their domain size was randomly set between 2 and 6. Finally, the number of deterministic nodes was chosen arbitrarily between 1 and 15. From these BNs, we generated datasets of sizes ranging from 1000 to 50000 records. Overall, 750 datasets were generated. It should be emphasized that each BN produced by MMHC, OR+Inter.IAMB, Luo and our algorithm on these datasets has been converted into a CPDAG corresponding to its Markov equivalence class (i.e., a skeleton in which the v-structures are oriented). This makes comparisons meaningful since two BNs represent exactly the same independence model if and only if they belong to the same Markov equivalence class, i.e., they have the same CPDAGs. Finally, the CPDAGs were compared against those of the true BNs by means of three metrics:

1. the "Precision" (or Positive Predictive value), Precision=$TP/(TP + FP)$, where TP denotes the number of arcs/edges present in the learnt CPDAG that also exist

in the true BN's CPDAG (True positive) and FP corresponds to the number of arcs/edges present in the learnt CPDAG that do not exist in the true BN's CPDAG (False Positive).

2. The "Recall" (True Positive rate), Recall=$TP/(TP+FN)$, where FN corresponds to the number of arcs/edges deleted erroneously during the learning process (False Negative).

3. The third criterion is the "F-score", which can be considered as a more global evaluation metric comparing to the Recall and the Precision, since it allows to quantify how good is the learning algorithm in both dependences and independences discovering. As such, it allows to give a general overview on the efficiency of each studied algorithm. Its equation is given as follows:

$$\text{F-score} = 2 \times (\text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})^{1}.$$

All the details related to the computations of the mentioned evaluation metrics given the learnt and the original CPDAGs are shown in Table 6.1. For compactness, Precision and Recall are denoted respectively by Prec. and Rec.

The purpose of this experimental chapter of our work is to reinforce and prove the reliability of the theoretical properties of our algorithm (particularly those that deal with deterministic nodes) and to highlight its improvement w.r.t. state-of-the-art algorithms.

6.2 Using simulated datasets from synthetic Bayesian networks

6.2.1 The evaluation of the whole network structure

In this section we perform the evaluation of the whole CPDAG returned by each of the aforementioned learning algorithms[MGJCC14, TBA06, RdMAC08, Luo06]. Here, we do not restrict ourselves to the parts of the structure concerned by the unfaithfulness (deterministic nodes) but we compute the metrics of the above section on the whole graph.

For the experiments, many BN parameters have been varied in order to study their effect on the behavior of each of the studied algorithms. This allows fine-grain interpretations. The considered parameters and their ranges have been defined as follows:

- the complexity of $\mathcal{G}$: 12-30, 31-50, 51-70, 71-170 arcs.

[1] It can be considered as a tradeoff between Precision and Recall metrics.

$N°$	Original edges/arcs	Learnt edges/arcs	TP	FN	FP	TN
Undirected part of the CPDAG						
1	$-$	$-$	$\times$			
2	$-$	$\rightarrow$			$\times$	
3	$-$	$\leftarrow$			$\times$	
4	$-$	$\not\leftrightarrow$		$\times$		
5	$\not\leftrightarrow$	$\not\leftrightarrow$				$\times$
6	$\not\leftrightarrow$	$-$			$\times$	
7	$\not\leftrightarrow$	$\leftrightarrow$			$\times$	
Directed part of the CPDAG						
8	$\rightarrow$	$\rightarrow$	$\times$			
9	$\rightarrow$	$-$		$\times$		
10	$\rightarrow$	$\leftarrow$		$\times$		
11	$\leftrightarrow$	$\not\leftrightarrow$		$\times$		
12	$\leftarrow$	$\rightarrow$		$\times$		
13	$\leftarrow$	$-$		$\times$		
14	$\leftarrow$	$\leftarrow$	$\times$			
Recall=		$\sum_i TP_i / \sum_i (TP_i + FN_i)$				
Precision=		$\sum_i TP_i / \sum_i (TP_i + FP_i)$				
F-score=		2×(Recall× Precision)/(Recall+Precision)				

TABLE 6.1: Computing the distance between the learnt and the original BNs

- the number of deterministic nodes in $\mathcal{G}$: 1-4, 5-9 and 10-15.
- the size of the datasets used to learn $\mathcal{G}$: 1000, 5000, 10000, 20000, 50000.
- the number of parents per family in $\mathcal{G}$: 1-4, 5-7.

Note that in our context, the parameters related to the complexity and the number of parents in $\mathcal{G}$ have been considered together since they entail almost the same information. Figure 6.1 displays the averages over the generated datasets of the F-score, Recall and Precision metrics for the four algorithms. Curves being similar, we averaged the results for BNs with 1-4 deterministic nodes (the top two curves), 5-9 deterministic nodes (the two curves at the middle) and 10-15 deterministic nodes at the bottom of the figure.

As can be seen from the resulting curves, except sometimes when the dataset is small (size equal to 1000 records), our algorithm substantially outperforms the other state-of-the-art algorithms, especially asymptotically for large datasets. The F-score of our algorithm is actually about 10%, 15% and 25% higher than the MMHC, Wei Luo and Or+Inter.Iamb algorithms respectively when the number of deterministic nodes lies between 1 and 4. The overall Recall and Precision rates are slightly lower but still significant. The obtained performance of our algorithm follows mainly from its learning rules dedicated to deterministic nodes: by using conditional entropy, it avoids discarding edges that are needed and, by Proposition 5.3, it correctly orients the edges in the neighborhood of deterministic nodes. This explains why its Recall is higher than the other methods. For the

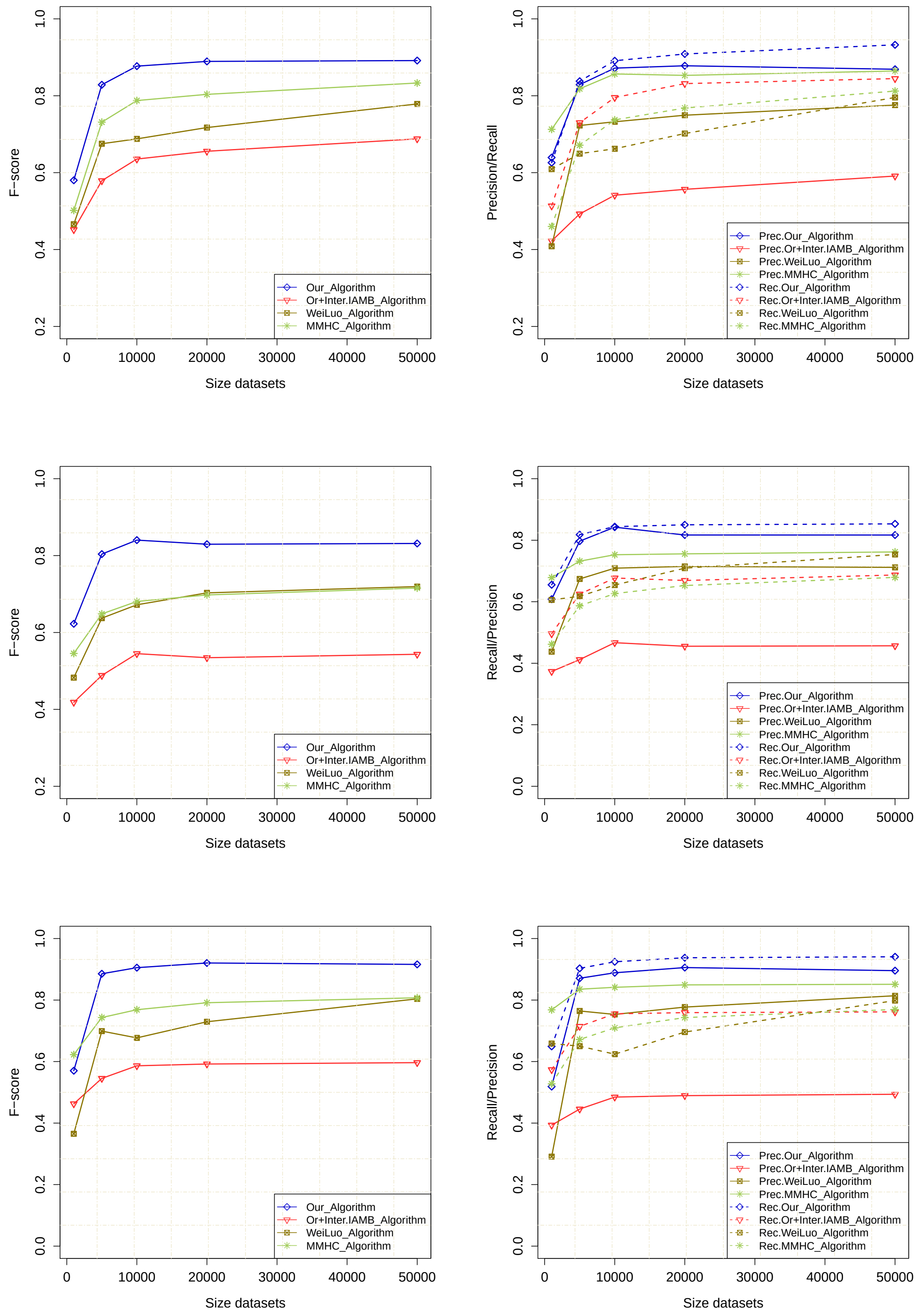


FIGURE 6.1: F-score (left side), Recall and Precision (right side) results in function of the number of deterministic nodes in $\mathcal{G}$.

F-score and Precision, in addition to correctly recovering dependences, Propositions 5.1 and 5.2 enabled our algorithm to remove arcs that the other methods were unable to remove (when statistical indistinguishability occurs for instance), hence making it more

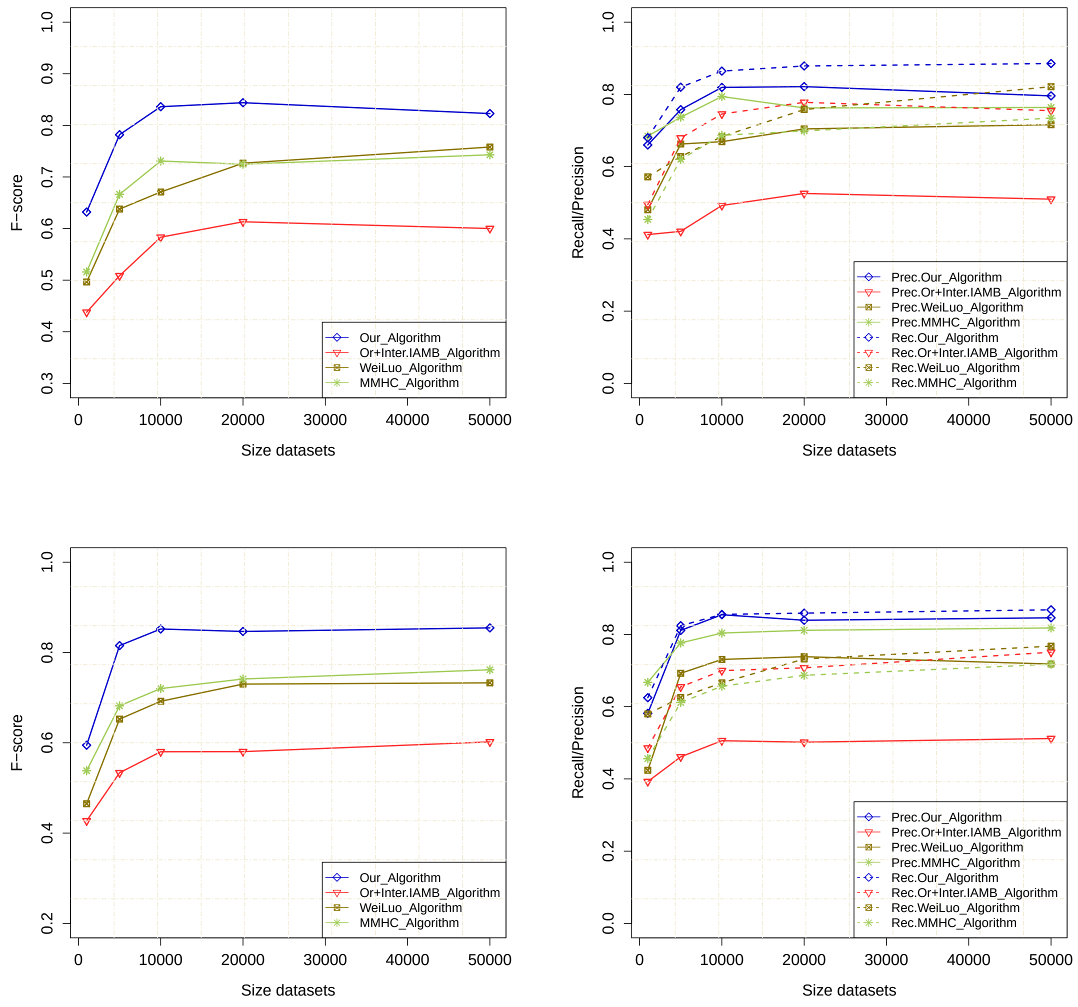


FIGURE 6.2: F-score, Recall and Precision with a number of arcs ranging from 12 to 50 and a number of parents smaller than or equal to 4 in $\mathcal{G}$: the top curves and the bottom curves are averages for BNs with 12 to 30 arcs and 31 to 50 arcs respectively.

suited for recovering independences. Finally, our original selection of the conditioning sets in the G^2-tests also helped discovering conditional independences. Roughly speaking, unlike state-of-the-art algorithms, via our approach, we succeeded to convert the problem of unfaithfulness induced by the existence of deterministic nodes in $\mathcal{P}$ to a valuable information that has been exploited to enhance and optimize the reliability of the structure learning process.

The accuracy of our algorithm is also demonstrated by varying the parameters related to the complexity of $\mathcal{G}$ and the number of parents as shown in Figures 6.2, 6.3, 6.4 and 6.5: in those figures, average F-scores, Recalls and Precisions are computed on BNs with arcs ranging from 12 to 170 arcs.

From these curves, except in rare exceptions when the datasets contain only 1000 records, our algorithm always outperforms the state-of-the-art methods, whatever the complexity

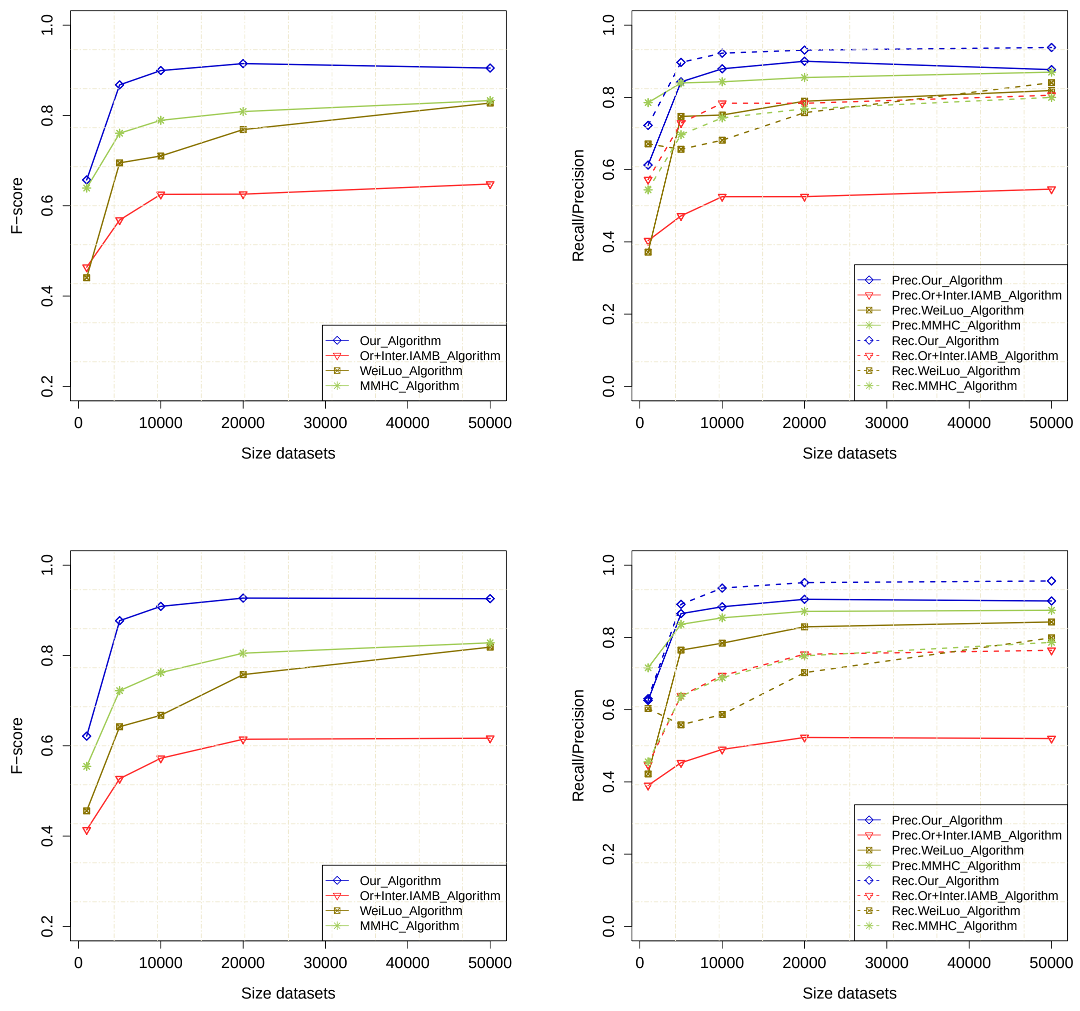


FIGURE 6.3: F-score, Recall and Precision with a number of arcs ranging from 51 to 170 and a number of parents smaller than or equal to 4 in $\mathcal{G}$: the top curves and the bottom curves are averages for BNs with 51 to 70 arcs and 71 to 170 arcs respectively.

of the BN. The obtained scores as well as the difference w.r.t. the others methods are characterized by a lower variance (or a better stability) and the F-score of our algorithm is always about 30%, 12% and 10% greater than the Or+Inter.Iamb, Wei Luo and MMHC algorithms respectively.

It should be noted that despite the rules dedicated to handle deterministic relations when using the Or+Inter.Iamb algorithm, the latter is doomed to be ineffective to learn a good structure (actually, in the best of cases, it does not exceed the F-score rate of 60% (for all data sizes) and obtains always the lowest F-score results, as seen in Figures 6.2 to 6.5). This can be explained by the fact that the substitution of the "and" operator by the "or" operator when executing the MMPC algorithm induces the addition of an important set of False Positive edges in all the substructures of $\mathcal{G}$ where the faithfulness assumption holds. For a better understanding of this result, refer to Figures 6.1 to 6.5: as can be seen, the Recall (True Positive rate) obtained by Or+Inter.Iamb exceeds the rate of 80% for dataset sizes ranging from 10000 to 50000 and a number of deterministic nodes ranging from 1 to 4. For the other values of the parameters, OR+Inter.Iamb obtains also good results in terms of Recall, outperforming in many cases MMHC and Wei Luo approaches. These results are not surprising because the Or+Inter.Iamb approach addresses efficiently the problem of the False Negative (FN) edges resulting from the deterministic nodes in $\mathcal{G}$. However, we can clearly notice in the same figures that OR+Inter.Iamb is faced with a problem of False Positive edges/arcs, especially when the number of deterministic nodes becomes important in $\mathcal{G}$. This explains why the best F-score reached does not exceed the rate of 60%. In addition, the impact of the number of deterministic nodes on the Precision of OR+Inter.Iamb can be also seen in Figure 6.1, where it decreases from 59% (when the dataset size is equal to 50000 and the number of deterministic nodes lies between 1 and 4) to 49% for the same dataset size and a number of deterministic nodes between 10 and 15.

It should be noted that, despite the unfaithfulness of the distribution, MMHC (which is not suited to cope with deterministic nodes) obtains good results in terms of F-score and outperforms in many occasions the other two state-of-the-art algorithms. In reality, this good behavior of MMHC does not mean necessarily that MMHC is appropriate for the task of BN structure learning under unfaithfulness, but it simply entails that MMHC learns better than Wei Luo and OR+Inter.Iamb algorithms some substructures in $\mathcal{G}$ for which the faithfulness is verified. This argument will be justified in the next section where, as we shall see, the efficiency of the MMHC algorithm becomes the lowest one when restricting our experimentations to the neighborhood of $\mathcal{G}$'s deterministic nodes.

6.2.2 Evaluation of substructures around deterministic nodes

In this section, we perform the evaluation of the studied algorithms by computing the Recall, Precision and F-score rates only around deterministic nodes of their resulting CPDAGs, i.e., only over the edges/arcs for which at least one extremal node is deterministic. For instance, if we are interested to evaluate the structure $\mathcal{G}$ depicted by Figure 6.6, only edges $Z-U$, $X-Z$, $U-V$, and $W-U$ are taken into consideration. In this figure, the obtained Precision values around deterministic nodes Z and U are respectively equal to 0.5 and 1, hence an average equal to 0.75 for the overall structure. Concerning the Recall values around the same deterministic nodes, they are respectively equal to 0.5 and 0.67, hence an average of 0.585.

It should be noted that the evaluations performed in this part of our experimentations rely on the BN benchmarks considered in the previous section, where we limit the boundaries of our evaluations on the neighborhood of each deterministic nodes in order to measure the sensitivity of each algorithm when learning these unfaithful parts. In other

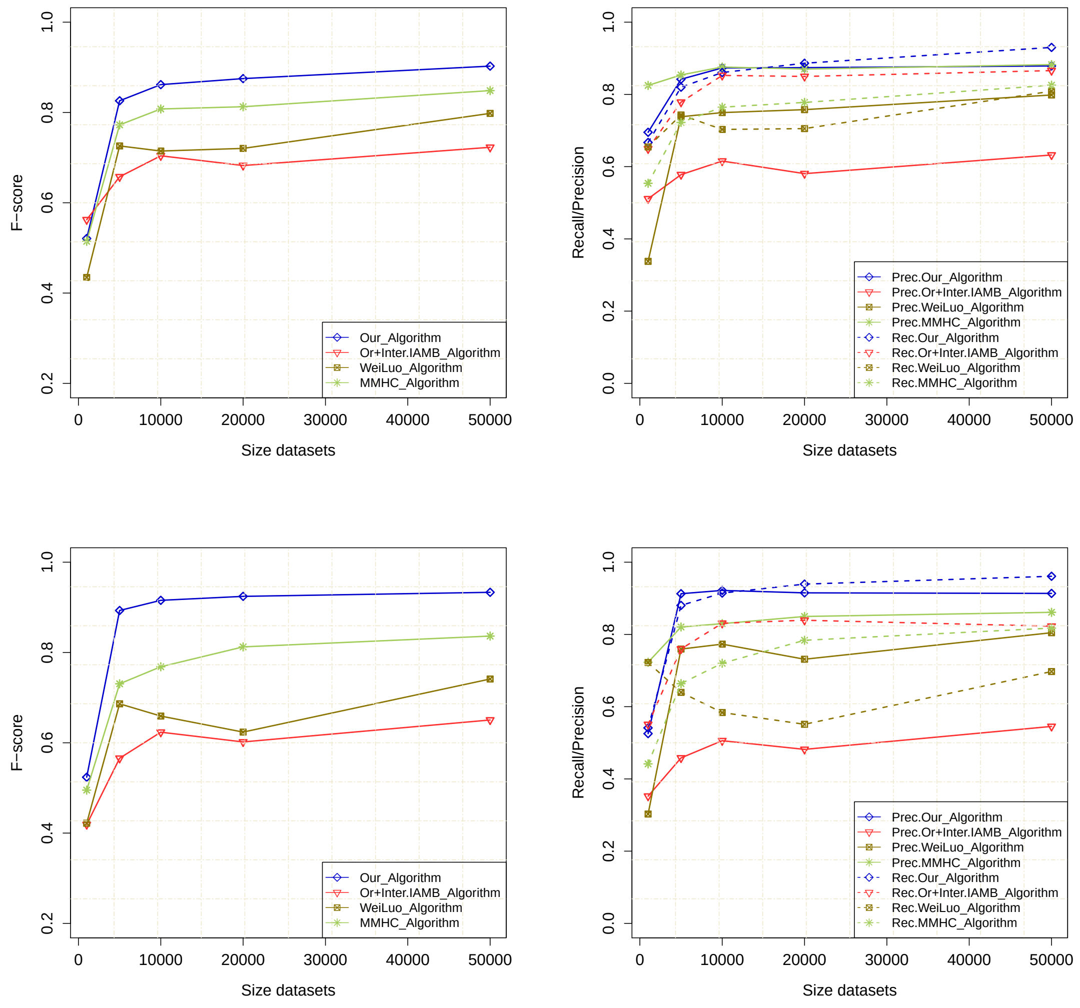


FIGURE 6.4: F-score, Recall and Precision with a number of arcs ranging from 12 to 50 and a number of parents greater than 4 in $\mathcal{G}$: the top curves and the bottom curves are averages for BNs with 12 to 30 arcs and 31 to 50 arcs respectively.

words, throughout these experiments, we wish to measure the robustness of each learning algorithm against the unfaithfulness incurred by deterministic nodes.

Tables 6.2 and 6.3 display the averages and standard deviations over the 750 datasets of the Recall, Precision and F-score for the four algorithms. The results shown in both tables have been generated by comparing the learnt substructures (by considering both the skeleton and CPDAG of $\mathcal{G}$) representing the neighborhood of each deterministic node w.r.t. to the original version. In addition to the mentioned evaluation criteria, we note in these tables the presence of an additional criterion in our results denoted by “var%Overall”. This metric indicates the differences between the results obtained when considering the overall learnt structure (as the previous experimentations do) and those obtained when restricting ourselves to some specific substructures in $\mathcal{G}$ (around deterministic nodes). Note that a positive var%Overall denoted by “+” means that we

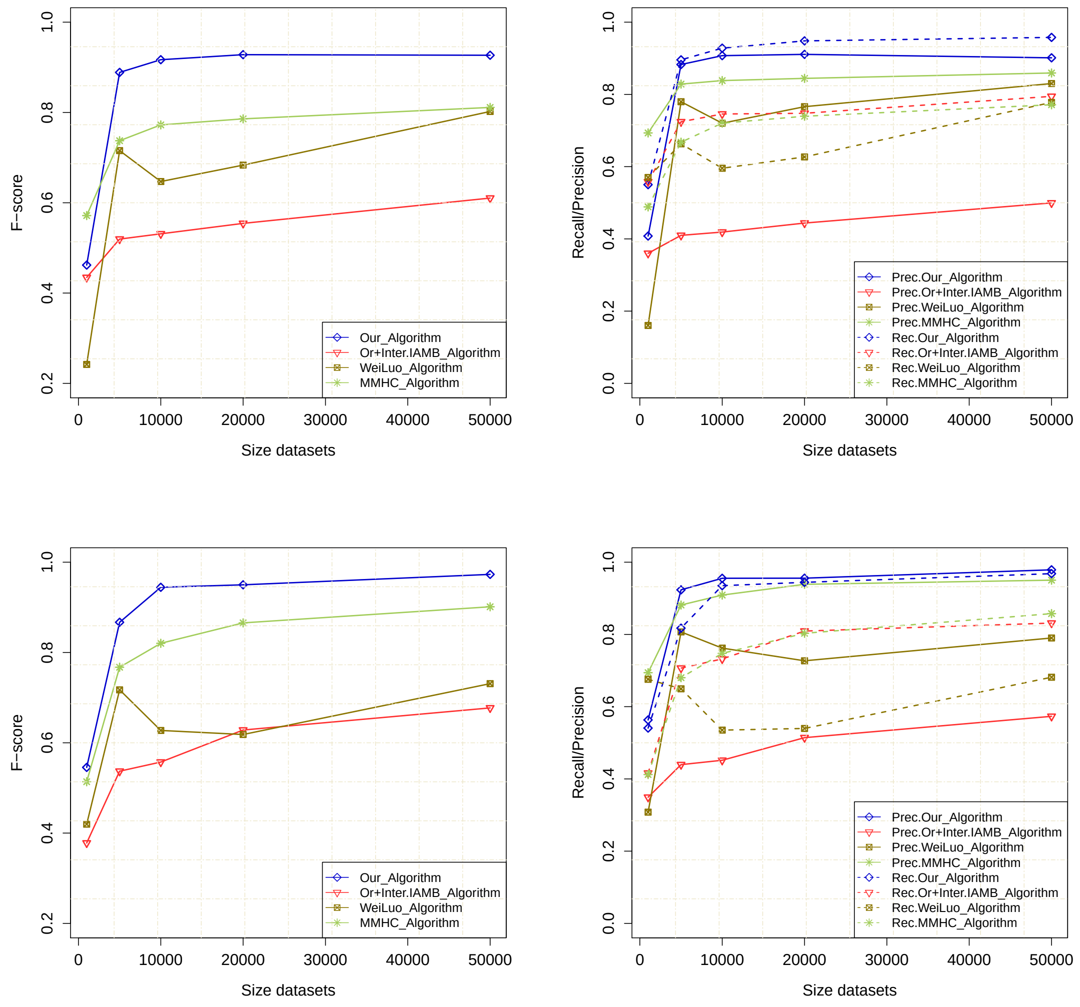


FIGURE 6.5: F-score, Recall and Precision with a number of arcs ranging from 51 to 170 and a number of parents greater than 4 in $\mathcal{G}$: the top curves and the bottom curves are averages for BNs with 51 to 70 arcs and 71 to 170 arcs respectively.

obtain a better result when we evaluate only the neighborhood of deterministic nodes and "-" otherwise.

Before discussing in details the results shown in Tables 6.2 and 6.3, note that we could expect from the beginning that the behaviors of the studied algorithms in these experimentations would be very different from those of the previous section since they do not highlight the same properties: they actually focus on the unfaithfulness. Notably, we could expect that the MMHC algorithm would produce the worse results since it is not adapted to deal with deterministic relations.

As can be seen in both skeletons and CPDAGs, our algorithm significantly outperforms the state-of-the-art algorithms in terms of F-score for all dataset sizes. These results highlight again the effectiveness of our proposed rules, notably that of Proposition 5.3

Sizes	Criterion	OR+Inter.Iamb	Wei Luo	Our algorithm	MMHC
1000	Recall	0.75±0.29	0.72±0.26	**0.77 ± 0.28**	0.62± 0.29
	Precision	0.89± 0.18	0.90±0.17	0.89±0.20	**0.97 ± 0.14**
	F-score	0.78±0.24	0.77±0.22	**0.80 ± 0.24**	0.72±0.24
	var%Overall	+0.34	+0.34	+0.21	+0.17
5000	Recall	0.83±0.25	0.76 ± 0.25	**0.86** ± 0.23	0.71±0.29
	Precision	0.91±0.14	**0.95 ± 0.12**	0.94±0.12	0.95±0.15
	F-score	0.843±0.20	0.82±0.20	**0.88 ± 0.19**	0.78±0.24
	var%Overall	+0.31	+0.15	+0.05	+0.08
10000	Recall	0.83±0.25	0.78±0.26	**0.87 ± 0.22**	0.72±0.28
	Precision	0.92±0.13	0.94±0.14	0.94±0.12	**0.96 ± 0.089**
	F-score	0.85±0.20	0.83±0.21	**0.88 ± 0.18**	0.79±0.22
	var%Overall	+0.27	+0.16	+0.01	+0.05
20000	Recall	0.80±0.26	0.80±0.25	**0.86 ± 0.22**	0.73±0.27
	Precision	0.88±0.18	0.94±0.11	0.94±0.10	**0.97 ± 0.08**
	F-score	0.81±0.22	0.84±0.19	**0.88 ± 0.18**	0.80±0.21
	var%Overall	+0.22	+0.13	+0.0.01	+0.04
50000	Recall	0.77±0.26	0.84±0.22	**0.89 ± 0.22**	0.73±0.26
	Precision	0.84±0.23	0.94±0.11	**0.95 ± 0.11**	0.94±0.11
	F-score	0.78±0.24	0.87±0.18	**0.90 ± 0.18**	0.80±0.21
	var%Overall	+0.18	+0.11	+0.02	+0.02

TABLE 6.2: Evaluation of the BN skeletons around deterministic nodes

Sizes	Criterion	OR+Inter.Iamb	Wei Luo	Our algorithm	MMHC
1000	Recall	0.34±0.36	0.55±0.39	**0.64 ± 0.37**	0.50±0.38
	Precision	0.41±0.42	0.63±0.40	0.69±0.38	**0.70 ± 0.42**
	F-score	0.36±0.37	0.57±0.38	**0.66 ± 0.37**	0.57±0.38
	var%Overall	-0.08	+0.14	+0.07	+0.02
5000	Recall	0.47±0.42	0.62±0.34	**0.75 ± 0.35**	0.62±0.37
	Precision	0.51±0.42	0.75±0.32	**0.79 ± 0.33**	0.78±0.35
	F-score	0.48±0.41	0.66±0.32	**0.76 ± 0.34**	0.67±0.35
	var%Overall	-0.05	-0.01	-0.07	-0.03
10000	Recall	0.52±0.42	0.65±0.35	**0.75 ± 0.35**	0.62±0.37
	Precision	0.55±0.42	0.76±0.35	**0.79 ± 0.33**	0.76±0.35
	F-score	0.52±0.42	0.69±0.34	**0.76 ± 0.34**	0.66±0.36
	var%Overall	-0.06	+0.02	-0.11	-0.08
20000	Recall	0.51±0.41	0.68±0.33	**0.74 ± 0.34**	0.64±0.36
	Precision	0.54±0.41	0.77±0.31	0.77±0.33	**0.79 ± 0.34**
	F-score	0.51±0.40	0.71±0.31	**0.75 ± 0.33**	0.69±0.34
	var%Overall	-0.08	0	-0.12	-0.07
50000	Recall	0.53±0.40	0.71±0.32	**0.74 ± 0.36**	0.63±0.36
	Precision	0.56±0.42	**0.77 ± 0.32**	0.76±0.35	0.74±0.36
	F-score	0.53±0.40	0.73±0.32	**0.74 ± 0.35**	0.67±0.35
	var%Overall	-0.07	-0.03	-0.04	-0.11

TABLE 6.3: Evaluation of the BN CPDAGs around deterministic nodes

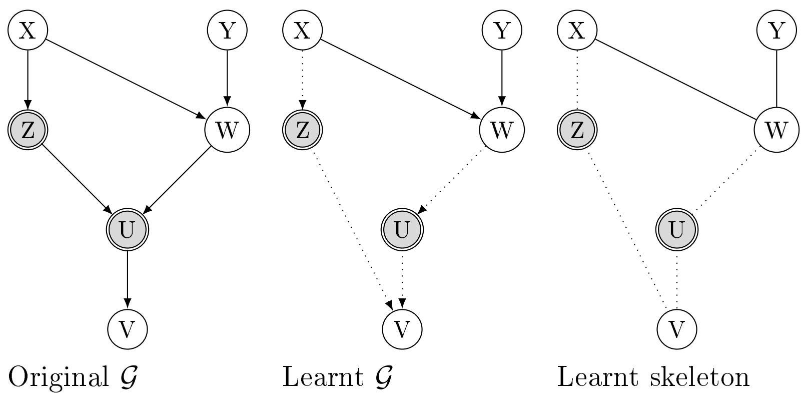
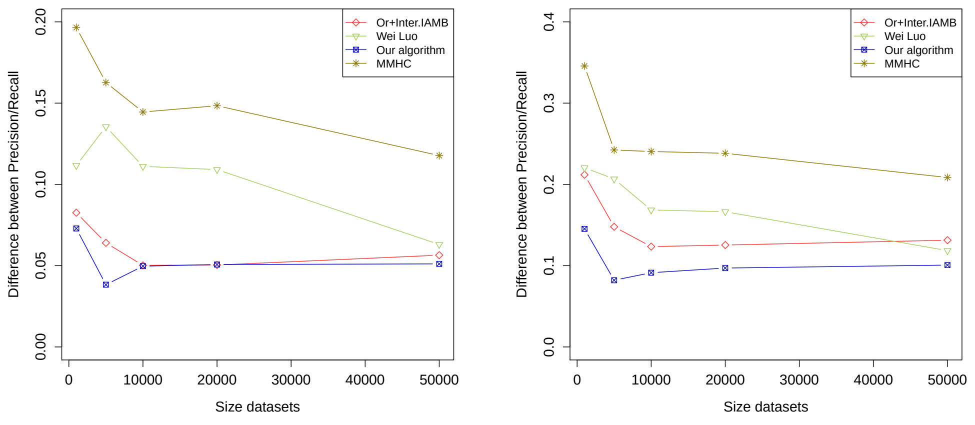


FIGURE 6.6: Evaluations around deterministic nodes



(a) CPDAG case (b) Skeleton case

FIGURE 6.7: Tradeoff between Precision and Recall of BN structure learning algorithms

which deals with the problem of unfaithfulness by adding a constraint about arcs deletions around deterministic nodes in $\mathcal{G}$ when performing the greedy search step.

In the skeleton context (Table 6.3), the difference in terms of Recall between our algorithm and MMHC exceeds 13% for all dataset sizes, while it is between 5%-9% and 2%-6% higher than Wei Luo and Or+Inter.Iamb approaches respectively. As explained previously, these results fit well with our expectations regarding the lowest Recall rate obtained by the MMHC algorithm. As a matter of fact, the first phase of MMHC (using the MMPC algorithm) leads to the deletion of an important set of false negative edges around the deterministic nodes that cannot be recovered during its second greedy search phase (due to the constraints related to the edge addition operations). However, as can be seen in Table 6.2, MMHC outperforms in most cases the other algorithms, including our approach, in terms of Precision, which is not a surprising result given the following reasons:

- in our approach but also in the state-of-the-art algorithms, the additional learning rules and the rigid controls dedicated to the neighborhood around each deterministic node does not guarantee a perfect result, and may obviously prevent the deletion of some false positive edges from $\mathcal{G}$ (due to statistical independence errors, insufficient data, etc.).

- as the number of arc/edges around deterministic nodes decreases, the Precision rate increases. For instance, if the learning algorithm returns an empty neighborhood set around U and Z nodes (see Figure 6.6), the overall obtained Precision rate in such a case will be equal to 1 (which represents the highest value).
- The Precision or Recall values alone do not reflect the effectiveness of the algorithm. That is the reason why it is important to ensure a good trade-off between these two criteria; when the difference between the Recall and Precision rates is lower, the reliability of the algorithm is higher. By referring to this information, the tradeoff shown by MMHC is the lowest one (the highest difference) as we can be seen in Figure 6.7.(b).

As we can see in Table 6.2, the increase in terms of the F-score values generated by Wei Luo and OR+inter.Iamb is significant (according to those obtained from the whole CPDAG): it ranges from 18% to 35% and from 11% to 34% respectively, outperforming therefore the results obtained by MMHC. These results highlight clearly the unstable characteristic of both previous algorithms, since they do not build in an homogeneous way the different parts of the structure, i.e., the learning effort is mainly concentrated around deterministic nodes while the rest of $\mathcal{G}$ is completely ignored (statistical indistinguishability [Luo06] and False Positive edges additions [RdMAC08]). Unlike these approaches, ours tries to establish a balance in terms of learning efficiency between the substructures where the faithfulness is guaranteed and those which suffer from the problem of determinism, i.e., our proposed learning rules guarantee the reliability of the structure building by calibrating the learning process when it is necessary without favoring some substructures to others. Regarding the MMHC algorithm, the increase shown by MMHC in terms of F-score is marginal (between 0.02 and 0.17), but it does not presents a good sign of reliability given the bad tradeoff between the Recall and Precision values (as shown in Figure 6.7.(b)).

Regarding the CPDAG case, as it is shown in Table 6.3, we can clearly remark that the results characterizing Wei Luo and OR+Inter.Iamb undergo a significant decrease comparing to those mentioned in Table 6.2, giving thereby the second position to the MMHC algorithm. Once again, our approach outperforms the rest of the algorithms in terms of Recall, F-score and also Precision except for the case when the dataset size is equal to 1000. This can be explained by the fact that unlike state-of-the-art approaches, our algorithm relies on Proposition 5.3 to infer in an original way the orientations of edges around deterministic nodes, which are therefore maintained during the second phase (to cope with the possible errors of orientation/deletion of these arcs). Such rule does not appear in the other algorithms, where the main goal is the correct learning of the skeleton (as shown in the results of Table 6.2). It should be emphasized that the difference between the results mentioned here and those obtained for the whole CPDAG structure using our approach are almost the same (variations between -0.04 and 0.12). This remains also valid for the tradeoff curves between Precision and Recall where the gaps between them, as shown in Figure 6.7.(a), are always the lowest ones compared to those of the other algorithms.

To summarize, in all the results that we have shown and for all parameters variations, our algorithm significantly outperforms the other classical learning approaches, almost on all criteria. Another important point that should be noted in this context concerns the remarkable stability characterizing all the results returned by our approach. Unlike our algorithm, the behaviors the other ones varies hugely w.r.t. the different parameters (complexity, CPDAG, skeleton, *etc.*).

6.3 Conclusion

In this chapter, we have performed an experimental study on the structure learning approach developed in the preceding chapter. We compared it with three state-of-the-art algorithms: MMHC, Wei Luo and OR+Inter.Iamb. The experimentations were carried out using a set of synthetic datasets generated from random BNs containing deterministic nodes. The comparisons between the studied algorithms were performed by computing the Recall, Precision and F-score rates of each skeleton/CPDAG learnt by each algorithm given the original BN. As shown in the previous sections, our proposed rules have enabled our approach to significantly outperform the other ones in terms of the quality of the structure learnt. This improvement has been shown throughout the evaluations made on the overall structures and on the set of substructures around the deterministic nodes (the source of the problems).

Performing comparisons in these two contexts (global/local structure around the deterministic nodes) highlighted the regions in which the algorithms focused their learning efforts. Clearly, Wei Luo's and OR+Inter.Iamb's rules focus too much on the deterministic neighborhood, at the expense of the regions in which faithfulness holds (for such regions, they make mistakes that could be avoided). MMHC being not tailored to deal with unfaithfulness, it performs significant mistakes around the deterministic nodes. Unlike the other methods, our approach provides a good balance between global and local contexts, which explains why it outperforms the other algorithms.

Chapter 7

Cluster-based multivariate discretization

For several decades, Bayesian networks (BN) have been successfully exploited for dealing with uncertainties. However, while their learning and inference mechanisms are relatively well understood when they involve only discrete variables, their coping with continuous variables is still often unsatisfactory. One actually has to trade-off between expressiveness and computational complexity: on one hand, conditional Gaussian models and their mixing with discrete variables are computationally efficient but they definitely lack some expressiveness [LW89]; on the other hand, mixtures of exponentials, bases or polynomials are very expressive but at the expense of tractability [MRS01, SW11]. In between lie discretization methods which, by converting continuous variables into discrete ones, can provide a satisfactory trade-off between expressiveness and tractability.

In many real-world applications, BNs are learnt from data and, when there exist continuous attributes, those are often discretized prior to learning, thereby opening the path to exploiting efficient discrete variable-based learning algorithms. However such an approach is doomed to be ineffective because the conditional dependences/arcs learnt during the learning phase bring valuable information that cannot be exploited by the discretization algorithm, thereby severely limiting its effectiveness. Unfortunately, there are surprisingly few papers in the literature on discretizing while learning, probably because it incurs substantial computational costs and it requires multivariate discretization instead of just a univariate discretization. In this direction, Minimum Description Length (MDL) and Bayesian scores used by search algorithms have been adapted to include multivariate discretizations taking into account the BN structure learnt so far [FG96, MC98]. But, to be naturally included into these scores, the latter heavily rely on entropy-related maximizations which, as we shall see, is not very well suited for BN learning. In [SEHK11], a non-linear dimensionality reduction process (Gaussian process latent variable model – GP-LVM) combined with a Gaussian mixture model-based discretization is proposed for BN learning. Unfortunately, the GP-LVM step maps the random variables space into another space, hence loosing their semantics as well as a lot of information. In addition, the second step does not rely on the BN structure and, consequently, do not exploit other useful information.

Unlike in BN learning, multivariate discretization has often been exploited in the machine learning community for supervised classification tasks [Ker92, Bou04, FI93, ZRR98, Bou06]. But the goal is only to maximize the classification power w.r.t. one target variable. As such, only the individual correlations of each variable with the target are of interest and, thus, only bivariate discretization is needed. BN structure learning is fundamentally different because the complete set of conditional dependences between all sets of variables is of interest and multivariate discretization shall most often involve more than two variables. This makes these approaches not easily transferable to BN learning. In [KK99], the authors propose a general multivariate discretization relying on genetic algorithms to construct rulesets. However, here again, the approach is very limited in the sense that it is designed to cope with only one target class variable and, in addition, the domain size of this variable needs to be small to keep the method tractable.

The unsupervised learning community has also exploited discretizations, but those are essentially univariate [DKS95, JLZW09, Rat03, RZWL13], which make them usable *per se* only as a preprocess prior to learning. However, BN learning can be related to unsupervised learning in the sense that all the BN's random variables can be thought of as targets whose discretized values are unobserved. This suggests that some of the key ideas underlying unsupervised learning algorithms might be adapted for learning BN structures. Clustering is one such popular framework. In [MC99], for instance, multivariate discretization is performed by clustering but, unfortunately, independences between random variables are only considered given a latent variable. This limits considerably the range of applications of the method because numerous continuous variables require the latent variable to have a large domain size in order to get good quality discretizations. This approach is therefore limited to small datasets and, by not exploiting the BN structure, it is best suited as a BN learning preprocess. Finally, by relying on entropy, its effectiveness for BN learning is certainly not optimal. However, in this context, we advocate to exploit clustering methods for discretization w.r.t. BN learning.

More precisely, we propose two versions of a clustering-based approach for multivariate discretization that take into account the conditional dependences among variables discovered during the learning process. Both versions rely on the assumption that data are distributed according to truncated normal distributions within each interval of discretization. Their difference lies in the way the optimal distributions are evaluated: one version approximates them using a mixture of untruncated Gaussians combined with an optimization problem while the second version directly use truncated normal distributions. By exploiting clustering rather than entropy, both approaches overcome the shortcomings induced by the latter and, by taking into account the dependences between random variables, they significantly increase the quality of the discretization compared to state-of-the-art clustering approaches.

The rest of the chapter is organized as follows. We start by recalling some generalities about BN in order to introduce the notations necessary for the rest of the chapter, we also recall some discretization schema learning process. Next, we describe the general architecture of our algorithms for multivariate discretization while learning BN structure: those actually exploit conditional dependence/arc learnt during the learning phase to provide effective continuous variables discretization. Then we provide some details about these algorithms and justify their correctness. Finally, some concluding remarks are given.

7.1 Basics on BN Structure Learning and Discretization

To avoid ambiguities between continuous variables and their discretized counterparts, in the rest of this chapter, letters, when superscripted by "$\circ$", e.g., $\mathring{X}, \mathring{x}$, represent variables and their instantiations prior to discretization (for discrete variables, $\mathcal{X} = \mathring{\mathcal{X}}$ and $x = \mathring{x}$). In the rest of the chapter, n will always denote the number of variables in the BN (see Definition 2.1 Chapter 2), and we will assume that $X_1, \dots, X_l$ are discrete whereas $\mathring{X}_{l+1}, \dots, \mathring{X}_n$ are continuous. N will always represent the number of records in the input database $\mathring{\mathcal{D}}$. Given $\mathring{\mathcal{D}} = \{\mathring{\mathbf{x}}^{(1)}, \mathring{\mathbf{x}}^{(2)}, \dots, \mathring{\mathbf{x}}^{(N)}\}$, BN learning consists of finding DAG $\mathcal{G}$ that most likely accounts for the observed data in $\mathring{\mathcal{D}}$. Recall, that BN structure learning algorithm can be divided into 3 classes [KF09]:

i) the search-based approaches that look for the structure optimizing a score (BD, BDeu, BIC, AIC, K2, *etc.*);

ii) the constraint-based approaches that exploit statistical independence tests (χ^2, G^2, *etc.*) to find the best structure $\mathcal{G}$;

iii) the hybrid methods that exploit a combination of both.

In the rest of the chapter, we will focus on search-based approaches since our closest competitors, [MC98, FG96], belong to this class of DAGs search.

Remind that these algorithms start with a structure $\mathcal{G}_0$ (often empty). Then, at each step, they look in the neighborhood of the current structure for another structure, say $\mathcal{G}$, that increases the likelihood of structure $\mathcal{G}$ given observations $\mathcal{D}$, i.e., $P(\mathcal{G}|\mathcal{D})$. The neighborhood is often defined as the set of graphs that differ from the current one only by one atomic graphical modification (arc addition, arc deletion, arc reversal). $P(\mathcal{G}|\mathcal{D})$ is computed locally through the aforementioned scores. As seen in Chapter 3, by assuming a uniform prior on all structures $\mathcal{G}$, we have that:

$$P(\mathcal{G}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})} \propto P(\mathcal{D}|\mathcal{G}) = \int_{\boldsymbol{\theta}} P(\mathcal{D}|\mathcal{G},\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathcal{G})d\boldsymbol{\theta}, \tag{7.1}$$

The different hypotheses on prior π and on $\boldsymbol{\theta}$ result in the different scores (see, e.g., [S+78] for the hypotheses for the BIC score used later).

When database $\mathring{\mathcal{D}}$ contains continuous variables, those can be discretized. A discretization of a continuous variable $\mathring{X}$ is a function $f : \mathbb{R} \to \{1, \ldots, g\}$ defined by an increased sequence of g cut points $\{d_1, d_2, ..., d_g\}$ such that:

$$f(\mathring{x}) = \begin{cases} 0 & \text{if} \quad \mathring{x} < d_1, \\ k & \text{if} \quad d_k \le \mathring{x} < d_{k+1}, \quad \text{for all } k \in \{1, \ldots, g-1\} \\ g & \text{if} \quad \mathring{x} \ge d_g. \end{cases}$$

In other words, the discretization task aims to find the number of intervals and its corresponding set of cut points that jointly maximize the used objective (or scoring) function.
Let $\mathcal{F}$ be a set of discretization functions, one for each continuous variable. Then, given $\mathcal{F}$, if $\mathcal{D}$ denotes the (unique) database resulting from the discretization of $\mathring{\mathcal{D}}$ by $\mathcal{F}$., the BN scoring function defined by Equation (7.1) becomes:

$$P(\mathcal{G}|\mathring{\mathcal{D}},\mathcal{F}) \propto P(\mathring{\mathcal{D}}|\mathcal{G},\mathcal{F}) = P(\mathcal{D}|\mathring{\mathcal{D}},\mathcal{G},\mathcal{F})P(\mathring{\mathcal{D}}|\mathcal{G},\mathcal{F}) = P(\mathring{\mathcal{D}}|\mathcal{D},\mathcal{G},\mathcal{F})P(\mathcal{D}|\mathcal{G},\mathcal{F}),$$

Assuming that all databases $\mathring{\mathcal{D}}$ compatible with $\mathcal{D}$ given $\mathcal{F}$ are equiprobable, we thus have that:

$$P(\mathcal{G}|\mathring{\mathcal{D}},\mathcal{F}) \propto P(\mathcal{D}|\mathcal{G},\mathcal{F}) = \int_{\boldsymbol{\theta}} P(\mathcal{D}|\mathcal{G},\mathcal{F},\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathcal{F},\mathcal{G})d\boldsymbol{\theta}. \tag{7.2}$$

From Equation (7.2), we can therefore conclude that instead of finding the structure $\mathcal{G}^*$ that maximizes the likelihood given the discrete data, BN structure learning with continuous random variables amounts to find the structure $\mathcal{G}^*$ such that:

$$\mathcal{G}^* = \underset{\mathcal{G}}{\text{Argmax}}\, P(\mathcal{G}|\mathcal{D},\mathcal{F}) \tag{7.3}$$

Note that $P(\mathcal{D}|\mathcal{G},\mathcal{F},\boldsymbol{\theta})$ corresponds to a classical score over discrete data, $\pi(\boldsymbol{\theta}|\mathcal{F},\mathcal{G})$ is the prior over the parameters of the BN given $\mathcal{F}$.

Equation (7.2) is precisely the one used when discretization is performed as a preprocess before learning.

When discretization is performed while learning, like in [MC98, FG96], both the structure and the discretization should be optimized simultaneously. In other words, the problem consists in computing $\text{Argmax}_{\mathcal{F},\mathcal{G}} P(\mathcal{G},\mathcal{F}|\mathring{\mathcal{D}})$, where finding the best discretization amounts to find the best set of cut points (including the best size for this set) for each continuous random variable. And we have that:

$$P(\mathcal{G},\mathcal{F}|\mathring{\mathcal{D}}) = P(\mathcal{G}|\mathcal{F},\mathring{\mathcal{D}})P(\mathcal{F}|\mathring{\mathcal{D}}) \propto P(\mathcal{F}|\mathring{\mathcal{D}}) \int_{\boldsymbol{\theta}} P(\mathcal{D}|\mathcal{G},\mathcal{F},\boldsymbol{\theta})\pi(\boldsymbol{\theta}|\mathcal{F},\mathcal{G})d\boldsymbol{\theta}. \tag{7.4}$$

As can be seen, the resulting equation combines the classical score on the discretized data (the integral) with a score $P(\mathcal{F}|\mathring{\mathcal{D}})$ for the discretization algorithm itself. The logarithm of the latter corresponds to what [FG96] and [MC98] call $DL_{\Lambda}(\Lambda) + DL_{\mathring{\mathcal{D}}\to\mathcal{D}}(\mathring{\mathcal{D}},\Lambda)$ and $\mathcal{S}_c(\Lambda;\mathring{\mathcal{D}})$ respectively (defined in the chapter about discretization).

7.2 A New Multivariate Discretization-Learning Algorithm

As mentioned earlier, we believe that taking into account the conditional dependences/independences between random variables is important to provide high-quality discretizations that minimize the information loss. Our approach thus follows Equation (7.4) and our goal is to find the discretization corresponding to $\text{Argmax}_{\mathcal{F},\mathcal{G}} P(\mathcal{G},\mathcal{F}|\mathring{\mathcal{D}})$. It must be emphasized that optimizing jointly over $\mathcal{F}$ and $\mathcal{G}$ is too computationally intensive a task to be usable in practice. Fortunately, we can approximate it efficiently through a gradient descent algorithm, alternating optimizations over the discretization $\mathcal{F}$ given a fixed structure $\mathcal{G}$ and optimizations over the structure $\mathcal{G}$ given a fixed discretization $\mathcal{F}$. This suggests the BN structure learning method described as Algorithm 9.

Algorithm 9: Our structure learning architecture.

```
  Input: a database $\mathring{\mathcal{D}}$, an initial graph $\mathcal{G}$, a score function $sc$ on discrete variables
  Output: the structure $\mathcal{G}$ of the Bayesian network
1 repeat
2     Find the best discretization $\mathcal{F}$ given $\mathcal{G}$
3     $\{X_{l+1},\dots,X_n\} \leftarrow$ discretize variables $\{\mathring{X}_{l+1},\dots,\mathring{X}_n\}$ given $\mathcal{F}$
4     $\mathcal{G} \leftarrow \mathcal{G}$'s neighbor that maximizes scoring function $sc$ w.r.t. $\{X_1,\dots,X_n\}$
5 until $\mathcal{G}$ maximizes the score;
```

Note that the task of multivariate discretization when alternating between $\mathcal{G}$ and $\mathcal{F}$ as shown in Algorithm 9 becomes untractable especially when the size of the network as well as the number of continuous variables to be discretized is large. As discussed earlier, the multivariate discretization is much more time consuming than univariate or bivariate discretization since it considers many variables at each time. As such, Line 2 of Algorithm 9 could thus incur a strong overhead to the learning algorithm because the discretization search space increases exponentially with the number of variables to discretize. To alleviate this problem, we suggest a local search algorithm that iteratively fixes the discretizations of all the continuous variables but one and optimizes the discretization of the latter (given the other variables) until some stopping criterion is met. As such, discretizations being optimized one continuous variable at a time, the combinatorics and the computation times are significantly limited. Line 2 of Algorithm 9 can thus be detailed as Algorithm 10.

Algorithm 10: One-variable discretization architecture.

```
Input: a database $\mathring{\mathcal{D}}$, a graph $\mathcal{G}$, a scoring function $sc$ on discrete variables
Output: a discretization $\mathcal{F}$
1 repeat
2     $i_0 \leftarrow$ Select an element in $\{l+1, \dots, n\}$
3     Discretize $\mathring{X}_{i_0}$ given $\mathcal{G}$ and $\{X_1, \dots, X_{i_0-1}, X_{i_0+1}, \dots, X_n\}$
4 until stopping condition;
```

In the following, we provide the details related to our proposed approaches and we start by listing the set of assumptions under which our methods are capable to return a "good quality" discretization. Then we provide the way by which each continuous variable is discretized while taking into consideration its correlations with the other variables.

7.2.1 Assumptions

The remainder of this chapter proposes a multivariate discretization approach under the following assumptions:

Assumption 1. In our approach, we assume that within every interval, each continuous random variable, say X_{i_0}, is distributed w.r.t. a truncated normal distribution. Over its whole domain of definition, it is thus distributed as a mixture of truncated Gaussians defined as follows:

$$g(\mathring{x}_{i_0}|\boldsymbol{\Theta}) = \sum_{k=0}^{g} \pi_k f(\mathring{x}_{i_0}|\boldsymbol{\theta}_k) \tag{7.5}$$

where π_k represents the weight of the kth component (or truncated Gaussian) in the mixture (with the constraint that $\pi_k \geq 0$ and $\sum_{k=0}^{g} \pi_k = 1$) and $f(\cdot|\boldsymbol{\theta}_k)$ represents the probability density function (density for short) of the truncated Gaussian parameterized by $\boldsymbol{\theta}_k = (\mu_k, \sigma_k, \bigcup_{k=1}^{g}\{d_k\})$. Therefore:

$$f(\mathring{x}_{i_0}|\boldsymbol{\theta}_k) = \begin{cases} 0 & \text{if} \quad \mathring{x}_{i_0} < d_k, \\ \dfrac{h(\mathring{x}_{i_0}|\boldsymbol{\theta}_k)}{\displaystyle\int_{d_k}^{d_{k+1}} h(\mathring{x}|\boldsymbol{\theta}_k)dx} & \text{if} \quad \mathring{x}_{i_0} \in [d_k, d_{k+1}) \\ 0 & \text{if} \quad \mathring{x}_{i_0} \geq d_{k+1}. \end{cases} \tag{7.6}$$

Note that $h_k(.|\boldsymbol{\theta}_k)$ is the probability density function of the standard normal distribution which is defined by the following equation:

$$h(\mathring{x}_{i_0}|\boldsymbol{\theta}_k) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left[-\frac{1}{2}\left(\frac{\mathring{x}_{i_0} - \mu_k}{\sigma_k}\right)^2\right] \tag{7.7}$$

The term $\int_{d_k}^{d_{k+1}} h_k(\mathring{x}|\boldsymbol{\theta}_k)dx$ in Equation (7.6) represents a normalization factor, such that the integral of $f(\mathring{x}_{i_0}|\boldsymbol{\theta}_k)$ is equal to 1. Unfortunately, this integral cannot be expressed using a closed-form formula.

Given the above assumption, we can conclude that each observation $\mathring{x}_{i_0}$ is sampled from only one of the $g+1$ components of the mixture. It must be emphasized that, in our case, the choice of a model based on a mixture of truncated normal distributions to represent the distribution of the continuous random variables has been motivated by domain expert's knowledge. Actually, it fits well with the distributions of the physical variables characterizing the nuclear accidents. For example, when a nuclear accident situation occurs, pressure with very low (e.g., $<$0.1 bar) and very high ($>$30 bar) values cannot appear in reality, hence these values are not considered during accident simulations. In addition, many physical variables are always greater than a threshold (which is fixed by an expert).

Figure 7.1 gives an example of a truncated normal distributions mixture composed by 4 truncated Gaussians.

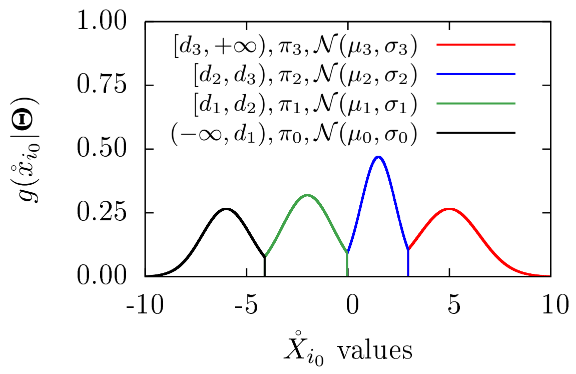


FIGURE 7.1: Truncated mixture Gaussian model example

Assumption 2. Let $\mathring{X}_{i_0}$ be a continuous random variable and let X_{i_0} be its discretized counterpart. Then $\mathring{X}_{i_0}$ is independent of the rest of the random variables given X_{i_0}.

Assumption 3. The observations in the database are mutually independent and identically distributed (*i.i.d. hypothesis*).

7.2.2 Discretization Criterion

Once the discretization architecture of the algorithm is defined, the next step consists in finding the discretization criterion under which the optimal $\mathcal{F}$ will be found. Recall that the implementation of Algorithm 10 needs a discretization criterion to be optimized throughout the search process. Basic ideas include trying to find cut points minimizing the discrepancy between the frequencies or the sizes of intervals $[d_i, d_{i+1})$. A more sophisticated approach consists of limiting as much as possible the quantity of information lost after discretization, or equivalently to maximize the quantity of information remaining after discretization. This naturally calls for maximizing an entropy. This is essentially what our closest competitors do [FG96, MC98] when performing their multivariate discretization while learning the BN structure. But entropy may not be the most appropriate measure when dealing with BNs.

In order to highlight the shortcoming of entropy based discretization methods, let us consider the example of the continuous variable distribution shown in Figure 7.2. We have a random variable $\mathring{A}$ with the following domain of definition $\{a_1, a_2, a_3\}$. Then, it is possible that, for some BN:

$$P(A = a_1) = \frac{1}{6} \qquad P(A = a_2) = \frac{1}{3} \qquad P(A = a_3) = \frac{1}{2}.$$

With a sufficiently large database $\mathcal{D}$, the frequencies of observations of a_1, a_2, a_3 in $\mathcal{D}$ would certainly be close to these probabilities. Hence, learning $P(A)$ from $\mathcal{D}$ would result in:

$$P(A = a_1) \approx \frac{1}{6} \qquad P(A = a_2) \approx \frac{1}{3} \qquad P(A = a_3) \approx \frac{1}{2}.$$

Now, assume that the observations in $\mathcal{D}$ are noisy, say with a Gaussian noise with an *infinitely small* variance, as in Figure 7.2. Then, after discretization, we shall expect to have 3 intervals with respective frequencies $\frac{1}{6}$, $\frac{1}{3}$ and $\frac{1}{2}$, i.e., intervals similar to $(-\infty, d_1)$, $[d_1, d_2)$ and $[d_2, +\infty)$ of Figure 7.2. However, w.r.t. entropy, the best discretization corresponds to intervals $[-\infty, s_1)$, $[s_1, s_2)$ and $[s_2, +\infty)$ of Figure 7.2 whose frequencies are all approximately equal to $\frac{1}{3}$ since the entropy criterion reaches its maximum value when the different modalities (or intervals) of a given variable are equiprobable (as seen in Section 5.2.2 of Chapter 5). Therefore, whatever the infinitesimal noise added to data in $\mathcal{D}$, an entropy-based discretization will always produce a discretized variable A with a uniform distribution $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ instead of $[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}]$. This suggests that entropy is probably not a good criterion for discretizing continuous variables for BN learning.

By giving the assumption of a truncated Gaussian mixture model as the probability distribution of $\mathring{A}$ (hereafter called TGM) on which relies our approach, Figure 7.2 suggests

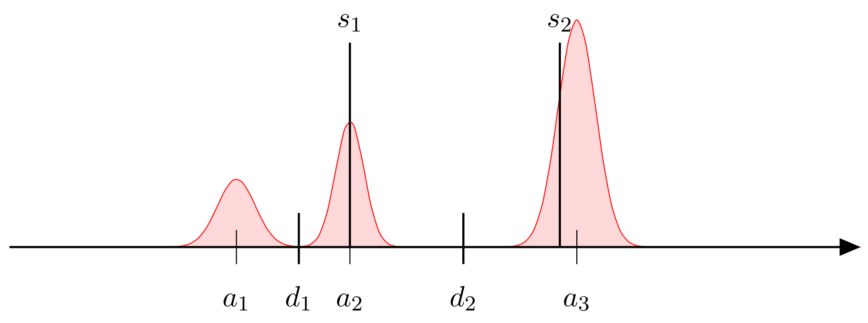


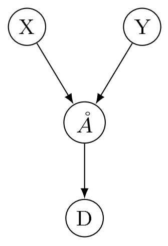
FIGURE 7.2: Discretization: entropy v.s. clustering.

that clustering would probably be more appropriate: here, one cluster/interval per truncated Gaussian would provide a better discretization. The weights of the mixture then form precisely the CPT of A (the discrete counterpart of $\mathring{A}$) in the discrete BN.

In particular, if $\mathring{A}$ has some parents in the BN, there are as many TGM as the product of the domain sizes of its parents in $\mathcal{G}$. The parameters of such a discretization scheme are therefore: i) a set of g cut points (to define $g+1$ intervals) and ii) a mean and a variance for each interval (to define its truncated Gaussian). Figure 7.3 depicts an example of BN with its corresponding CPTs before and after discretizing the continuous variable $\mathring{A}$ into three intervals.

In this example, all the parents of $\mathring{A}$ (composed by X and Y) are assumed to be binary, hence the product of their domain sizes as well as the number of TGMs are equal to 4. This leads to the following mixture models:

$$\begin{aligned}
(X = f \wedge Y = f) \Longrightarrow \mathcal{TGM}_0 : &\quad 0.1 f_0(\mathring{a}|\boldsymbol{\theta}_0) + 0.4 f_1(\mathring{a}|\boldsymbol{\theta}_1) + 0.5 f_2(\mathring{a}|\boldsymbol{\theta}_2) \\
(X = f \wedge Y = t) \Longrightarrow \mathcal{TGM}_1 : &\quad 0.9 f_0(\mathring{a}|\boldsymbol{\theta}_0) + 0.01 f_1(\mathring{a}|\boldsymbol{\theta}_1) + 0.09 f_2(\mathring{a}|\boldsymbol{\theta}_2) \\
(X = t \wedge Y = f) \Longrightarrow \mathcal{TGM}_2 : &\quad 0.9 f_0(\mathring{a}|\boldsymbol{\theta}_0) + 0 f_1(\mathring{a}|\boldsymbol{\theta}_1) + 0.1 f_2(\mathring{a}|\boldsymbol{\theta}_2) \\
(X = t \wedge Y = t) \Longrightarrow \mathcal{TGM}_3 : &\quad 0.1 f_0(\mathring{a}|\boldsymbol{\theta}_0) + 0.7 f_1(\mathring{a}|\boldsymbol{\theta}_1) + 0.2 f_2(\mathring{a}|\boldsymbol{\theta}_2)
\end{aligned} \tag{7.8}$$



		$P(\mathring{A}\mid X,Y)$
X	Y	$\mathring{A}$
f	f	$\mathcal{TGM}_0$
f	t	$\mathcal{TGM}_1$
t	f	$\mathcal{TGM}_2$
t	t	$\mathcal{TGM}_3$

		$P(A\mid X,Y)$		
		A		
X	Y	a_1	a_2	a_3
f	f	0.1	0.4	0.5
f	t	0.9	0.01	0.09
t	f	0.9	0	0.1
t	t	0.1	0.7	0.2

Hybrid BN CPT for continuous node $\mathring{A}$ CPT after discretization of $\mathring{A}$

FIGURE 7.3: Example of mixture Gaussian and discrete CPTs

Note that the parameters $\boldsymbol{\theta}_k$ of each component f_0, f_1, f_2 of the example do not depend on the values of the parents: they are fixed once and for all. Only the weight parameters $\{\pi_k\}$, which are needed to represent the mixture model, change from $\bigcup_{k=0}^{2} \pi_k$ when $\mathring{A}$ has no parent to $\bigcup_{j=1}^{4}(\bigcup_{k=0}^{2} \pi_{jk})$ when its parents are X and Y. In other words, the dependence between the mixture weights and the values taken by the parents of the continuous node in $\mathcal{G}$ is the only difference between univariate and multivariate discretization.

7.2.3 Two proposed versions for multivariate discretization

By assuming that each continuous random variable $\mathring{X}_{i_0}$ is distributed as a mixture of truncated normal distributions of parameters $\boldsymbol{\theta}_k$, the joint optimization of the model parameters $\boldsymbol{\Theta}$ is really hard to perform due to the normalization requirement that the integrals of the truncated Gaussian distributions of each interval must sum to 1 (as seen in Equation (7.6)). Actually, these integrals cannot be expressed in closed-form formulas. For this reason, we have chosen in a first version of the algorithm to relax this task by estimating a mixture of **untruncated** Gaussians (the normalization factor does not appear in such a case), and then approximating the truncated model using the parameters computed previously. In a second version, we have proposed a more sophisticated approach which directly uses truncated Gaussians to make the continuous variables discretization. This results in the development of two versions of multivariate discretization algorithms:

Version 1. we alleviate the discretization computational burden by proposing an approximation of the optimal cut points, means and variances of the truncated mixture model using a two-step process:

- first, we approximate the density of the joint $\{X_1, \ldots, X_{i_0-1}, \mathring{X}_{i_0}, X_{i_0+1}, \ldots, X_n\}$ as a mixture of *untruncated* Gaussians. Figure 7.4 shows an example of a Gaussian mixture model with its respective parameters.
- in a second step, we compute the best cut points w.r.t. the Gaussians. As each Gaussian is associated with an interval, the parts of the Gaussian outside the interval can be considered as a loss of information and we therefore look for cut points that minimize this loss. Figure 7.5 depicts a computation example of the set of cut points associated to the untruncated mixture model shown in Figure 7.4.

Version 2. it consists in jointly optimizing the mixture model parameters $\boldsymbol{\Theta}$ (cut point, mean variance and proportions) without performing any intermediate step (as the first approach does).

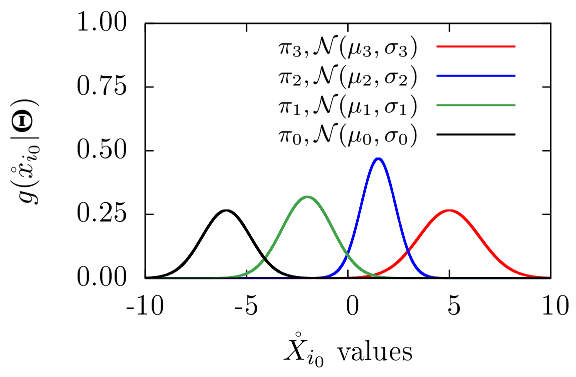


FIGURE 7.4: Mixture Gaussian parameters estimation

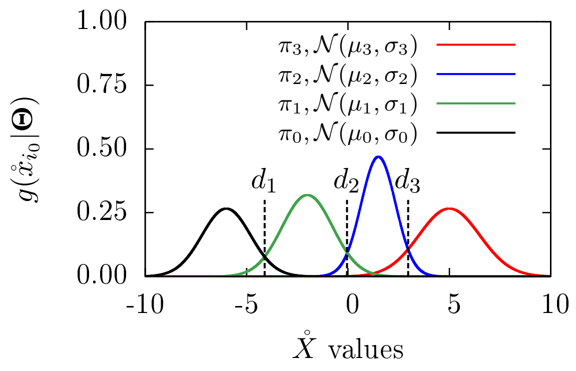


FIGURE 7.5: Cut point estimation for the mixture Gaussian model of Figure 7.4

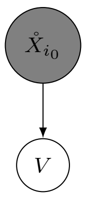
Now, let us delve into the details related to the parameters estimation of the truncated and untruncated mixture models. We start by describing the common shared principles of our proposed approaches that concern, notably, the way by which the BN structure (dependences/arcs) is taken into account when performing the discretization of each continuous variable $\mathring{X}_{i_0}$ in $\mathcal{G}$. Then, we provide the details of our proposed multivariate discretization algorithms (with and without truncated Gaussian distributions) and we justify their correctness.

7.2.4 Discretization exploiting the BN structure

Assume that structure $\mathcal{G}$ is fixed and that all the variables except $\mathring{X}_{i_0}$ are discrete. The density over all the variables, $p(\mathring{\mathbf{X}})$, is expressed by the following chain rule:

$$p(\mathring{\mathbf{X}}) = p(\mathring{X}_{i_0}|\mathbf{Pa}(\mathring{X}_{i_0})) \prod_{i \neq i_0} P(X_i|\mathbf{Pa}(X_i)) \tag{7.9}$$

where $p(\mathring{X}_{i_0}|\mathbf{Pa}(\mathring{X}_{i_0}))$ represents a mixture of the truncated Gaussians for each value of $\mathring{X}_{i_0}$'s parents as shown in the example of Figure 7.3 (there are always a finite number of values since all the variables except $\mathring{X}_{i_0}$ are discrete in the BN). $P(X_i|\mathbf{Pa}(X_i))$ should be the CPT of discrete variable X_i given its parents but, unfortunately, it is not well defined if $\mathring{X}_{i_0}$ (the continuous variable to be discretized) is part of $\mathbf{Pa}(X_i)$ because, in this case, $\mathbf{Pa}(X_i)$ has infinitely many values. An example of such a problem can be seen in Figure 7.6, where the continuous node $\mathring{X}_{i_0}$ is the parent of the discrete (Boolean) node V. As shown in this example, even for a simple BN with only two nodes, the resulting CPT of node V cannot be well defined (node $\mathring{X}_{i_0}$ takes an infinity of values), hence this prevents discretizing $\mathring{X}_{i_0}$ so as to maximize Equation (7.9). Fortunately, this problem can be easily overcome by enforcing that $\mathring{X}_{i_0}$ has no child while guaranteeing that the density represented by the whole BN structure remains unchanged. Actually, in [Sha86], Shachter provides some arc reversal operators that, when applied, never alter the density/probability distribution.



	$P(V\|\mathring{X}_{i_0})$	
$\mathring{X}_{i_0}$	t	f
$-\infty$	?	?
...	...	...
$\mathring{x}_{i_0}^{(i)}$	?	?
...	...	...
$\mathring{x}_{i_0}^{(k)}$	?	?
...	...	...
$+\infty$	?	?

FIGURE 7.6: CPT representation when a discrete node has continuous parent(s) in $\mathcal{G}$

Theorem 7.1. *An arc $X \rightarrow Y$ connecting two nodes in $\mathcal{G}$ can be replaced by arc $Y \rightarrow X$ without altering the distribution represented by the BN provided that the two following condition hold:*

1. *the arc reversal does not create a directed cycle;*
2. *all the parents of X are added to Y and all the parents of Y except X are added to X.*

With such a transformation, we can ensure that the continuous variable $\mathring{X}_{i_0}$ to be discretized at each iteration is represented by a leaf node in $\mathcal{G}$, hence the problem induced by the continuous parents described above is overcome given that the rest of the CPTs are composed by only discrete families (represented by finite CPTs). As an example of these transformations, reversing arc $X \rightarrow V$ of Figure 7.7.(a) results in Figure 7.7.(b) and, then, reversing arc $X \rightarrow W$ results in Figure 7.7.(c).

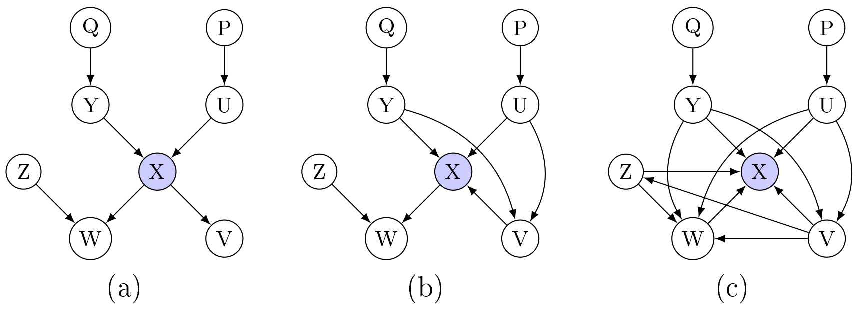


FIGURE 7.7: Shachter's arc reversals.

Therefore, to enforce that $\mathring{X}_{i_0}$ has no child, if $\{i_1, \ldots, i_c\}$ denotes the set of the index of the children variables of $\mathring{X}_{i_0}$, sorted by a topological order of $\mathcal{G}$, then, by reversing sequentially all the arcs $X_{i_j} \rightarrow \mathring{X}_{i_0}$, $j = 1, \ldots, c$, we get:

$$p(\mathring{\mathbf{X}}) = p(\mathring{X}_{i_0}|\mathbf{Pa}(\mathring{X}_{i_0})) \times \prod_{i \neq \{i_0,\ldots,i_c\}} P(X_i|\mathbf{Pa}(X_i)) \times \prod_{j=1}^{c} P(X_{i_j}|\mathbf{Pa}(X_{i_j})), \tag{7.10}$$

$$\begin{aligned} p(\mathring{\mathbf{X}}) = p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0})) &\times \prod_{i \neq \{i_0,\ldots,i_c\}} P(X_i|\mathbf{Pa}(X_i)) \\ &\times \prod_{j=1}^{c} P(X_{i_j}| \bigcup_{h=1}^{j} (\mathbf{Pa}(X_{i_h}) \backslash \{\mathring{X}_{i_0}\}) \cup \mathbf{Pa}(\mathring{X}_{i_0})), \end{aligned} \tag{7.11}$$

where $\mathbf{MB}(\mathring{X}_{i_0})$ is the Markov blanket of $\mathring{X}_{i_0}$ in $\mathcal{G}$: note that, in the last expression of $p(\mathring{\mathbf{X}})$, only the first term involves $\mathring{X}_{i_0}$, hence all the other CPTs are well defined. As a side effect, we can clearly see in the chain rule given in Equation (7.11) that X_{i_0} depends only to its Markov Blanket (in the network after arcs reversal operations). As a matter of fact, the discretization of $\mathring{X}_{i_0}$ can be optimized by taken into account only $p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0}))$, since none of the other terms is related to $\mathring{X}_{i_0}$. In other words, the discretization of $\mathring{X}_{i_0}$, after Shachter's arc reversals, depends only on the discretization of its new parents, since by conditioning on those nodes, $\mathring{X}_{i_0}$ becomes independent from the rest of the variables.

Theorem 7.2. *Let $\mathring{X}_{i_0} \in \mathcal{X}$ be a variable to be discretized in order to optimize $p(\mathring{\mathbf{X}})$. Then the value of $\mathbf{\Theta}$ for this optimal discretization is obtained as:*

$$\mathbf{\Theta} = \underset{\mathbf{\Theta}}{Argmax}\ p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0})). \tag{7.12}$$

From Theorem 7.2 and Equation (7.11), we can conclude that only the Markov blanket of the variable $\mathring{X}_{i_0}$ should be considered when making the discretization of the latter. By taking into account this property, Algorithm 10 is then replaced by the version shown in Algorithm 11.

Algorithm 11: One-variable discretization new architecture.

```
   Input: a database $\mathring{\mathcal{D}}$, a graph $\mathcal{G}$, a scoring function $sc$ on discrete variables
   Output: a discretization $\mathcal{F}$
1  repeat
2      $i_0 \leftarrow$ Select an element in $\{l+1,\dots,n\}$
3      Discretize $\mathring{X}_{i_0}$ given $\mathcal{G}$ and $\mathbf{MB}(\mathring{X}_{i_0})$
4  until stopping condition;
```

Once the architecture of the discretization approach given $\mathcal{G}$ is defined, it remains for us to find an efficient strategy to determine the optimal parameters associated to the untruncated mixture model corresponding to $p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0}))$. The EM algorithm is known to be efficient for solving such a task.

In the following, we start by explaining the most common principles of the EM algorithm that will be used to perform the estimation of the parameters of the mixture model. Next, we provide the details and justify the correctness of the mixture parameters estimation using the EM algorithm. Then, we discuss the way by which the set of cut points (of the different intervals) are identified in both truncated and untruncated mixture model contexts.

7.2.4.1 EM algorithm preliminaries

When the available data do not allow to perform parameters estimation and/or the expression of the likelihood cannot be analytically maximized, the expectation maximization (EM) algorithm can be a solution. This algorithm aims to address the parameters estimation task when this impossibility is due to the presence of hidden (or missing) observations in the data. Proposed by [DLR77], the EM algorithm estimates the parameters of a model with an iterative process by starting from some initial parameter values, and then performing a set of iterations characterized by the following two steps:

- **Expectation (E-step)**: uses the current parameters values to complete the data and computes the expected value of the log-likelihood measure.
- **Maximization (M-step)**: treats the completed data as if they were completely observed, and then learns the new parameters values by maximizing the log-likelihood function.

Before giving the details of each of the steps composing the EM iterations, let us mention the inequality property that characterizes the log-likelihood, which we try to maximize within the EM iterations. For every $m \in \{1, \dots, N\}$, let $Z^{(m)}$ be a hidden discrete random variable whose domain is $\{z_0^{(m)}, \dots, z_g^{(m)}\}$. Variable $Z^{(m)}$ takes value $z_k^{(m)}$ if and only if the mth observation $\mathring{x}_{i_0}^{(m)}$ of variable $\mathring{X}_{i_0}$ in database $\mathring{\mathcal{D}}$ is believed to be part of the kth component (or cluster) represented by the hidden variable $Z^{(m)}$. The completed log-likelihood of $\mathring{X}_{i_0}$ taking into account the hidden variables $Z^{(m)}$ and parameters $\mathbf{\Theta}$ is given as follows:

$$\begin{aligned} LL(\mathring{X}_{i_0}|\mathbf{\Theta}) &= \sum_{m=1}^{N} \log \left(\sum_{k=0}^{g} p(\mathring{x}_{i_0}^{(m)}, z_k^{(m)}|\mathbf{\Theta}) \right) \\ &= \sum_{m=1}^{N} \log \left(\sum_{k=0}^{g} P(z_k^{(m)}|\mathring{x}_{i_0}^{(m)}, \mathbf{\Theta}) \frac{p(\mathring{x}_{i_0}^{(m)}, z_k^{(m)}|\mathbf{\Theta})}{P(z_k^{(m)}|\mathring{x}_{i_0}^{(m)}, \mathbf{\Theta})} \right) \\ &\geq \sum_{m=1}^{N} \sum_{k=0}^{g} P(z_k^{(m)}|\mathring{x}_{i_0}^{(m)}, \mathbf{\Theta}) \log \left(\frac{p(\mathring{x}_{i_0}^{(m)}, z_k^{(m)}|\mathbf{\Theta})}{P(z_k^{(m)}|\mathring{x}_{i_0}^{(m)}, \mathbf{\Theta})} \right) \end{aligned} \tag{7.13}$$

where $\mathring{x}_{i_0}^{(m)}$ represents the observed value of $\mathring{X}_{i_0}$ in the mth record of $\mathring{\mathcal{D}}$ and $P(z_k^{(m)}|\mathring{x}_{i_0}^{(m)}, \mathbf{\Theta})$ represents the probability that the kth component of $Z^{(m)}$ obtains given that $\mathring{x}_{i_0}^{(m)}$ has been observed. As this is a probability, we have of course that $\sum_{k=0}^{g} P(z_k^{(m)}|\mathring{x}_{i_0}^{(m)}, \mathbf{\Theta}) = 1$. The inequality shown by Equation (7.13) is deduced from the well-known Jensen's inequality: for any $\lambda_0, \dots, \lambda_g \geq 0$ with $\sum_{k=0}^{g} \lambda_k = 1$, and any tuple $(y_0, \dots, y_g)$ of positive numbers, we have that:

$$\log \left(\sum_{k=0}^{g} \lambda_k y_k \right) \geq \sum_{k=0}^{g} \lambda_k \log(y_k). \tag{7.14}$$

Therefore, instead of maximizing $\mathbf{\Theta}$ using the exact log-likelihood expression $LL(X|\mathbf{\Theta})$ expressed by the log of a summation[1], the EM algorithm relies on the maximization of

[1] The log of a summation is difficult to be maximized in an analytic way.

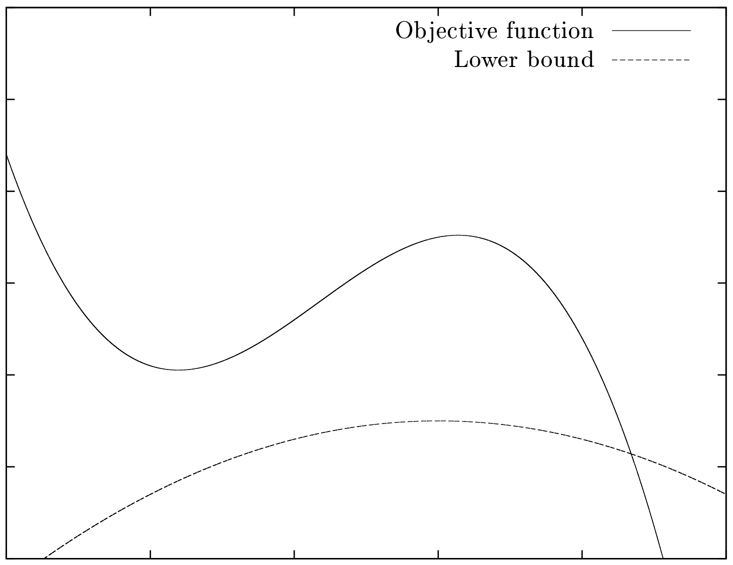


FIGURE 7.8: The maximization of log-likelihood using a lower bound approximation.

a lower-bound function (as can be seen in the example of Figure 7.8) using the following steps:

- E-step :

$$\begin{aligned} Q_m^{t+1}(z_k^{(m)}) &= P(z_k^{(m)}|\mathring{x}_{i_0}^{(m)}, \mathbf{\Theta}^t), \forall m \in \{1, ..., N\} \\ LL^{t+1}(\mathring{X}_{i_0}|\mathbf{\Theta}^t) &= \sum_{m=1}^{N}\sum_{k=0}^{g} Q_m^{t+1}(z_k^{(m)}) \log\left(\frac{P(\mathring{x}_{i_0}^{(m)}, z_k^{(m)}|\mathbf{\Theta}^t)}{Q_m^{t+1}(z_k^{(m)})}\right) \end{aligned} \tag{7.15}$$

- M-step:

$$\mathbf{\Theta}^{t+1} = \underset{\mathbf{\Theta}}{\text{Argmax}}\, LL^{t+1}(\mathring{X}_{i_0}|\mathbf{\Theta}) \tag{7.16}$$

To solve $\text{Argmax}_{\mathbf{\Theta}}\, LL(\mathring{X}_{i_0}|\mathbf{\Theta})$, EM [DLR77] iteratively alternates expectations (E-step) and maximizations (M-step) until the convergence toward a local maximum, which is guaranteed to correspond to the Argmax we look for due to the concavity of the log-likelihood function (as seen in Figure 7.9). Remind that the iterative process of EM starts by an initial parameters assignment $\mathbf{\Theta}^0$ generated randomly (or by using some other methods). The algorithm then performs the computation of the probabilities of missing observations by using the current parameters (E-step). In the M-step a new set of parameters are estimated by MLE over the completed (or weighted) data. Then, the resulting new parameters are used as a new starting point for the next iteration, and so on.

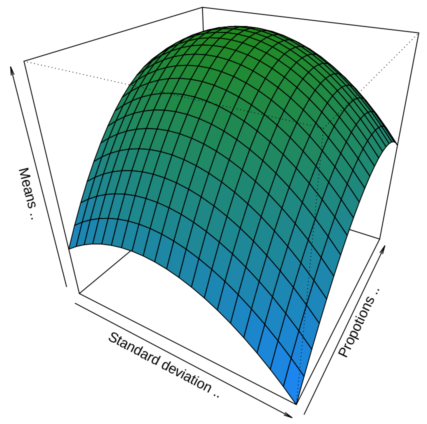


FIGURE 7.9: The concavity of the log-likelihood function according to $\boldsymbol{\Theta}$.

7.2.5 Gaussian Mixture model-based discretization

7.2.5.1 Parameters estimation with EM algorithm

Let q_{i_0} represent the (finite) number of values of $\mathbf{MB}(\mathring{X}_{i_0})$. For simplicity, we will denote by $\{1, \ldots, q_{i_0}\}$ the set of values of the joint discrete random variable $\mathbf{MB}(\mathring{X}_{i_0})$. Let g denote the number of cut points in the discretization and let $\{\mathcal{N}(\mu_k, \sigma_k) : k \in \{0, \ldots, g\}\}$ be the corresponding set of Gaussians. Then, the density function of an observation $\mathring{x}_{i_0}$ given its Markov blanket values is expressed as follows:

$$p(\mathring{X}_{i_0} = \mathring{x}_{i_0} | \mathbf{MB}(\mathring{X}_{i_0}) = j) = \sum_{k=0}^{g} \pi_{jk} f(\mathring{x}_{i_0} | \boldsymbol{\theta}_k) \quad \forall\, j \in \{1, \ldots, q_{i_0}\}. \tag{7.17}$$

Remember that each value of $\mathbf{MB}(\mathring{X}_{i_0})$ (which corresponds to the parents of $\mathring{X}_{i_0}$ after Shachter's transformations) induces its own set of weights $\{\pi_{j0}, \ldots, \pi_{jg}\}$ as shown in the example of Figure 7.3 and Equation (7.8). As discussed earlier, the joint optimization of the truncated mixture parameters as described in Equation (7.17) is really hard to perform so we propose an approximate estimation of the density $p(\mathring{X}_{i_0} = \mathring{x}_{i_0} | \mathbf{MB}(\mathring{X}_{i_0}) = j)$ using an untruncated Gaussian mixture model. First, we shall focus our discussion on the estimation of the parameters of this untruncated Gaussians mixture model using EM. Then we will show how this estimation can be used to approximate the parameters and cutpoints for the truncated Gaussians mixtures. So, for the moment, Equation (7.17) is replaced by the following expression:

$$p(\mathring{X}_{i_0} = \mathring{x}_{i_0} | \mathbf{MB}(\mathring{X}_{i_0}) = j) \approx \sum_{k=0}^{g} \pi_{jk} h(\mathring{x}_{i_0} | \boldsymbol{\theta}_k) \quad \forall\, j \in \{1, \ldots, q_{i_0}\}, \tag{7.18}$$

As can be seen, the only difference between Equations (7.17) and (7.18) lies in the density function they use to represent the different components in the mixture model, i.e., $f(.|\boldsymbol{\theta}_k)$ and $h(.|\boldsymbol{\theta}_k)$ represent respectively the truncated and untruncated Gaussians.

By assuming that data in $\mathring{\mathcal{D}}$ are i.i.d., the log-likelihood of $\mathring{\mathcal{D}}$ given $\boldsymbol{\Theta} = \bigcup_{k=0}^{g}(\bigcup_{j=1}^{q_{i_0}}\{\pi_{jk}\} \cup \{\boldsymbol{\theta}_k\})$ is equal to:

$$LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) = \sum_{m=1}^{N} \log p(\mathring{X}_{i_0} = \mathring{x}_{i_0}^{(m)}|\mathbf{MB}(\mathring{x}_{i_0})^{(m)}, \boldsymbol{\Theta}), \tag{7.19}$$

By replacing the probability in the above equation by its appropriate expression, we obtain:

$$LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) = \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \log \left[\sum_{k=0}^{g} \pi_{jk} h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)\right] \tag{7.20}$$

It must be emphasized that in this estimation step, we take into account weights π_{jk} only for the records in the database that correspond to $\mathbf{MB}(\mathring{x}_{i_0})^{(m)} = j$.

Let $Q_m^{t+1}(z^{(m)}) = P(z^{(m)}|\mathring{x}_{i_0}^{(m)}, \boldsymbol{\Theta}^t)$, i.e., $Q_m^{t+1}(z^{(m)})$ represents the distribution that, at the $t+1$th step of the algorithm, $\mathring{x}_{i_0}^{(m)}$ is believed to have been generated by such and such untruncated Gaussian. The term $Q_m^{t+1}(z_k^{(m)})$ is calculated by the following equation:

$$Q_m^{t+1}(z_k^{(m)}) = \frac{\pi_{jk}^t h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k^t)}{\sum_{k'=0}^{g} \pi_{jk'}^t h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_{k'}^t)}, \tag{7.21}$$

As discussed in section 7.2.4.1, instead of maximizing $LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta})$, EM tries to maximize a lower bound function which results from the concavity property of the log-likelihood function and Jensen's inequality. The lower bound function in our case is given as follows:

$$\begin{aligned} LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) &= \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \log \left[\sum_{k=0}^{g} Q_m(z_k^{(m)}) \frac{\pi_{jk} h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)}{Q_m(z_k^{(m)})}\right] \\ &\geq \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k=0}^{g} Q_m(z_k^{(m)}) \log \left[\frac{\pi_{jk} h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)}{Q_m(z_k^{(m)})}\right] \end{aligned} \tag{7.22}$$

Given the above inequality, choosing a new parameter set $\boldsymbol{\Theta}$ that maximizes the lower bound function, will automatically enhance the original log-likelihood.

After identifying the objective function to be optimized during the estimation process, EM can be used to find the appropriate parameters of the untruncated mixture model given $\mathcal{G}$. The pseudocode of the resulting EM algorithm is described in Algorithm 12.

It must be noted that in Algorithm 12, only the M-step can be computationally intensive. Fortunately, here, we can derive in closed-form the optimal values of Line 4 (which is not the case for the truncated mixture model, as we shall see hereafter). This step consists in

Algorithm 12: The EM algorithm.

```
  Input: a database $\mathring{\mathcal{D}}$, a number $g$ of cut points, DAG $\mathcal{G}$
  Output: an optimal set of parameters $\boldsymbol{\Theta}$
1 Select (randomly) an initial value $\boldsymbol{\Theta}^0$
2 repeat
      // E-step (expectation)
3     $Q_m^{t+1}(z^{(m)}) \leftarrow P(z^{(m)}|\mathring{x}_{i_0}^{(m)}, \boldsymbol{\Theta}^t) \quad \forall\, m \in \{1,\dots,N\}$
      // M-step (maximization)
4     $\boldsymbol{\Theta}^{t+1} \leftarrow \operatorname{Argmax}_{\boldsymbol{\Theta}} \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k=0}^{g} Q_m^{t+1}(z_k^{(m)}) \log\left[\frac{\pi_{jk} h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)}{Q_m^{t+1}(z_k^{(m)})}\right]$
5 until convergence;
```

searching the best maximization of the log-likelihood function w.r.t. the mixture model parameters.

Proposition 7.3. *At the E-step, probability* $Q_m^{t+1}(z_k^{(m)}) = \dfrac{\pi_{jk}^t h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k^t)}{\sum_{k'=0}^{g} \pi_{jk'}^t h(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_{k'}^t)}$, *where* π_{jk}^t *and* $\boldsymbol{\theta}_k^t$ *are weights, means and variances in* $\boldsymbol{\Theta}^t$. *The optimal parameters of the M-step are respectively:*

$$\pi_{jk}^{t+1} = \frac{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})}{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k'=0}^{g} Q_m^{t+1}(z_{k'}^{(m)})},$$

$$\mu_k^{t+1} = \frac{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})\mathring{x}_{i_0}^{(m)}}{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})} \qquad \sigma_k^{t+1} = \sqrt{\frac{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})(\mathring{x}_{i_0}^{(m)} - \mu_{z_k}^{t+1})^2}{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})}}.$$

Proof. Let us delve into the details of the mentioned update equations associated to each parameter of the mixture model. We begin by the weight π_{jk} for each kth component given the parents modality j. We should keep in mind that parameter π_{jk} needs a little more work since its values are constrained to being positive and adding up to 1. Therefore, we get:

$$\frac{\partial LL(\mathring{D}|\boldsymbol{\Theta}^t)}{\partial \pi_{jk}} = \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \frac{Q_m^{t+1}(z_k^{(m)})}{\pi_{jk}} - \frac{Q_m^{t+1}(z_g^{(m)})}{\left(1 - \sum_{k=0}^{g-1} \pi_{jk}\right)} = 0 \tag{7.23}$$

From this result, we can immediately conclude that Equation (7.23) is equal to 0 when $\pi_{j,k}$ verifies:

$$\frac{Q_m^{t+1}(z_k^{(m)})}{\pi_{jk}} = \frac{Q_m^{t+1}(z_g^{(m)})}{\left(1 - \sum_{k=0}^{g-1} \pi_{jk}\right)} \quad \forall k \in \{0, ..., g-1\}.$$

Therefore:

$$\frac{Q_m^{t+1}(z_1^{(m)})}{\pi_{j1}} = \cdots = \frac{Q_m^{t+1}(z_g^{(m)})}{\pi_{jg}} = \frac{\sum_{k=0}^{g} Q_m^{t+1}(z_k^{(m)})}{\sum_{k=0}^{g} \pi_{jk}} = \sum_{k=0}^{g} Q_m^{t+1}(z_k^{(m)}).$$

Substituting these equivalences into Equation (7.23) gives us the following update equation:

$$\pi_{jk}^{t+1} = \frac{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})}{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k=1}^{g} Q_m^{t+1}(z_k^{(m)})}.$$

Concerning the parameters μ_k, its optimal value in the M-step can be determined by the cancellation of the following derivative equation:

$$\begin{aligned} \frac{\partial LL(\mathring{D}|\mathbf{\Theta}^t)}{\partial \mu_k} &= -\frac{1}{\sigma_k^2} \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})(-\mathring{x}_{i_0}^{(m)} + \mu_k) = 0 \\ &= -\frac{1}{\sigma_q^2} \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})\mu_k - Q_m^{t+1}(z_k^{(m)})\mathring{x}_{i_0}^{(m)} = 0. \end{aligned} \tag{7.24}$$

This leads to the following equality:

$$\sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})\mu_k = \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})\mathring{x}_{i_0}^{(m)}. \tag{7.25}$$

Substituting this result into Equation (7.24) gives the following μ_k update equation:

$$\mu_k^{t+1} = \frac{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})\mathring{x}_{i_0}^{(m)}}{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})}.$$

Finally, the optimal σ_k^{t+1} value is obtained by setting the following first order partial derivation to zero:

$$\begin{aligned} \frac{\partial LL(\mathring{D}|\mathbf{\Theta}^t)}{\partial \sigma_k} &= \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)}) \left[-\frac{1}{\sigma_k} + \frac{1}{\sigma_k^3}(\mathring{x}_{i_0}^{(m)} - \mu_k)^2\right] = 0 \\ &= \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)}) \left[-1 + \frac{1}{\sigma_k^2}(\mathring{x}_{i_0}^{(m)} - \mu_k)^2\right] = 0. \end{aligned} \tag{7.26}$$

From this result, we get:

$$\frac{1}{\sigma_k^2} \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})(\mathring{x}_{i_0}^{(m)} - \mu_k)^2 = \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)}).$$

Substituting this result into Equation (7.26) gives the following σ_k update equation:

$$\sigma_k^{t+1} = \sqrt{\frac{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})(\mathring{x}_{i_0}^{(m)} - \mu_k^{t+1})^2}{\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})}}.$$

Hence the update equations of Proposition 7.3. □

In this version of our approach, the use of the Algorithm 12 and the update formulas shown in Proposition 7.3 together allow to estimate efficiently the parameters of the mixture model. For instance, Figure 7.1 depicts an example of a mixture Gaussian model where the optimal parameters are given as follows: $\mathcal{N}_0(\pi_0 = 0.21, \mu_0 = -6, \sigma_0 = 1.2)$, $\mathcal{N}_1(\pi_1 = 0.23, \mu_1 = -2, \sigma_1 = 1.25)$, $\mathcal{N}_2(\pi_2 = 0.2, \mu_2 = 1.5, \sigma_2 = 0.85)$ and $\mathcal{N}_3(\pi_3 = 0.36, \mu_3 = 5, \sigma_3 = 1.5)$.

However, as discussed earlier, our ultimate goal in this phase does not consist in computing only the mixture Gaussian parameters, but to approximate the truncated Gaussian model (our initial assumption) and to exploit it to make the discretization of each continuous variable $\mathring{X}_{i_0}$. Therefore, it remains to determine the best cut points $\{d_1, \dots, d_g\}$ (or the discretization) from our estimation of the mixture model. Let us see how this task can be performed.

7.2.5.2 Determination of the Cut Points

As mentioned at the end of Subsection 7.2.2, each Gaussian $\mathcal{N}(\mu_k, \sigma_k)$ is associated with an interval $[d_k, d_{k+1})$[2] and the parts of the Gaussian outside the interval can be considered as a loss of information. The optimal set of cut points $\widehat{\mathcal{T}} = \{\hat{t}_1, \dots, \hat{t}_g\}$ is thus that which minimizes this loss. In other words, it is equal to:

$$\widehat{\mathcal{T}} = \underset{\{d_1,\dots,d_g\}}{\text{Argmin}} \sum_{k=0}^{g} \int_{d_k}^{+\infty} h(x|\boldsymbol{\theta}_{k-1})dx + \int_{-\infty}^{d_k} h(x|\boldsymbol{\theta}_k)dx,$$

where $\boldsymbol{\theta}_k$ represents pairs (μ_k, σ_k) and d_0 and d_{g+1} are set to $-\infty$ and $+\infty$ respectively. As each Gaussian $\mathcal{N}(\mu_k, \sigma_k)$ is associated with interval $[d_k, d_{k+1})$, we can assume that $\hat{t}_k \in [\mu_{k-1}, \mu_k)$, for all k. Therefore:

$$\widehat{\mathcal{T}} = \left\{ \underset{d_k \in [\mu_{k-1}, \mu_k)}{\text{Argmin}} \int_{d_k}^{+\infty} h(x|\boldsymbol{\theta}_{k-1})dx + \int_{-\infty}^{d_k} h(x|\boldsymbol{\theta}_k)dx : k \in \{1, \dots, g\} \right\}. \tag{7.27}$$

All the $\hat{t}_k$ can thus be determined independently. In addition, as shown below, their values are the solution of a quadratic equation:

Proposition 7.4. *Let $u(d_k)$ represent the sum of the integrals in Equation (7.27). Let α_k be a solution (if any) within interval (μ_{k-1}, μ_k) of the quadratic equation in d_k:*

$$d_k^2 \left(\frac{1}{\sigma_{k-1}^2} - \frac{1}{\sigma_k^2} \right) + 2d_k \left(\frac{\mu_k}{\sigma_k^2} - \frac{\mu_{k-1}}{\sigma_{k-1}^2} \right) + \left(\frac{\mu_{k-1}^2}{\sigma_{k-1}^2} - \frac{\mu_k^2}{\sigma_k^2} - 2log \frac{\sigma_k}{\sigma_{k-1}} \right) = 0. \tag{7.28}$$

Then $\hat{t}_k$ is, among $\{\mu_{k-1}, \mu_k, \alpha_k\}$, the element with the highest value of $u(\cdot)$ (which can be quickly approximated using a table of the Normal distribution).

[2]Without loss of generality, we consider here that the the μ_k's resulting from the EM algorithm are sorted by increasing order.

Proof. Let $g(\cdot)$ and $k(\cdot)$ be two functions such that $\partial g(x)/\partial x = h(x|\boldsymbol{\theta}_{k-1})$ and $\partial k(x)/\partial x = h(x|\boldsymbol{\theta}_k)$. Then:

$$\begin{aligned}\hat{d}_k &= \operatorname*{Argmin}_{d_k\in[\mu_{k-1},\mu_k)} u(d_k) = \operatorname*{Argmin}_{d_k\in[\mu_{k-1},\mu_k)} \int_{d_k}^{+\infty} \frac{\partial g(x)}{\partial x}dx + \int_{-\infty}^{d_k} \frac{\partial k(x)}{\partial x}dx \\ &= \operatorname*{Argmin}_{d_k\in[\mu_{k-1},\mu_k)} -g(d_k) + k(d_k) + \lim_{d\to+\infty}[g(d) - k(-d)].\end{aligned}$$

Let us relax the optimization problem and try to find the Argmin over $\mathbb{R}$. Then the min is obtained when $\partial u(d_k)/\partial d_k = 0$ or, equivalently, when $\partial(-g(d_k) + k(d_k))/\partial d_k = -h(d_k|\boldsymbol{\theta}_{k-1}) + h(d_k|\boldsymbol{\theta}_k) = 0$. Since $h(\cdot|\boldsymbol{\theta})$ represents the density of the Normal distribution of parameters $\boldsymbol{\theta}$, this is equivalent to:

$$-\frac{1}{\sqrt{2\pi}\sigma_{k-1}}\exp\left(-\frac{1}{2}\left(\frac{d_k-\mu_{k-1}}{\sigma_{k-1}}\right)^2\right) + \frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{1}{2}\left(\frac{d_k-\mu_k}{\sigma_k}\right)^2\right) = 0,$$

or, equivalently:

$$\frac{\sigma_k}{\sigma_{k-1}} = \frac{\exp\left[-\frac{1}{2}\left(\frac{d_k-\mu_k}{\sigma_k}\right)^2\right]}{\exp\left[-\frac{1}{2}\left(\frac{d_k-\mu_{k-1}}{\sigma_{k-1}}\right)^2\right]} = \exp\left[\frac{1}{2}\left(\frac{d_k-\mu_{k-1}}{\sigma_{k-1}}\right)^2 - \frac{1}{2}\left(\frac{d_k-\mu_k}{\sigma_k}\right)^2\right],$$

which, by a log transformation, is equivalent to:

$$2\log\frac{\sigma_k}{\sigma_{k-1}} = \frac{d_k^2}{\sigma_{k-1}^2} - \frac{2\mu_{k-1}d_k}{\sigma_{k-1}^2} + \frac{\mu_{k-1}^2}{\sigma_{k-1}^2} - \frac{d_k^2}{\sigma_k^2} + \frac{2\mu_k d_k}{\sigma_k^2} - \frac{\mu_k^2}{\sigma_k^2}.$$

This corresponds precisely to Equation (7.28). So, to summarize, if the optimal solution lies inside interval (μ_{k-1}, μ_k), then it satisfies Equation (7.28). Otherwise, either $u(d_k)$ is strictly increasing or strictly decreasing within (μ_{k-1}, μ_k), which implies that the optimal solution for $\hat{t}_k$ is either μ_{k-1} or μ_k, which completes the proof. □

For a better understanding of the use of the previous proposition, let us apply it on a concrete example which aims to estimate the cut point associated to the first two Gaussians $\mathcal{N}_0(-6, 1.2)$ and $\mathcal{N}_1(-2, 1.25)$ shown in Figure 7.1. By replacing the variables corresponding to parameters μ and σ in the above quadratic equation by their appropriate values we get:

$$d_k^2(0.05) + d_k(5.77) + 22.35 = 0$$

The square-root part of the previous quadratic equation is as follows:

$$\begin{aligned}\Delta &= b^2 - 4ac \\ &= (5.77)^2 - 4(0.05 * 22.35) \\ &= 28.462\end{aligned} \tag{7.29}$$

Given that $\Delta > 0$, then the previous quadratic equation can be resolved in two ways:

$$\begin{aligned} d_{k_1} &= \frac{-b-\sqrt{\Delta}}{2a} \quad \mathit{or} \quad d_{k_2} &= \frac{-b+\sqrt{\Delta}}{2a} \\ &= -102.01 \quad \mathit{or} &= -4.02 \end{aligned} \tag{7.30}$$

As discussed earlier, the computed cut point $\hat{d}_k$ between the considered Gaussians, must lies inside interval $[-6, -2]$. Therefore, the chosen solution for the considered quadratic equation will be $d_k = -4.02$, which fits exactly with the cut point d_1 depicted in Figure 7.5. The same principle is followed to make the estimation of the other cut points (d_2 and d_3).

As can be seen in Figure 7.5, the resulting mixture model (after cut points estimations) is a good approximation of the original truncated mixture model which is given in Figure 7.1. Note that in this example, the resulting discretization schema is:

$$f(\mathring{x}_{i_0}) = \begin{cases} 0 & \text{if} \quad \mathring{x}_{i_0} < -4.02, \\ 1 & \text{if} \quad -4.02 \leq \mathring{x}_{i_0} < -0.045, \\ 2 & \text{if} \quad -0.045 \leq \mathring{x}_{i_0} < 2.97, \\ 3 & \text{if} \quad \mathring{x}_{i_0} \geq 2.97. \end{cases}$$

7.2.5.3 Score and Number of Cut Points

To complete the description of the algorithm, there remains to determine the number of cut points (or the number of components in the mixture model). Of course, the higher the number of cut points, the higher the likelihood but the lower the compactness of the representation[3]. To reach of good trade-off, we simply propose to exploit the penalty functions included into the score used for the evaluation of different BN structures (see Line 5 of Algorithm 9). Here, we use the BIC score [S+78], which can be locally expressed by the following expression:

$$BIC(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0})) = LL(\mathring{\mathcal{D}}|\mathbf{\Theta}) - \frac{|\mathbf{\Theta}|}{2}\log(N) \tag{7.31}$$

where $LL(\mathring{\mathcal{D}}|\mathbf{\Theta})$ is the log-likelihood with the parameters estimated by EM, given the current structure $\mathcal{G}$. $|\mathbf{\Theta}|$ represents the number of parameters, i.e., $|\mathbf{\Theta}| = q_{i_0} \times g + 2 \times (g+1)$: the first and second terms correspond respectively to the number of parameters π_{jk} and of (μ_k, σ_k) needed to encode the conditional distributions (recall that there are $g+1$ Gaussians and q_{i_0} represents the domain size of $\mathbf{MB}(\mathring{X}_{i_0})$. Now, the best number of cut points is simply that which optimizes equation (7.31).

[3]Note that $N-1$ cut points leads to a zero log-likelihood.

7.2.5.4 Summary

To summarize, for a given network $\mathcal{G}$ and a set of continuous variables $\mathring{\mathbf{X}}$, our discretization approach lies in the joint use of a tailored version of EM (see Algorithm 12) and our proposed cut points estimation method (given by Proposition 7.4). As discussed earlier, to alleviate the problem of the strong overhead that the joint discretization of all the continuous variables may incur to the learning process, discretizations are performed one continuous variable at a time. In this direction, the discretization of $\mathring{X}_{i_0}$ in $\mathcal{G}$ is carried out by considering only its interactions with the set of variables representing its Markov blanket, i.e., $\mathring{X}_{i_0}$ is discretized by maximizing the log-likelihood of $p(\mathring{X}_{i_0}|\mathbf{MB}(\mathring{X}_{i_0}))$. Given the previous steps, one important question may arise about the way by which each new discretization will be propagated (or communicated) to the rest of continuous variables schema in $\mathcal{G}$. To do so, our algorithm follows the same propagation principle as in [MC98, FG96], where, each time we change the discretization of $\mathring{X}_{i_0}$, the discretizations of the variables of its Markov blanket are also subsequently recomputed. Figure 7.10 depicts a schematization example of the propagation trajectories (or the impact) of each new discretization in a network $\mathcal{G}$ containing 7 continuous random variables. In this example, changing for instance the discretization of node C may have an impact on the incoming discretizations of variables composing its Markov blanket $\mathbf{MB}(C) = \{A, B, D, E\}$, i.e., this might lead to reconsider the discretization of these variables (as mentioned by the dotted edges between C and $\mathbf{MB}(C)$ in Figure 7.10).

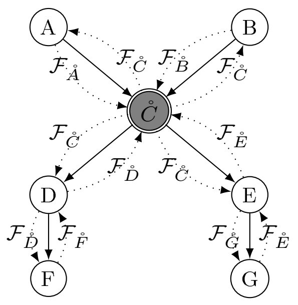

A
B
$\mathcal{F}_{\mathring{A}}$
$\mathcal{F}_{\mathring{C}}$
$\mathcal{F}_{\mathring{B}}$
$\mathcal{F}_{\mathring{C}}$
$\mathring{C}$
$\mathcal{F}_{\mathring{C}}$
$\mathcal{F}_{\mathring{E}}$
$\mathcal{F}_{\mathring{D}}$
$\mathcal{F}_{\mathring{C}}$
D
E
$\mathcal{F}_{\mathring{D}}$
$\mathcal{F}_{\mathring{F}}$
$\mathcal{F}_{\mathring{G}}$
$\mathcal{F}_{\mathring{E}}$
F
G

FIGURE 7.10: The propagation trajectories of the discretization schema in $\mathcal{G}$ (the discretization of variables A, B, D, E, F, G are fixed while discretizing variable C)

Roughly speaking, the principle of our multivariate discretization approach goes as follows: we begin by performing a univariate discretization for all the continuous variables, starting with an empty structure $\mathcal{G}_0$. This results in a discrete dataset $\mathcal{D}$, hence we can use standard learning algorithms to discover a new network $\mathcal{G}_1$ (not empty). At this step, the network $\mathcal{G}_1$ is fixed and we select a variable $\mathring{X}_{i_0}$ and search for its optimal discretization while treating all the other variables as discrete (i.e., we use their discretization computed in the previous iteration). This can be done by using the approach previously described (EM and cut points estimation). We repeat this step for each continuous variable until no discretization schema leads to an improvement of the BIC scoring function. Given this approach, it is obvious to deduce that such an optimization approach must converge since, as seen earlier, the EM algorithm guarantees that at each iteration the log-likelihood $P(\mathcal{F}|\mathcal{G}_1, \mathring{D})$ is increased. Hence it must stop at some stage. After completing the discretization of all the variables in $\mathcal{G}_1$, we repeat the learning of a new structure $\mathcal{G}_2$ using the resulting discretization and so on.

It must be emphasized that our algorithm decides to continue (stop criterion) the discretization process if and only if $\mathcal{G}_t$ (the resulting structure from the tth discretization) is different from $\mathcal{G}_{t-1}$, i.e., it means that structure $\mathcal{G}_t$ entails a different set of conditional independences than those encoded by $\mathcal{G}_{t-1}$ (note that if $\mathcal{G}_{t-1} \equiv \mathcal{G}_t$ the resulting discretization will be equivalent).

7.2.6 Truncated mixture Gaussian model based discretization

In this section, we present an alternative method which aims to identify the best discretization $\mathcal{F}$ by directly estimating the truncated mixture model parameters (without performing any additional intermediate steps). Similarly to the previous approach, the current one addresses the model parameters estimation by taking into account the conditional independences entailed by $\mathcal{G}$ (as explained in Section 7.2.4) and it relies on the use of the EM algorithm, which is known to be efficient for addressing such a difficult optimization task.

Before explaining the details of our estimation method, it must be underlined that each truncated Gaussian in the mixture model (to be estimated) can be defined using the following two steps: a classical normal distribution with parameters μ_k and σ_k are chosen initially, and next, a truncation interval $[d_k,\, d_{k+1})$ is assigned to the considered Gaussian. By adding the truncation, the density function of the classical normal distribution is modified by assigning a 0 density for all the values outside the fixed interval, and uniformly scaling the values inside the interval in order to ensure that $f(.|\boldsymbol{\theta}_k)$ integrates to unity (a valid density function).

As seen earlier in Equations (7.6) and (7.18), the only difference between the truncated and the untruncated mixture distributions is the normalization factor which, as we shall see hereafter, does not complicate the E-step of the EM algorithm, but can be an issue for the maximization of certain parameters of the mixture model during the M-step: except the update equation related to the parameter π_{ij}, the other ones cannot be computed analytically.

It should be noted that despite the strong overhead and the complexity that the truncated mixture model parameters estimation incurs to the multivariate discretization process (since it cannot be expressed using closed form-formula due to presence of the integral), we think that the result returned by our new method can be of great explanatory importance because:

- when estimating discretization the function $\mathcal{F}$ given $\mathcal{G}$, the joint optimization of the truncated mixture model parameters (means, variances, weights, intervals) guarantees the maximization of the likelihood defined as $\text{Argmax}_{\mathcal{F}}\, P(\mathcal{F}|\mathcal{G}, \mathring{D})$, hence leading to the optimization of the global initial objective function defined as $P(\mathcal{G}, \mathcal{F}|\mathring{\mathcal{D}})$.

- the returned results can also be used as a validation approach that allows us to check the quality of our first discretization version, which, as discussed earlier, produces only an approximate estimation of the original truncated mixture parameters. Actually, in the previous approach, the optimality of the cut points estimation method w.r.t. likelihood $P(\mathcal{F}|\mathcal{G}, \mathring{D})$ (by solving the quadratic equation) cannot be proved. This can be explained, in particular, by the fact that the cut points are not computed by taking into account the whole dataset $\mathring{\mathcal{D}}$, but only by using some summary in the form of the mean and variance parameters found by EM.

Before discussing the details of our approach, we should keep in mind that several truncation modalities of the Gaussian density function can be considered in our case:

$$\begin{aligned}
(d_k = -\infty) \wedge (d_{k+1} = +\infty) &\Longrightarrow \text{untruncated}\\
(d_k > -\infty) \wedge (d_{k+1} = +\infty) &\Longrightarrow \text{lower truncated}\\
(d_k = -\infty) \wedge (d_{k+1} < +\infty) &\Longrightarrow \text{upper truncated}\\
(d_k > -\infty) \wedge (d_{k+1} < +\infty) &\Longrightarrow \text{complete truncated}
\end{aligned}$$

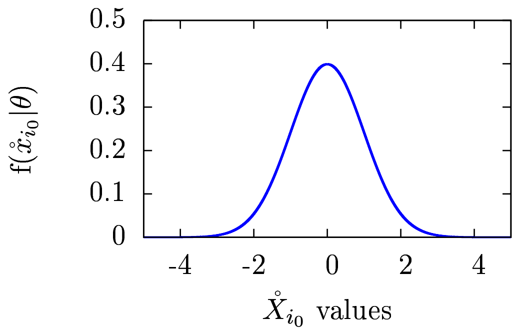
Figure 7.11 shows some examples of the previous truncations where: (a) observations are not truncated $(-\infty, +\infty)$ (b) observations are left censored at -2 by $[-2, +\infty)$ (c) observations are right censored at 2 by $(-\infty, 2]$ (d) observations are censored between -2 and 2 by $[-2, 2]$.

In the rest of this section, we shall focus our discussion on complete truncated Gaussian distribution. Our approach assigns to the left and right sides of respectively the first and last intervals of the discretization schema of $\mathring{X}_{i_0}$ the min and max values taken by the latter, which are denoted hereafter by $min(\mathring{X}_{i_0})$ and $max(\mathring{X}_{i_0})$ (instead of $-\infty$ and $+\infty$). Regarding the truncations of the middle intervals, they are unknown and should be estimated from dataset $\mathring{\mathcal{D}}$ using the MLE method.

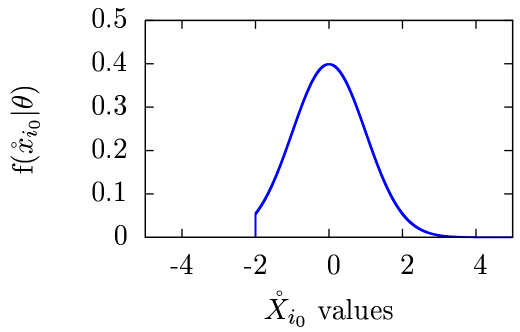
Now, let us delve into the details of truncated mixture parameters estimation using our new variant of the EM algorithm.

7.2.6.1 Truncated mixture model parameters estimating with EM

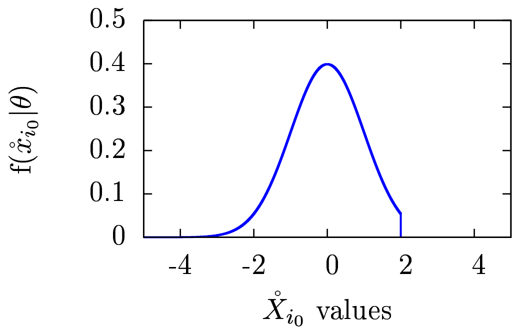
In our approach, we propose to estimate the parameters $\mathbf{\Theta}$ of the truncated mixture model from $\mathring{\mathcal{D}}$ by maximum likelihood. For this purpose, we use EM to provide a good



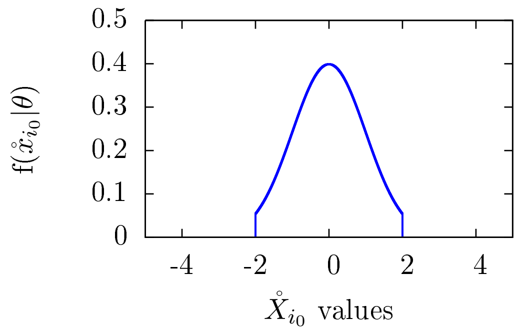
(a) Untruncated Gaussian



(b) Lower truncated Gaussian



(c) Upper truncated Gaussian



(d) Complete truncated Gaussian

FIGURE 7.11: Modalities of truncated Gaussian distribution

approximation of the parameters. For every $m \in \{1, \dots, N\}$, let $Z^{(m)}$ be a hidden discrete random variable whose domain is $\{z_0^{(m)}, \dots, z_g^{(m)}\}$. Variable $Z^{(m)}$ takes value $z_k^{(m)}$ if and only if the mth observation $\mathring{x}_{i_0}^{(m)}$ of variable $\mathring{X}_{i_0}$ in database $\mathring{\mathcal{D}}$ is believed to have been generated by the kth truncated normal distribution. Assuming that data in $\mathring{\mathcal{D}}$ are *i.i.d*, the log-likelihood of $\mathring{\mathcal{D}}$ given $\boldsymbol{\Theta} = \bigcup_{k=0}^{g}\{\bigcup_{j=1}^{q_{i_0}}\{\pi_{jk}\}, \boldsymbol{\theta}_k\}$ (where $\boldsymbol{\theta}_k = (\mu_k, \sigma_k, d_{k-1}, d_k)$) is calculated as follows:

$$LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) = \sum_{m=1}^{N} \log p(\mathring{X}_{i_0} = \mathring{x}_{i_0}^{(m)}|\mathbf{MB}(\mathring{x}_{i_0})^{(m)}, \boldsymbol{\Theta}). \tag{7.32}$$

By replacing $p(\mathring{X}_{i_0} = \mathring{x}_{i_0}^{(m)}|\mathbf{MB}(\mathring{x}_{i_0})^{(m)}, \boldsymbol{\Theta})$ by its corresponding mixture model equation, we obtain:

$$LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) = \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \log \left[\sum_{k=0}^{g} \pi_{jk} f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)\right] \tag{7.33}$$

where the expression of $f(.|\boldsymbol{\theta}_k)$ is given by Equation (7.6).

It must be emphasized that this version of our approach shares the same principle as in the first step of the previous approach (based on the untruncated mixture model), where the goal of the EM algorithm consists in optimizing $LL(\boldsymbol{\Theta}|\mathring{\mathcal{D}})$ by relying on a lower bound given as follows:

$$LL(\mathring{\mathcal{D}}|\mathbf{\Theta}) \geq \sum_{j=1}^{q_{i_0}} \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k=0}^{g} Q_m(z_k^{(m)}) \log \left[\frac{\pi_{jk} f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k)}{Q_m(z_k^{(m)})} \right] \tag{7.34}$$

As seen in Figure 7.8, this lower bound can have a functional form, in principle. In fact, choosing a set of new parameter values to maximize the lower bound function will always result in an improvement over the original objective function (which is the likelihood), i.e., maximizing this optimal lower bound in the M-step of EM will guarantee to take us closer to the maximum of the initial log-likelihood function. Remind that in this approach, we just need to apply the standard EM, considering for weights π_{jk} only the records in the database that correspond to $\mathbf{MB}(\mathring{x}_{i_0}) = j$ (as the previous approach does). For each record of $\mathring{\mathcal{D}}$, $Z^{(m)} = z_k^{(m)}$ if and only if the mth observation $\mathring{x}_{i_0}$ of continuous variable $\mathring{X}_{i_0}$ in the database is believed to have been generated from the kth truncated Gaussian. Let $Q_m^{t+1}(z^{(m)}) = P(z^{(m)}|\mathring{x}_{i_0}, \mathbf{\Theta}^t)$. Having the same kind of formula as in the untruncated mixture model, the term $Q_m^{t+1}(z_k^{(m)})$ is defined by the following equation:

$$Q_m^{t+1}(z_k^{(m)}) = \frac{\pi_{jk}^t f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_k^t)}{\sum_{k'=0}^{g} \pi_{jk'}^t f(\mathring{x}_{i_0}^{(m)}|\boldsymbol{\theta}_{k'}^t)}. \tag{7.35}$$

The main difference between the E-step in the untruncated and the truncated mixture models appears in the way by which the term $Q_m^{t+1}(z_k^{(m)})$ is computed. Since we rely on truncated Gaussian distributions here, Equation (7.35) amounts to restrict the term $Q_m^{t+1}(z_k^{(m)})$ to take only 0 and 1 values, i.e.:

$$Q_m^{t+1}(z_k^{(m)}) = \begin{cases} 1 & \text{if} \quad d_k \leq \mathring{x}_{i_0}^{(m)} < d_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{7.36}$$

Exploiting the above equations, the term $Q_m^{t+1}(z_k^{(m)})$ can be found by simply testing whether $\mathring{x}_{i_0}^{(m)}$ belongs to the interval $[d_k, d_{k+1})$[4]. For instance, in the mixture depicted by Figure 7.1, the term $Q_m^{t+1}(z_2^{t+1})$ is equal to 0 for all $\mathring{x}_{i_0}^{(m)} \notin [-0.045, 2.97)$.

Regarding the M-step of the EM algorithm, it yields to the maximization of the log-likelihood function by estimating the optimal parameter values. Note that at each step, $\mathbf{\Theta}^{t+1}$ are found by solving the equations in which the first order partial derivatives of the lower bound function w.r.t. μ_k, σ_k, π_{jk}, d_k, are equal to zero.

In the following, we will provide the details related to the different M-step update equations.

[4]It is not the case for the untruncated model, where $Q_m^{t+1}(z_k^{(m)})$ is positive for all the untruncated normal distributions.

We start by parameter π_{jk}^{t+1}, which corresponds to the weight (or probability) of observations assigned to (or sampled from) the kth component in the mixture model. As discussed earlier, π_{jk}^{t+1} imposes the following constraints: i) all π_{jk}^{t+1} are non-negative, and ii) $\sum_{k=0}^{g} \pi_{jk}^{t+1} = 1$. So:

$$\frac{\partial LL(\mathring{D}|\mathbf{\Theta}^t)}{\partial \pi_{jk}} = \sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \frac{Q_m^{t+1}(z_k^{(m)})}{\pi_{jk}} - \frac{Q_m^{t+1}(z_g^{(m)})}{\left(1-\sum_{k=0}^{g-1}\pi_{jk}\right)} = 0 \tag{7.37}$$

From the previous result, we can immediately conclude that this equation is equal to 0 when π_{jk}^{t+1} verifies:

$$\pi_{jk}^{t+1} = \frac{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} Q_m^{t+1}(z_k^{(m)})}{\sum_{m:\mathbf{MB}(\mathring{x}_{i_0})^{(m)}=j} \sum_{k=0}^{g} Q_m^{t+1}(z_k^{(m)})} = \frac{N_{j[k-1,k)}}{N_j} \tag{7.38}$$

where, $N_{j[k-1,k)}$ is the number of occurrences of $\mathring{X}_{i_0}$ within the interval $[d_{k-1}, d_k)$ when $\mathring{X}_{i_0}$'s Markov blanket's value is j. N_j represents the number of records in $\mathring{D}$ in which $\mathbf{MB}(\mathring{x}_{i_0})^{(m)} = j$. Therefore, for parameters π_{jk}^{t+1}, it is possible to obtain an analytic expression.

Now, let us examine the optimization of parameter μ_k^{t+1}. By taking the derivative of the lower bound function w.r.t. μ_k, we get:

$$\frac{\partial LL(\mathring{\mathcal{D}}|\mathbf{\Theta})}{\partial \mu_k} = \sum_{m=1}^{N} Q_m(z_k^{(m)}) \left(\left(\frac{\mathring{x}_{i_0}^{(m)} - \mu_k}{\sigma_k^2} \right) - \frac{\frac{\partial}{\partial \mu_k} \int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx}{\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx} \right) = 0. \tag{7.39}$$

Let $K = \left(\frac{x-\mu_k}{\sigma_k}\right)$. Then, as $h(x|\boldsymbol{\theta}_k)$ is the probability density function of the normal distribution, the derivative shown in the above equation can be rewritten as follows:

$$\frac{\partial LL(\mathring{\mathcal{D}}|\mathbf{\Theta})}{\partial \mu_k} = \sum_{m=1}^{N} Q_m(z_k^{(m)}) \left(\left(\frac{\mathring{x}_{i_0}^{(m)} - \mu_k}{\sigma_k^2} \right) - \frac{\frac{\partial}{\partial \mu_k} \int_{\frac{d_k-\mu_k}{\sigma_k}}^{\frac{d_{k+1}-\mu_k}{\sigma_k}} \underbrace{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}K^2}}_{F'(K)} dK}{\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx} \right) \tag{7.40}$$

To simplify the previous equation, we replace hereafter the term $\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}K^2}$ by $F'(K)$ to obtain the following expression:

$$\begin{aligned}\frac{\partial}{\partial \mu_k}\int_{\frac{d_k-\mu_k}{\sigma_k}}^{\frac{d_{k+1}-\mu_k}{\sigma_k}} F^{'}(K)dK &= \frac{\partial}{\partial \mu_k}\left(F\left(\frac{d_{k+1}-\mu_k}{\sigma_k}\right) - F\left(\frac{d_k-\mu_k}{\sigma_k}\right)\right)\\ &= -\frac{1}{\sigma_k}F^{'}\left(\frac{d_{k+1}-\mu_k}{\sigma_k}\right) + \frac{1}{\sigma_k}F^{'}\left(\frac{d_k-\mu_k}{\sigma_k}\right)\\ &= -h(d_{k+1}|\boldsymbol{\theta}_k) + h(d_k|\boldsymbol{\theta}_k).\end{aligned} \tag{7.41}$$

Substituting the previous result into Equation (7.39), we get:

$$\frac{\partial LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta})}{\partial \mu_k} = \sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)})\left(\left(\frac{\mathring{x}_{i0}^{(m)}-\mu_k}{\sigma_k^2}\right) + \frac{h(d_{k+1}|\boldsymbol{\theta}_k) - h(d_k|\boldsymbol{\theta}_k)}{\displaystyle\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx}\right) = 0. \tag{7.42}$$

As can be seen, setting the derivative shown in Equation (7.42) to zero cannot be solved analytically due to the presence of the integral. For this reason we resort to a gradient descent algorithm to solve μ_k^{t+1} numerically as follows:

- iteration 0 : initialization of $\mu_k^{t+1} = \mu_k^t$
- repeat until convergence:

$$\mu_k^{t+1} := \mu_k^{t+1} - \alpha\frac{\partial}{\partial \mu_k}LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta}) \tag{7.43}$$

  where α represents the step size, sometimes called the learning rate in the machine learning community. Large values of α tend to make the gradient descent converge faster but it may prevent it from finding the optimal solution. Small values increase the chance to find a good solution, but at the expense of a slow convergence.

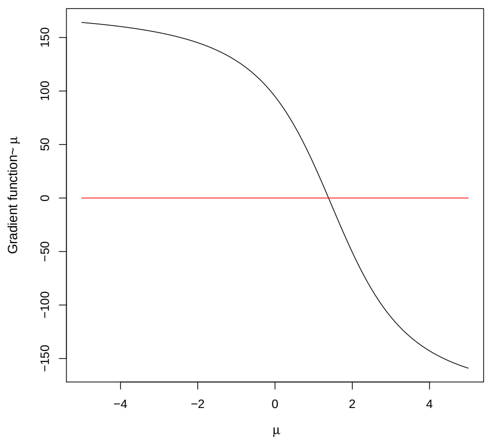


FIGURE 7.12: Gradient function of the parameter μ_2 of the mixture model of Figure 7.1

At each iteration, we choose the best gradient according to our LL function in order to reach a local maximum. As an example, let us compute the optimal value of μ_2 for the truncated mixture model shown in Figure 7.1. Figure 7.12 depicts the curve of the gradient function related to the computation of μ_2 following Equation (7.42). As we know, the optimal value is the one that sets this equation to 0. Therefore, we can clearly deduce that the optimal solution corresponds to the mean-value $\mu_2 \simeq 1.5$, which fits with the original value shown in Figure 7.1.

Now, let us examine the maximization of $LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta})$ w.r.t. σ_k. As discussed previously, the search value will be the one that sets the following first order partial derivative equal to zero:

$$\frac{\partial LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta})}{\partial \sigma_k} = \sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)}) \left(-\frac{1}{\sigma_k} + \frac{1}{\sigma^3}(\mathring{x}_{i0}^{(m)} - \mu_k)^2 - \frac{\frac{\partial}{\partial \sigma_k} \int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx}{\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx} \right) = 0. \tag{7.44}$$

By considering the same change of variables K as that used for μ_k, the derivative of the integral of Equation (7.44) is given as follows:

$$\begin{aligned} \frac{\partial}{\partial \sigma_k} \int_{\frac{d_k-\mu_k}{\sigma_k}}^{\frac{d_{k+1}-\mu_k}{\sigma_k}} F^{'}(K)dK &= -\left(\frac{d_{k+1}-\mu_k}{\sigma_k^2}\right) F^{'}\left(\frac{d_{k+1}-\mu_k}{\sigma_k}\right) + \left(\frac{d_k-\mu_k}{\sigma_k^2}\right) F^{'}\left(\frac{d_k-\mu_k}{\sigma_k}\right) \\ &= -\left(\frac{d_{k+1}-\mu_k}{\sigma_k}\right) h(d_{k+1}|\boldsymbol{\theta}_k) + \left(\frac{d_k-\mu_k}{\sigma_k}\right) h(d_k|\boldsymbol{\theta}_k). \end{aligned} \tag{7.45}$$

Substituting this result into Equation (7.44) and simplifying by σ_k everywhere, we get:

$$\sum_{m=1}^{N} Q_m^{t+1}(z_k^{(m)}) \left(-1 + \frac{1}{\sigma_k^2}(\mathring{x}_{i0}^{(m)} - \mu_k)^2 + \frac{(d_{k+1}-\mu_k)h(d_{k+1}|\boldsymbol{\theta}_k) - (d_k-\mu_k)h(d_k|\boldsymbol{\theta}_k)}{\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx} \right) = 0. \tag{7.46}$$

Notice that Equation (7.46) looks like the derivative equation of the untruncated mixture case (see Equation (7.24)), but it is augmented with an additional term related to the normalization factor. Unfortunately, due to its integral, this term prevents having a close-form formula for the optimal value of σ_k^{t+1}. Therefore, we resort again to a gradient descent to solve σ_k^{t+1} numerically. As for μ_k^{t+1}, we start the descent by setting $\sigma_k^{t+1} \leftarrow \sigma_k^t$.

Figure 7.13 illustrates the computation of σ_2^{t+1} of the truncated mixture model shown in Figure 7.1. From the gradient curve, we can conclude that the optimal value of σ_2^{t+1} is around 0.85, which represents the unique optimal solution (where the gradient is equal to 0).

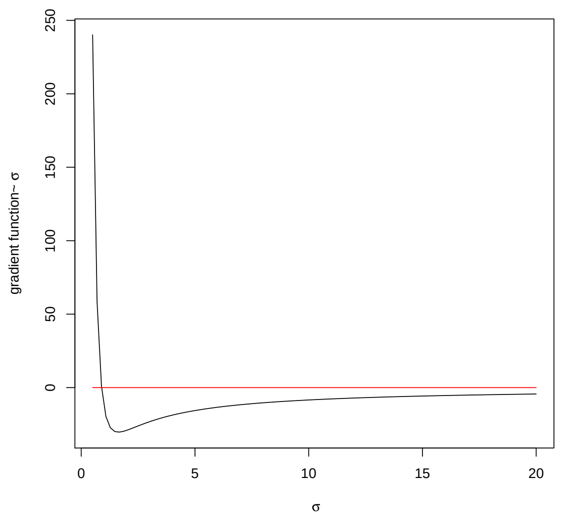


FIGURE 7.13: Gradient function of the parameter σ_2 of the mixture model of Figure 7.1

Finally, there remains to maximize the truncation (or cut point) parameter d_k between the $k-1$th and the kth in the mixture model. Here, it is important to keep in mind that, unlike the other parameters, the optimal computation of d_k involves at the same time two Gaussians. Differentiating the log-likelihood w.r.t. d_k gives:

$$\frac{\partial LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta})}{\partial d_k} = \sum_{m=1}^{N} \quad Q_m^{t+1}(z_{k-1}^{(m)}) \left(- \frac{\dfrac{\partial}{\partial d_k} \displaystyle\int_{d_{k-1}}^{d_k} h(x|\boldsymbol{\theta}_{k-1})dx}{\displaystyle\int_{d_{k-1}}^{d_k} h(x|\boldsymbol{\theta}_{k-1})dx} \right) - Q_m^{t+1}(z_k^{(m)}) \left(\frac{\dfrac{\partial}{\partial d_k} \displaystyle\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx}{\displaystyle\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx} \right) = 0.$$

Then the above equation is equivalent to:

$$\frac{\partial LL(\mathring{\mathcal{D}}|\boldsymbol{\Theta})}{\partial d_k} = \sum_{m=1}^{N} \quad Q_m^{t+1}(z_{k-1}^{(m)}) \left(- \frac{\dfrac{\partial}{\partial d_k} \displaystyle\int_{\frac{d_{k-1}-\mu_{k-1}}{\sigma_{k-1}}}^{\frac{d_k-\mu_{k-1}}{\sigma_{k-1}}} F^{'}(K)dK}{\displaystyle\int_{d_{k-1}}^{d_k} h(x|\boldsymbol{\theta}_{k-1})dx} \right) - Q_m^{t+1}(z_k^{(m)}) \left(\frac{\dfrac{\partial}{\partial d_k} \displaystyle\int_{\frac{d_k-\mu_k}{\sigma_k}}^{\frac{d_{k+1}-\mu_k}{\sigma_k}} F^{'}(K)dK}{\displaystyle\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx} \right) = 0 \tag{7.47}$$

Deriving $F(.)$ w.r.t. d_k in both the $k-1$th and th$k$th truncated Gaussians gives:

$$\sum_{m=1}^{N} -Q_m^{t+1}(z_{k-1}^{(m)}) \frac{h(d_k|\boldsymbol{\theta}_{k-1})}{\displaystyle\int_{d_{k-1}}^{d_k} h(x|\boldsymbol{\theta}_{k-1})dx} + Q_m^{t+1}(z_k^{(m)}) \frac{h(d_k|\boldsymbol{\theta}_k)}{\displaystyle\int_{d_k}^{d_{k+1}} h(x|\boldsymbol{\theta}_k)dx} = 0. \tag{7.48}$$

Like both previous update equations, the current one cannot be solved in an analytic way. That is the reason why a numeric optimization using a gradient descent algorithm is carried out to approximate the optimal d_k value. As in the truncated case, we assume that the estimated d_k value lies inside interval $[\mu_{k-1}, \mu_k]$. Otherwise, if the estimated value of d_k is either higher than μ_k or lower than μ_{k-1}, we force it to be equal to either μ_{k-1} or μ_k.

7.3 Conclusion

In this chapter, we have proposed two versions of a multivariate discretization algorithm designed for the BN structure learning task. It takes into account the dependences among the random variables acquired during learning. As seen, this task is really hard, since it should require the discretization of many variables at the same time, which appears to be untractable, especially for complex network $\mathcal{G}$ (with many nodes and arcs). To alleviate the combinatorics and the computational burden of the multivariate discretization, we have suggested to perform a local search algorithm that iteratively fixes the discretizations of all the continuous variables but one and optimizes the discretization of the latter until some stopping criterion is met. The conditional independences between variables have been taken into account during the discretization process by considering a discretization given the Markov blanket of each continuous variable.

It should be emphasized that our approach assumes that observations of each continuous variable are distributed following a truncated Gaussian mixture model. As such, the optimal discretization is then obtained by computing the best set of truncation intervals (or cut points) of the different components (or Gaussians) in the mixture model. To do so, we have proposed two versions of our approach, both of them relying on the use of the EM algorithm, which is know to be efficient for addressing such a task. Our first approach performs the discretization task by relying on a two-step approach: i) it starts by estimating from dataset $\mathring{\mathcal{D}}$ the set of parameters of an untruncated Gaussian mixture model (instead of a truncated one) using the EM algorithm; and ii) the set of cut point are then identified by solving a quadratic equation aiming to minimize the loss of information shifting from untruncated to truncated Gaussians. Unlike the first version, our second approach relies on the direct estimation of the truncated mixture model parameters using another variant of the EM algorithm and some gradient descent algorithms.

In the next chapter, we perform an experimental study of our proposed approaches on real and synthetic datasets to prove their efficiency.

Chapter 8

Evaluation of the multivariate discretization algorithms

In this chapter, we highlight the effectiveness of our proposed truncated and untruncated multivariate discretization methods by comparing them with the algorithms provided in [RZWL13] and [FG96], hereafter called Ruichu and Friedman respectively. It should be noted from the beginning that Step 4 of Algorithm 9 (see the previous chapter) was performed using a simple *Tabu* search method. For the comparisons, three criteria have been taken into account:

i) the quality of the structure learnt by the algorithm (which strongly depends on the discretization). Arc deletion/orientation/adding errors are considered as a loss of information that results from the discretization process;

ii) the computation time;

iii) the quality of the learnt CPT parameters, which has been evaluated by the BN prediction power on the values taken by some variables given observations.

We designed the datasets as follows: we first selected a (discrete) BN. Then we generated continuous datasets containing from 1000 to 10000 records in the following way: for each random variable X_i, we mapped its finite set of values into a set of consecutive intervals $\{[d_{k-1}, d_k)\}_{k=1}^{|X_i|}$ of arbitrary lengths. Then, we assigned a truncated Gaussian to each interval, the parameters (μ_k, σ_k) of which were randomly chosen. Finally, to generate a continuous record, we first generated a discrete record from the discrete BN using a logic sampling algorithm. Then, this record was mapped into a continuous one by sampling from the truncated Gaussians. Overall, 400 continuous datasets were generated.

To compare the BN structures produced by Ruichu, Friedman and our approach, those were converted into their Markov equivalence class, i.e., into a partially directed DAG

(CPDAG). Such a transformation increases the quality of comparisons since two BNs encode the same distribution if and only if they belong to the same equivalence class. The CPDAGs were then compared w.r.t. their Recall, Precision and F-score (see their respective definitions in Chapter 6). These metrics describe how well the dependences between variables are preserved by learning/discretization. In addition, the quality of the structure learning results has also been tested on real-world BNs which have been taken from the BN repository using the aforementioned criteria.

We compared the discretizations w.r.t. the quality of the produced CPTs. To do so, we used real-world datasets taken from the UCI repository (mentioned in Table 8.4) which contained both continuous and integer random variables. It should be emphasized that the purpose of such a validation method consists in testing the efficiency of the learnt discrete distribution in the prediction task by estimating how accurately a predictive model (learnt from the obtained discrete data) will perform in practice. Remind that in any prediction task, the model is usually obtained from a dataset on which the training task is run, and a dataset of unknown instances against which the learnt model is tested (called also testing dataset)[1]. Given this principle, two subdatasets to "test" and “train” the model were obtained respectively by splitting each UCI database into 2/3 for training and 1/3 for testing.

8.1 Evaluations on synthetic Bayesian networks

For the first set of experiments, on synthetic datasets, we randomly generated discrete BNs following the guidelines given in [ICR04]. Those contained from 10 to 30 nodes and from 17 to 56 arcs. Each node had at most 6 parents and its domain size was randomly chosen between 2 and 5. The CPTs of these BNs represented the π_{jk} of the preceding chapter.

To make the comparison study more efficient, we varied many parameters related to the generated BNs each time we computed the considered criteria as follows:

- the number of nodes: 10, 15, 20, 25 and 30,
- the number of arcs: 17-28, 29-34, 35-44, >44,
- maximum in-degree (or parents) per node: lower or greater than 4 parents.

Note that the last two parameters have been considered together because they entail the same information, which is related to the complexity of the BN. As discussed previously, in this part of our evaluations, we provide a comparison study in terms of structure

[1] In other words, this technique gives an insight on how well the learnt model will generalize to an independent dataset

quality resulting from each discretization algorithm and the runtime taken by the latter. All experiments were performed on a 3 GHz Intel Core2 computer with 3GB of memory running a Debian 4.3 Linux box. The studied discretization algorithms have been implemented using R and C++.

8.1.1 Benchmarks on datasets generated from random BNs

Figure 8.1 shows the average Recall (filled lines), Precision (dashed lines) and F-score over the 400 generated databases. As can be observed, both proposed truncated and untruncated approaches (denoted by TMGCD and MGCD respectively) outperform the others state-of-the-art algorithms for all dataset sizes and for all numbers of nodes. We can clearly remark that both proposed approaches obtain almost the same structure results. This is expected from the beginning since the untruncated approach tries to perform an approximation of the original truncated distribution using the proposed two-step methods (probability estimation and cut points calculation). Yet, there is still a slight improvement shown by the untruncated method. The Recall and Precision obtained by our approaches are actually about 10% higher than Ruichu (with a slight advantage for the MGCD approach for some data sizes) and more than 40% higher than Friedman for all data sizes and node numbers. Regarding the F-score values, it is slightly lower but still significant. The performance of our approaches w.r.t. Ruichu can be explained by that fact that, unlike Ruichu, we fully take into account the conditional dependence/arcs each time we perform the discretization of any node in $\mathcal{G}$. Remember that Ruichu's algorithm discretizes each node independently from the rest, hence leading to an important information loss in terms of conditional dependences between variables. Our performance w.r.t. Friedman can be explained by our choice of exploiting clustering rather than an entropy-based approach which, as shown in the preceding chapter, is not very suitable for the task of BN structure learning.

One important point that can be noted from the experiments' curves is that the comparisons between all the tested discretization approaches seem to be mostly independent from the number of node in $\mathcal{G}$: they give almost the same results for all the values taken by this parameter.

Table 8.1 provides computation time ratios (other method's runtime / TMGCD method's runtime). As can be seen, our truncated method outperforms slightly both the MGCD and Ruichu approaches with an average of 1.63 and 1.83 respectively (but it is significantly better than Ruichu in terms of structure quality). The difference is more significant w.r.t. Friedman since we are around 6.20 times faster than this one while at the same time being more than 40% higher in terms of structure quality (F-score). The time issue related to Friedman's approach can be explained by the fact that it tests each mid-point

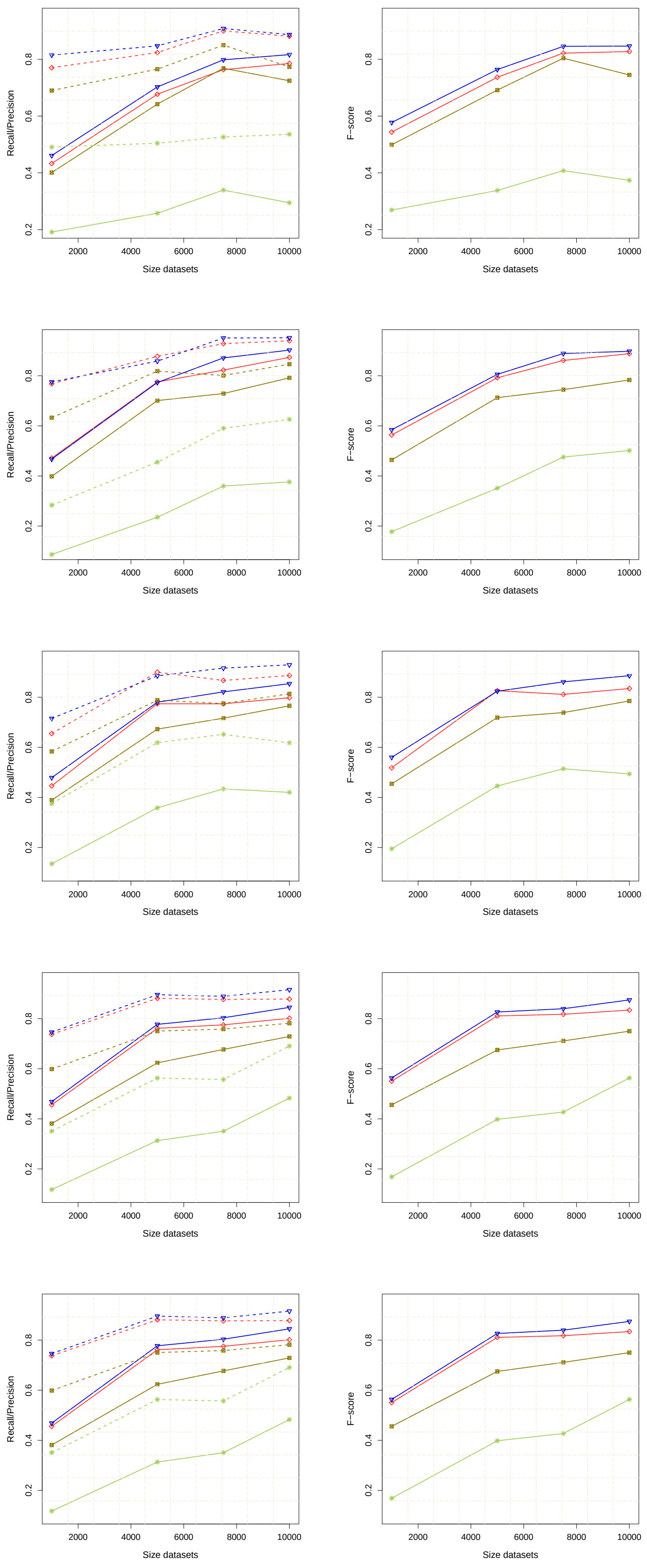


FIGURE 8.1: Recall-Precision and F-score for Benchmarks with 10, 15, 20, 25 and 30 nodes. The results are given for TMGCD (firebrick), MGCD (blue), Ruichu (gold) and Friedman (darkolivegreen)

.

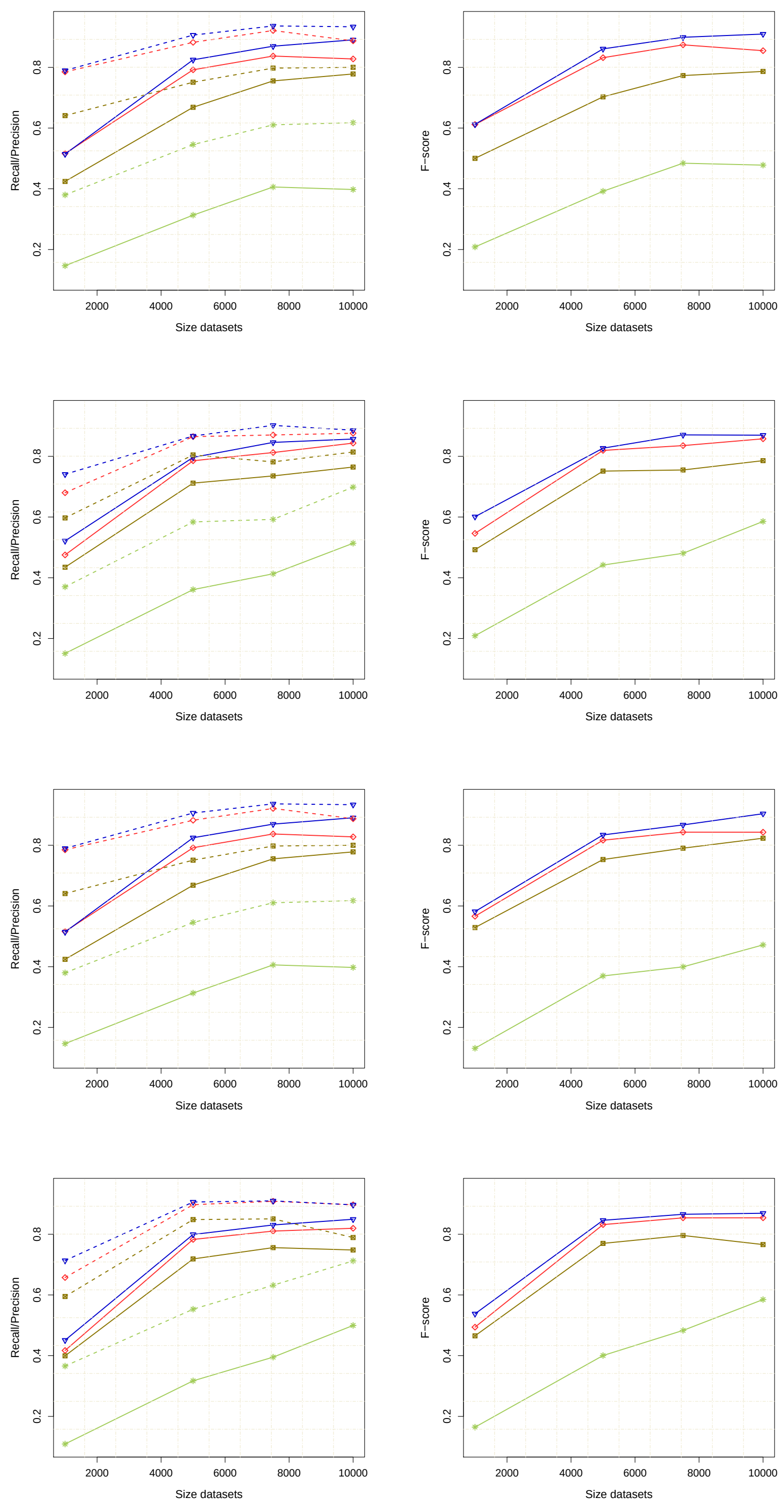


FIGURE 8.2: Recall/Precision and F-score for Benchmarks with a maximum number of parents lower than or equal to 4 and a number of arcs in $\mathcal{G}$ between: 17-28, 29-34, 35-44, and >44. The results are given for TMGCD (firebrick), MGCD (blue), Ruichu (gold) and Friedman (darkolivegreen)

.

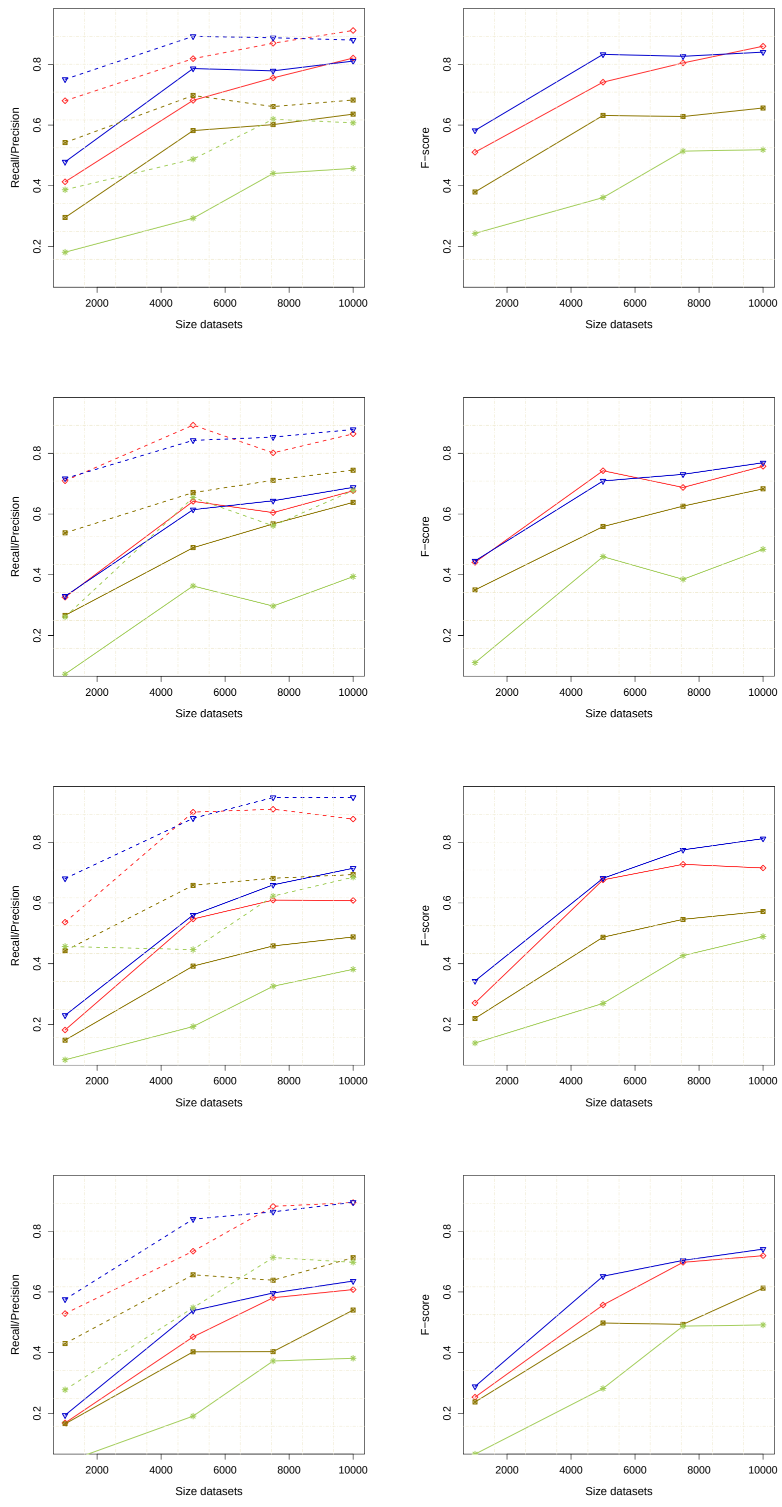


FIGURE 8.3: Recall/Precision and F-score for Benchmarks with a maximum number of parents greater than 4 and a number of arcs in $\mathcal{G}$ between: 17-28, 29-34,35-44,and >44. The results are given for TMGCD (firebrick), MGCD (blue), Ruichu (gold) and Friedman (darkolivegreen)

.

#Nodes	Dataset size	MGCD	Ruichu	Friedman
10	1000	1.21 ± 0.38	1.09 ± 0.49	3.39 ± 3.45
	5000	1.52 ± 0.58	1.94 ± 85	$6.33 \pm .59$
	7500	1.78 ± 0.65	2.18 ± 95	7.38 ± 5.9
	10000	1.78 ± 0.52	2.10 ± 1.11	7.20 ± 4.43
15	1000	1.05 ± 0.31	1.08 ± 0.46	4.21 ± 4.57
	5000	1.6 ± 0.54	1.62 ± 0.72	6.37 ± 5.05
	7500	1.94 ± 1.03	1.94 ± 1.04	7.68 ± 5.65
	10000	2.13 ± 1.01	2.28 ± 1.28	8.18 ± 6.7
20	1000	1.21 ± 0.34	1.24 ± 0.6	4.68 ± 4.39
	5000	1.76 ± 0.56	2.06 ± 0.75	7.07 ± 4.8
	7500	1.95 ± 1.03	2.31 ± 1.57	8.16 ± 6.63
	10000	2.02 ± 0.96	2.3 ± 1.66	8.53 ± 6.16
25	1000	1.21 ± 0.45	1.04 ± 0.5	4.08 ± 3.72
	5000	1.59 ± 0.38	2.02 ± 0.78	6.16 ± 4.13
	7500	1.77 ± 0.61	2 ± 0.81	6.21 ± 4.28
	10000	1.86 ± 0.71	2.17 ± 0.98	6.34 ± 4.1
30	1000	1.22 ± 0.29	1.13 ± 0.43	3.92 ± 3.07
	5000	1.59 ± 0.66	1.78 ± 0.73	5.61 ± 4.64
	7500	1.72 ± 0.65	2.13 ± 1.12	6.17 ± 4.07
	10000	1.87 ± 0.6	2.2 ± 0.96	6.38 ± 3.93

TABLE 8.1: Runtime ratio comparisons between the discretization approaches by varying the number of nodes in $\mathcal{G}$

.

among the continuous values. Therefore, when the data contains a large number of values for a continuous variable, this procedure becomes extremely expensive. This problem can be clearly seen in Table 8.1 where the time ratio between TMGCD and Friedman's approach increases w.r.t. the data sizes.

Regarding the complexity parameters variations (see Figures 8.2 8.3), our proposed approaches usually outperform the rest of the algorithms for all used criteria. When the max-indegree is smaller than or equal to 4 (Figure 8.2), MGCD and TMGCD outperform Ruichu's algorithm with respectively $\sim$ 10% and $\sim$ 5% in terms of Recall and $\sim$15% and $\sim$10% in terms of Precision when the number of arcs is between 17 and 28. In the same context, the differences w.r.t. Friedman (for all criteria) are very important since they are greater than 40% for all dataset sizes. Given these results, it should be noted that the complexity of the BNs, when the max-indegree is smaller than 4 does not exert an important impact on the resulting structure qualities (since they are almost equal whatever the number of arcs). However, by varying max-indegree higher than 4 (see Figure 8.3), this property is ruled out and the F-score values decrease when the complexity increases.

In such a case, except for Ruichu, in which the results do not depend on the BN complexity since the discretization of each variable is performed independently from the rest, the number of arcs has an impact on the structure quality since we observe a decrease from 60%, 54% and 37% (in terms of F-score) for MGCD, TMGCD and Friedman respectively

#Arcs	Dataset size	MGCD	Ruichu	Friedman
17-28	1000	1.19 ± 0.4	1.11 ± 0.46	3.63 ± 3.22
	5000	1.55 ± 0.48	1.81 ± 0.72	6.15 ± 3.93
	7500	1.78 ± 0.84	1.95 ± 0.95	7.3 ± 5.32
	10000	1.88 ± 0.78	2.21 ± 1.05	7.04 ± 4.57
29-34	1000	1.15 ± 0.34	1.07 ± 0.47	5.34 ± 5.23
	5000	1.67 ± 0.58	1.94 ± 0.65	7.45 ± 5.92
	7500	1.77 ± 0.87	1.93 ± 0.7	7.55 ± 6.57
	10000	1.98 ± 0.83	2.19 ± 1.39	8.91 ± 7.08
35-44	1000	1.2 ± 0.35	1.23 ± 0.64	4.00 ± 3.93
	5000	1.73 ± 0.64	2 ± 0.72	5.99 ± 4.8
	7500	1.95 ± 0.84	2.46 ± 1.55	6.84 ± 5.16
	10000	2.03 ± 0.94	2.37 ± 1.51	6.72 ± 4.85
>44	1000	1.18 ± 0.3	1.03 ± 0.32	4.23 ± 3.12
	5000	1.71 ± 0.48	1.74 ± 0.95	6.67 ± 4.45
	7500	2 ± 0.82	2.15 ± 1.38	7.62 ± 4.72
	10000	2.09 ± 0.71	2.12 ± 1	7.85 ± 4.35

TABLE 8.2: Runtime ratio comparisons between then discretization approaches by varying the number of edges in $\mathcal{G}$.

when the number of arcs is between 17 and 28 to 30%, 26%, 5% when the complexity becomes greater than 45 arcs in $\mathcal{G}$ (this variation slightly varies for the rest of data sizes).

These results highlight one important problem of the multivariate discretization approach, which is related to the BN complexity. In fact, when the complexity of the BN increases (in terms of arcs and indegrees), this incurs a strong overhead on the computation and propagation process of each new discretization in $\mathcal{G}$, leading therefore to a fast convergence to a local optimum solution. This impact can be clearly seen on the left side curves of Figure 8.3, where the Recall rates (or True Positive Rate) decrease when the complexity increases (because we loose a lot of information).

8.1.2 Benchmarks on continuous datasets from real BNs

In this section we consider three benchmark networks (presented in Table 8.3) taken from the BN repository [Scu09]. Samples of sizes ranging from 1000 to 10000 have been generated from the benchmark networks by varying the percentage of continuous variables α as follows: $\alpha = 0.1$, $\alpha = 0.3$ and $\alpha = 0.6$. This variation is performed in order to study in details the behaviors of each discretization algorithm when the number of continuous variables changes in the same network. Note that all the variables in the benchmark BNs are discrete at the beginning. This is the reason why, to be able to apply the studied learning algorithms, for each experiment, we select randomly a set of variables (given the α value) in each network and we generate N continuous observations from their domains using a logic sampling algorithm. It should be noted that in these experimentations we have defined a timeout of 30 minutes. This means

Details BN	ASIA	SACHS	CHILD
Number of nodes	8	11	20
Number of arcs	8	17	25
Number of parameters	18	178	230
Average Markov blanket size	2.5	3.09	3
Average degree	2	3.09	1.25
Maximum in-degree	2	3	2

TABLE 8.3: Characteristics of the real-world BNs used.

that when the algorithm exceeds the defined runtime duration, we stop the discretization process and we denote its result by "-". Figure 8.4 depicts the obtained results for the ASIA benchmark. As can be seen, when $\alpha = 0.1$ our approaches and Ruichu obtain almost the same results in terms of structure quality. Regarding Friedman's approach, it always obtains the lowest results. By increasing the number of continuous variables, the difference in terms of F-score between both MGCD and TMGCD and the state-of-the-art algorithms becomes significant, i.e., they reach 13% and 10% w.r.t. to Ruichu respectively when the dataset size is equal to 10000 and 38% and 42% comparing to Friedman. Given these results, we should mention that Ruichu and Friedman prove to be sensitive to the variations in the number of continuous variables because their reliabilities decrease (in terms of F-score) from 82% and 81% (when $\alpha = 0.1$) to 78% and 50% (when $\alpha = 0.6$) respectively, while our approaches remain stable.

Regarding the results of the SACHS network (see Figure 8.6), which is represented with a more complex graph than ASIA[2], our approaches always outperform the other methods (see the F-score curves). The result differences between the different approaches as well as the reliability of each one change according to the percentage of continuous variables. For instance, the F-score values decrease from 81%, 81%, 78% and 77% when $\alpha = 0.1$ for MGCD, TMGCD, Ruichu and Friedman respectively to 70%, 66%, 43% and 45% when $\alpha = 0.6$. By taking into account the results obtained for the ASIA and SACHS benchmarks, we can clearly remark the impact of the BN complexity (see Table 8.3) on the reliability of the discretization, since the results obtained for the SACHS network are considerably lower than those related to ASIA. This relationship between the discretization reliability and the BN complexity is reinforced by the results characterizing the CHILD network (which is more complex than ASIA and SACHS). Observing the curves shown in Figure 8.5, we can see that our approaches are better than the other algorithms in the majority of cases. The difference between MGCD and both Ruichu and Friedman reach 33% and 48% in terms of F-score respectively when the dataset size is equal to 1000 and $\alpha = 0.6$. This difference is slightly lower for the rest of the parameters values. From these results, a considerable deterioration with some instability of the structure qualities can be clearly seen, especially for the state-of-the-art approaches, in which the F-score values reach the low rates of 40% and 29% for Ruichu and Friedman

[2]We talk about complexity in terms of arcs and parameters numbers needed to represent the network.

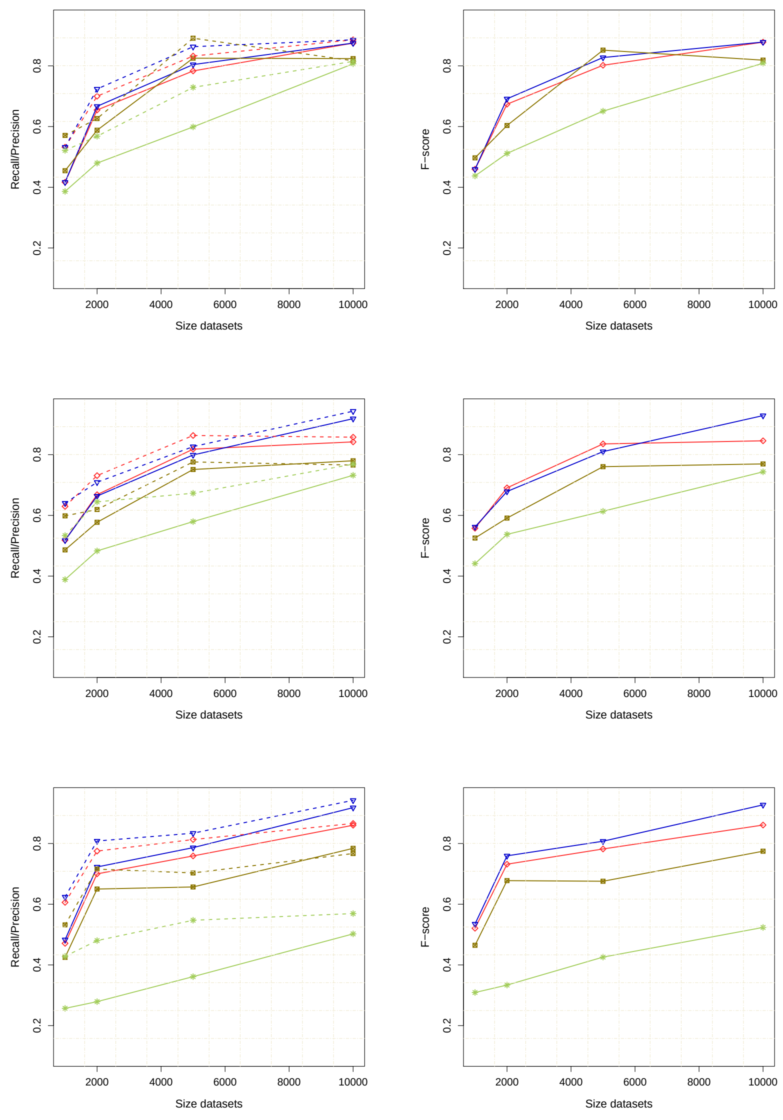


FIGURE 8.4: Recall/Precision and F-score for the ASIA datasets with α=10%, α=30% and α=60% of continuous variables. The results are given for TMGCD (firebrick), MGCD (blue), Ruichu (gold) and Friedman (darkolivegreen)

respectively. These deterioration results characterizing the classical methods are mainly due to the difficulty shown by Friedman in terms of discretization computation (using the entropy metric) and propagation when the network contains a lot of arcs. Regarding Ruichu's method, the univariate character of this approach ignores one of the most valuable information that can be exploited in the discretization process: that which is represented by the set of dependence/arcs learnt in the BN. These problems can be clearly seen in the set of Recall/Precision curves of Figure 8.5, where the Recall rates obtained by both Friedman and Ruichu do not exceed in the best cases 58% for all α and dataset sizes.

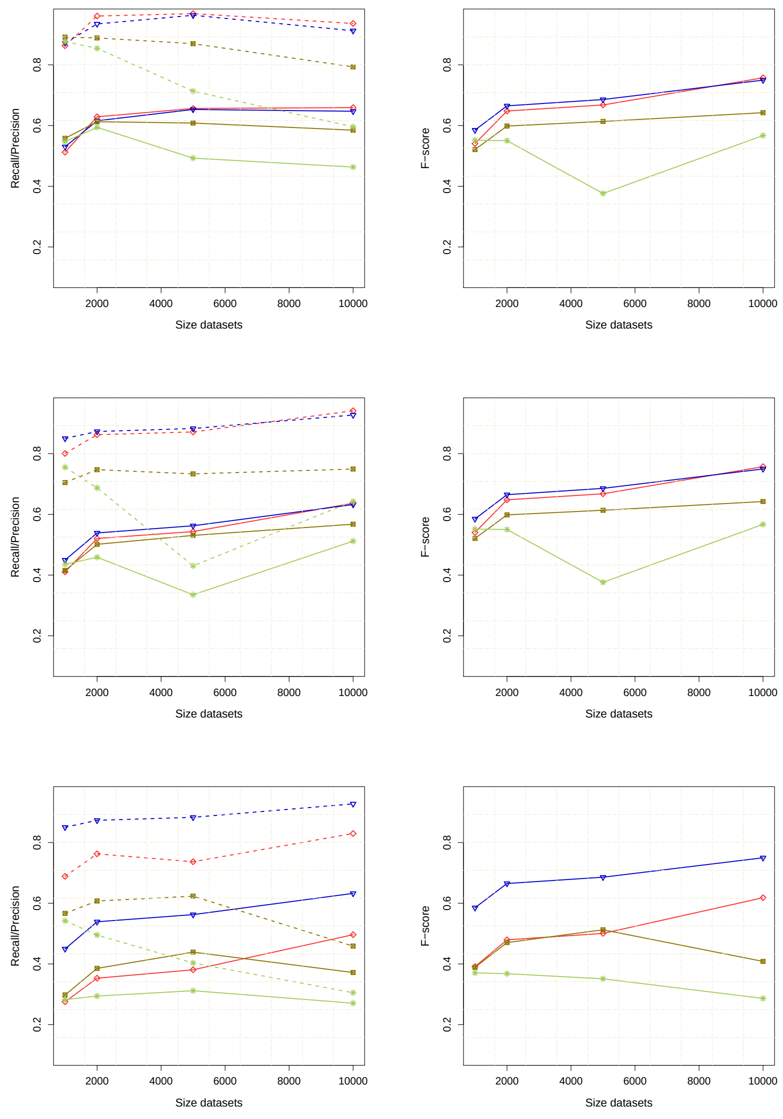


FIGURE 8.5: Recall/Precision band F-score for the CHILD datasets with α=10%, α=30% and α=60% of continuous variables. The results are given for TMGCD (firebrick), MGCD (blue), Ruichu (gold) and Friedman (darkolivegreen)

8.2 Evaluation of the learnt BN CPTs quality

In this section, we describe the experimentations designed to test the soundness of the CPTs returned by each discretization algorithm using a set of real-world databases that have been taken from the UCI repository. To do so, many evaluation criteria can be used in this context. Among them we can mention the Kullblack-Leibler distance which allows to measure the divergence between the learnt and the original distributions (by means of equation (8.1)[3]) and the prediction power resulting from the learnt CPTs (in a supervised classification task). Unfortunately, the first option (the Kullback-Leibler divergence computation) is doomed to be unfeasible in our case since the number of

[3]As the DL_{KL} decreases, the probability distributions are more and more similar.

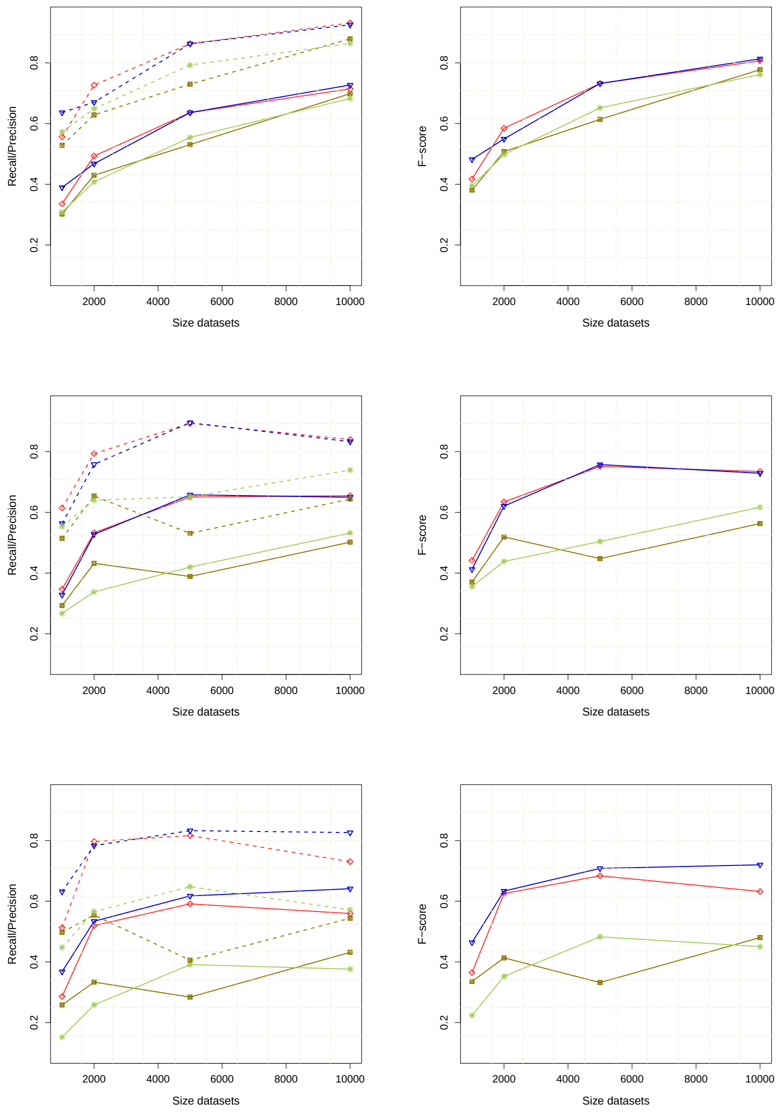


FIGURE 8.6: Recall/Precision and F-score for the SACHS datasets with 10%, 30% and 60% of continuous variables. The results are given for TMGCD (firebrick), MGCD (blue), Ruichu (gold) and Friedman (darkolivegreen)

modalities (or intervals) learnt by each discretization method are not necessarily equal to those entailed by the original probability distribution.

$$DL_{KL}(P||Q) = \sum_i P(\mathbf{x}_i) \log \frac{P(\mathbf{x}_i)}{Q(\mathbf{x}_i)} \tag{8.1}$$

8.2.1 Prediction accuracy results

In this section, we compare the discretizations w.r.t. the quality of the produced CPTs. Our experimentations are run on the 8 databases given in Table 8.4 from website: `https://archive.ics.uci.edu/ml/datasets.html`. Remind that each database is split into a learning (2/3) and a test (1/3) databases. For each record in the latter, we compute

Name	Default Task
Robot Navigation	Classification
Image Segmentation	Classification
EEG Eye State	Classification
TelescopeMAGIC Gamma Telescope	Classification
Letters recognition	Classification
Banknote authentication	Classification
TEBHRTamilnadu Electricity Board Hourly Readings	Class/Regress/ClusClassification, Regression

TABLE 8.4: Used UCI datasets

the value (learnt by each of the 4 algorithms on the learning database) of the class target variables given the observations about the explanatory variables. The evaluation of the supervised classification task (using the BN model) is been carried out by computing the following three criteria:

- the classification rate (CR): in a supervised classification problem, when we build a classifier model, we want to look at the accuracy of that latter by computing the number of correct predictions from all prediction made (as the greater the classification rate, the more reliable classifier). The classification rate is given by the following equation:

$$CR = \frac{\text{Instances classified correctly}}{\text{All classified instances}} \tag{8.2}$$

- the area under ROC curves: it is represented by a graphical plot that illustrates the accuracy of a binary classifier system. As shown in Figure 8.7, the ROC curve is obtained by plotting the true positive rate (TPR) against the false positive rate (FPR) for all classified instances. For instance, let us consider a two-class classification problem (or binary classification), in which the outcome of each considered instance is either positive (+) or negative (-). In such a case, if the result from the prediction is (+) and the original value is also (+), then it is said to be a true positive (TP); however if the actual value is (-) then it is considered as a false positive (FP). The coordinates (FPR_i and TPR_i) of each point in the curve of Figure 8.7 are calculated as follows:

$$FPR_i = \frac{\text{Number of negative among the i first predictions}}{\text{Total number of negative}} \tag{8.3}$$

$$TPR_i = \frac{\text{Number of positive among the i first predictions}}{\text{Total number of positive}} \tag{8.4}$$

- the F-score metric: it is calculated in the same manner as the one considered in the structure case.

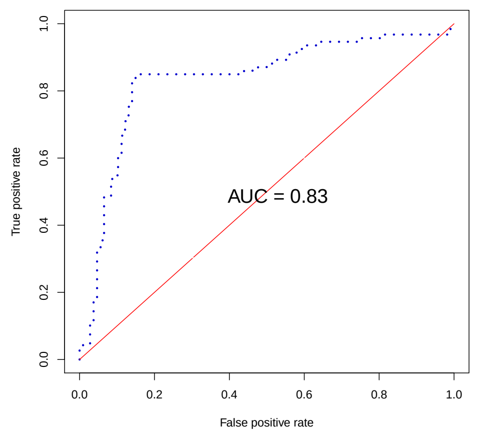


FIGURE 8.7: ROC curve with an area=0.83

As we can see in Tables 8.5, 8.6 and 8.7, except for the datasets Banknote and Image segmentation, our algorithms outperforms the other methods, especially Ruichu, which fails to have correct predictions due to its univariate discretization not taking into account the conditional dependences among the random variables and especially those between the explanatory variables and the class one. For instance, the classification rate difference between MGCD and Ruichu is significantly higher in both small and high dimensional datasets, since they reach 9.78% and 8.16% for respectively Robot Navigation (3 variables) and Image Segmentation (19 variables). These differences are also significant in terms of ROC curve and F-score metrics (see Tables 8.7 and 8.7). Regarding the Friedman's results, they are closer to ours given Banknote, Image Segmentation and TEBHR but they are significantly lower than ours for the rest of databases: MGCD is higher with 9.9%, 9% and 8% for respectively Robot Navigation, EEG Eye State and Telescope. It should be noted that, unlike its results in terms of structure, the prediction power shown by Friedman's approach can be explained by the fact that the entropy measure can be considered as a very well suited metric for the classification task (which is also used by many supervised classification algorithms such as the decision tree) because it tries to find the set of cut points that enhance the entropy measure of the class given the rest of variables, increasing therefore the efficiency of the classifier. But we should remind that we are significantly better than Friedman in terms of structure quality and run times computation. Our cluster-based approaches perform efficiently both structure learning and class classification problems.

8.3 Conclusion

In this chapter, we have performed an experimental study on the multivariate discretization approaches developed in the preceding chapter. We compared them with two

Databases	MGCD	TMGCD	Ruichu	Friedman
Robot Navigation	**88.03**	85.76	78.32	76.65
Image Segmentation	90.19	85.47	82.03	**90.82**
Eye	64.22	**65.34**	59.49	61.24
Telescope	76.00	**76.32**	75.21	72.33
Letters recognition	60.14	46.42	45.85	-
Banknote	89.27	88.84	80.04	**89.48**
TEBHR	100	100	100	100

Table 8.5: Resulting classification rates using the UCI datasets

Databases	MGCD	TMGCD	Ruichu	Friedman
Robot Navigation	**0.979**	0.969	0.858	0.88
Image Segmentation	**0.98**	**0.98**	0.96	**0.98**
Eye	**0.70**	**0.70**	0.6	0.61
Telescope	**0.83**	**0.83**	0.79	0.75
Letters recognition	**0.95**	0.9	0.9	-
Banknote	0.92	**0.94**	0.79	0.89
TEBHR	1	1	1	1

Table 8.6: Resulting ROC are using the UCI datasets

Databases	MGCD	TMGCD	Ruichu	Friedman
Robot Navigation	**0.87**	0.84	0.72	0.68
Image Segmentation	0.89	0.84	0.81	**0.9**
EEG Eye State	0.63	**0.64**	0.35	0.6
Telescope	**0.74**	**0.74**	0.72	0.69
Letters recognition	0.6	0.47	0.46	-
Banknote	**0.89**	0.88	0.8	**0.89**
TEBHR[4]	1	1	1	1

Table 8.7: Resulting F-score rates using the UCI datasets

state-of-the-art algorithms: Ruichu (univariate approach) and Friedman (multivariate approach). The experimentations were carried out using synthetic datasets generated from both random and real world BNs. In addition, real-world datasets (taken from the UCI repository), have been used to test the quality of the CPTs resulting from each discretization.

As shown in the previous sections, the original way by which the conditional independences between variables have been taken into consideration during the discretization process has enabled our methods to significantly outperform the state-of-the-art algorithms. This improvement has been shown in terms of structure quality: unlike the other methods, our approaches have allowed to efficiently learn the set of conditional dependences/arcs between random variables, minimizing therefore the information loss (which has been measured in terms of Precision, Recall and F-sore).

The reliability of our methods in terms of CPTs parameters quality has been demonstrated throughout the computation of the prediction power related to each discretization by using a set of real world datasets that do no made any assumption about the variables distribution (which make the evaluation more reliable). As shown, our approaches allow to perform efficiently the supervised classification task since they outperform in the majority of the datasets the other methods in terms of classification rate, area under ROC curve and F-score measures. In all cases (structure and prediction), the reliability of our approaches compared to the other methods comes precisely from the way by which the conditional independences are taken into consideration, but also from the proposed methods to compute the set of cut points of each interval. In this context, the significant difference w.r.t. Ruichu has been explained by the fact that the latter does not take into consideration the set of conditional dependences/arcs learnt during the BN learning phase, since it performs a simple univariate discretization. Regarding Friedman, despite the multivariate feature of this method, it fails to obtain good results (especially in terms of structure quality) because of the bad choice of the entropy criterion to perform its multivariate discretization in the BN learning context.

Chapter 9

Conclusion and Future Works

9.1 Conclusion

Performing the diagnosis of severe nuclear accident scenarios from sensors' partial observations is a challenging task. In that respect, the Fukushima nuclear accident highlighted several issues. Notably, the experts of the domain were confronted with the difficulty of giving a detailed explanation of the appropriate severe accident scenarios that gave rise to the observed phenomena (following the accident). To cope with this problem, several simulations of possible accidental scenarios have been conducted using the ASTEC software packages in order to obtain a good approximation (or reconstruction) of the most probable scenario that induced the available observations (vessel level, pressure in the containment, rejected iodine amount, containment break, etc.). In this context, ASTEC simulations have focused mainly on the best known accident scenarios that could occur in any pressurized water reactor (PWR) that may lead to the core melting. However, this approach was inefficient because, as is well known, the diagnosis task needs a backward induction on the time scale of the accident to determine exactly what happened in the PWR rather than performing a prognosis analysis (as the ASTEC simulation does). More precisely, the difficulty of applying such an approach in this diagnosis application results from at least three reasons:

i) the existence of an infinity of accident scenarios that could be imagined in a given PWR;

ii) the uncertainty related to many phenomena that may occur during the accident (chemical reactions, containment failure, etc.), which are not easily represented with the ASTEC code;

iii) the lack of assignment of a credibility degree (or a probability) to the different possible scenarios at the end of the diagnosis process.

To cope with these issues, BNs and their inference engines seem better suited for helping Decision Makers make the best diagnosis of the accident scenario given the partial available observations. We believe that they could have helped limit significantly the severe consequences of the accident.

Bayesian networks (BN) allow to represent the different accident scenarios with a graphical structure whose nodes represent the random variables involved in the different phenomena inside the reactor. They enable the compact representation of the system under study by means of a graphical and probabilistic formalism, easily understandable by any user. In addition, they are supplied with fast inference engines that enable to answer efficiently various types of probabilistic queries (computation of marginal, a priori, a posteriori probabilities, of most probable explanations, of maximum a posteriori, etc.), which perfectly suits our diagnosis purpose. This is the reason why we have decided to work with this predictive model.

As discussed in Chapter 3, in the past twenty years, numerous efforts have been devoted to learn both BN's graphical structure and the parameters of their conditional probability tables from datasets. As a consequence, these algorithms can be considered as a good support helping to learn automatically the BN from the simulated dataset without necessarily needing the domain expert knowledge. However, there exist important critical applications for which no current BN learning algorithm proves to be satisfactory. Such a situation occurs in our diagnosis case because some deterministic relations between some variables exist in the data. In this context, deterministic relations essentially represent equations modeling the various phenomena occurring in a damaged nuclear power plant during a severe accident (e.g., ideal law gas). The presence of such relations in the dataset makes BNs learning a complex task since the faithfulness property, on which rely the majority of the learning algorithms, is ruled out, i.e., the equivalence between the conditional independences represented by graph $\mathcal{G}$ and those of the true distribution P fails to hold. In this case, learning algorithms can fail to recover a correct BN structure.

The problem of BN structure learning from data containing deterministic relations has been studied in details in Chapters 4 and 5. For solving this issue, we have provided an algorithm, which relies on very effective dedicated rules whose mathematical correctness has been proven. The general architecture of the approach we proposed is as follows: it first constructs the skeleton of the BN, i.e., its graphical structure in which arcs are substituted by edges; then our algorithm converts this structure into a directed acyclic graph which is subsequently refined by a greedy search. At each step of the algorithm, we have introduced our own rules to cope with deterministic relations. Those exploit both valuable information coming from deterministic nodes and other useful properties entailed by BNs. Exploiting these rules, we have been able to improve significantly the structure learning quality compared to the state-of-the-art methods. This has been highlighted in the experimental study of Chapter 6.

Unfortunately, BN structure learning in the presence of deterministic relations is only one of the problems raised by severe accident diagnosis. Another one results from the existence of a mixture of continuous and discrete variables in the dataset, which forms an obstacle on the effectiveness of BNs learning algorithms. A a matter of fact, BN learning and inference involve only discrete variables. One common way to address this issue consists in preprocessing the dataset by first discretizing continuous variables and, then, in resorting to classical BN learning algorithms. However, as discussed in Chapter 5, such a method is inefficient since the conditional dependences/arcs learnt during the structure learning phase are not taken into consideration, which prevents the BN learning algorithm to be fully effective.

In Chapter 7, we therefore advocated to discretize while learning the BN structure and we proposed two new algorithms that take into consideration all the conditional dependences/arc learnt so far, thereby minimizing the information loss. Unfortunately, multivariate discretization is very hard to perform when many variables are involved. So, in order to be scalable and to reduce as much as possible the computational overhead incurred by multivariate discretization on the learning process, we have proposed to perform a local search algorithm rather than a global one, discretizing only one variable at a time while maintaining the discretization of the rest fixed. Iterating this process, we reach “local” optima. Regarding the conditional independences between variables, they have been taken into account by considering the discretization modalities of the Markov blanket of each continuous variable. As proven in Chapter 7, this makes the considered continuous variables independent from the discretization of the rest of the model.

To compute the number of intervals in the discretization of the continuous variables as well as their boundaries (or cut points), our algorithms assume that the continuous variables are distributed w.r.t. a truncated Gaussian mixture model, i.e., each interval is represented by its own truncated normal distribution. As such, in our case, the discretization process consists in finding the optimal parameters of the considered discretization. To do so, we have proposed two different algorithms, both of them relying on the use of the EM algorithm to perform the discretization task:

- the first version consists to first estimate the set of parameters of an untruncated (instead of a truncated) mixture model using the EM algorithm. In a second step, it computes the best corresponding set of cut points by solving a quadratic equation.
- the second version directly estimates the truncated mixture model parameters using another variant of the EM algorithm with a gradient descent algorithm.

The efficiency of both proposed approaches has been shown in the experimental study performed in Chapter 8. The improvement of our algorithms w.r.t. the state-of-the-art methods has been shown both in terms of the quality of the learnt structure and that of the learnt probability distribution (better predictive model).

9.2 Future works

Since our problem comes from a real-world application, several extensions of our proposed algorithms can be considered, which aim to enhance our approaches by taking into consideration others constraints related to the diagnosis process that were not considered in our scope. In the following, we describe some of the possible extensions, and we begin by the structure learning algorithm.

First, the proposed BN structure learning algorithms were essentially tested on synthetic datasets which have been sampled from random BNs. The main reasons for using such datasets were that it allowed us:

i) to generate CPTs representing deterministic relations between variables;

ii) to vary many important parameters related to the BN (the number of deterministic nodes, the complexity of the BN, etc.);

iii) to have a ground truth that could be exploited to assess the quality of the BNs produced by our algorithms.

However, it would of course be important to investigate in more details the behavior of our structure learning algorithms on real-world datasets. In this context, the datasets generated with ASTEC would be a very good choice to strongly validate the performance of our algorithm. We should emphasize here that we could not perform such experiments due to the absence of complete databases containing the possible accident scenarios that may occur in a PWR. Note that the datasets used during our work and from which the previous problems have been revealed (related to the structure learning and to the discretization process), represent a very simple case of accidents that cannot be exploited to perform a complete experimental study. Second, the only source of unfaithfulness that has been considered in our scope came from the presence of deterministic nodes in $\mathcal{G}$. However in many real-world application domains (and very likely in ours), others causes of unfaithfulness of P may occur, notably the equivalence partitions and the pseudo-independent relations. The latter should be taken into account because they may induce many errors during the BN structure learning (if they exist in $\mathcal{D}$). Finally, we believe that the representation of the temporal evolution of the severe nuclear accident should also be taken into account by using dynamic BNs rather than a static one. Besides the dynamic aspect of the severe accidents, the latter are characterized by a non-stationary process over time. For instance, the Hydrogen (H_2) amount released at time t is less important than the one induced by the molten core-concrete interactions at time $t + n$. Hence, we believe that an extended version of the BN structure learning algorithm taking into consideration the non-stationarity of the process may significantly improve the prognosis/diagnosis analysis.

Regarding the discretization problem, other possible extensions of our algorithms can be incorporated. First, it is important to make our discretization approach more generic through, for instance, the use of non-parametric probability estimation methods rather than a truncated mixture model. Another possibility to make our discretization approaches more generic consists in replacing our assumption about the distribution of the observations within each interval by a mixture of truncated (or untruncated) Gaussian models rather than only one Gaussian. Second, the way by which conditional independences between variables are taken into account using the Markov blanket reaches its limit when the number of variables composing the blanket becomes large. In such a case, memory consumption becomes too large and the algorithms fails for that reason. It would therefore be of interest to produce new algorithms that do not need to store in memory as many parameters as those induced by the Markov blanket. Finally, merging our BN structure learning algorithm with deterministic relations and those with the discretization approaches is a necessity in our diagnosis case, since the *Tabu* search used in our discretization algorithm does not suit the cases where the faithfulness property is lost. Recall that the main issue in such a case is that our approach starts by learning the skeleton (undirected graph) of the BN while the discretization approach needs a completely oriented graph to perform the parameters estimation. So, new algorithms should be developed in this direction.

Bibliography

[AC96] Silvia Acid and Luis M. De Campos. An algorithm for finding minimum d-separating sets in belief networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 3–10, 1996.

[AD03] D. Allen and A. Darwiche. New advances in inference by recursive conditioning. In *Uncertainty in Artificial Intelligence (UAI)*, pages 2–10, 2003.

[AGM06] Joaquín Abellán, Manuel Gómez-Olmedo, and Serafín Moral. Some variations on the PC algorithm. In *Workshop on Probabilistic Graphical Models (PGM)*, pages 1–8, 2006.

[Aka70] Hirotugu Akaike. Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, 22(1):203–217, 1970.

[BDTP03] F. Bacchus, S. Dalmao, and T. Toniann Pitassi. DPLL with caching: A new algorithm for #sat and Bayesian inference. In *Electronic Colloquium on Computational Complexity (ECC)*, 2003.

[BHKL91] PP Bonissone, M Henrion, LJ Kanal, and JF Lemmer. Equivalence and synthesis of causal models. In *Uncertainty in Artificial Intelligence (UAI)*, volume 6, page 255, 1991.

[Bou04] Marc Boullé. Khiops: A statistical discretization method of continuous attributes. *The Journal of Machine Learning Research*, 55(1):53–69, 2004.

[Bou06] M. Boullé. MODL: a Bayes optimal discretization method for continuous attributes. *The Journal of Machine Learning Research*, 65(1):131–165, 2006.

[Bun91] Wray Buntine. Theory refinement on bayesian networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 52–60. Morgan Kaufmann Publishers Inc., 1991.

[C$^+$75] United States Nuclear Regulatory Commission et al. Reactor safety study. an assessment of accident risks in us commercial nuclear power plants. executive summary. Technical report, United States Nuclear Regulatory Commission, 1975.

[CC90] R Martin Chavez and Gregory F Cooper. A randomized approximation algorithm for probabilistic inference on bayesian belief networks. *Networks*, 20(5):661–685, 1990.

[CD08] M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence Journal*, 172(6-7):772–799, 2008.

[CGK$^+$02] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence Journal*, 137(1-2):43–90, 2002.

[CGOM08] A Cano, M Gómez-Olmedo, and S Moral. A score based ranking of the edges for the pc algorithm. In *Workshop on Probabilistic Graphical Models (PGM)*, pages 41–48. Citeseer, 2008.

[CH92a] G. F. Cooper and E. Herskovits. Bayesian method for the induction of probabilistic networks from data. *The Journal of Machine Learning Research*, 9:309–347, 1992.

[CH92b] Gregory F Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *The Journal of Machine Learning Research*, 9(4):309–347, 1992.

[CH96] David Maxwell Chickering and David Heckerman. Efficient approximations for the marginal likelihood of incomplete data given a bayesian network. In *Uncertainty in Artificial Intelligence (UAI)*, pages 158–168. Morgan Kaufmann Publishers Inc., 1996.

[Chi95a] David Maxwell Chickering. A transformational characterization of equivalent bayesian network structures. In *Uncertainty in Artificial Intelligence (UAI)*, pages 87–98. Morgan Kaufmann Publishers Inc., 1995.

[Chi95b] David Maxwell Chickering. A transformational characterization of equivalent bayesian network structures. In *Uncertainty in Artificial Intelligence (UAI)*, pages 87–98. Morgan Kaufmann Publishers Inc., 1995.

[Chi02a] David Maxwell Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, February 2002.

[Chi02b] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, November 2002.

[CM14] Diego Colombo and Marloes H Maathuis. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

[Dar01] A. Darwiche. Recursive conditioning. *Artificial Intelligence Journal*, 125(1-2):5–41, 2001.

[Daw79] A Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–31, 1979.

[DD02] Denver Dash and Marek J Druzdzel. Robust independence testing for constraint-based learning of causal structure. In *Uncertainty in Artificial Intelligence (UAI)*, pages 167–174. Morgan Kaufmann Publishers Inc., 2002.

[Dec99] Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence Journal*, 113(1):41–85, 1999.

[Die96] F.J. Diez. Local conditioning in Bayesian networks. *Artificial Intelligence Journal*, 87:1–20, 1996.

[DJM$^+$10] P. Daniusis, D. Janzing, J. Mooij, J. Zscheischler, B. Steudel, K. Zhang, and B. Schölkopf. Inferring deterministic causal relations. In *Uncertainty in Artificial Intelligence (UAI)*, pages 143–150, 2010.

[DKS95] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *The International Conference on Machine Learning (ICML)*, pages 194–202, 1995.

[DLR77] A. P Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society,*, pages 1–38, 1977.

[ENFS02] Gal Elidan, Matan Ninio, Nir Friedman, and Dale Shuurmans. Data perturbation for escaping local maxima in learning. In *AAAI Conference on Artificial Intelligence*, pages 132–139, 2002.

[FG96] N. Friedman and M. Goldszmidt. Discretizing continuous attributes while learning Bayesian networks. In *The International Conference on Machine Learning (ICML)*, pages 157–165, 1996.

[FHJ08] Andrew Fast, Michael Hay, and David Jensen. Improving accuracy of constraint-based structure learning. Technical report, Technical report 08-48, University of Massachusetts Amherst, Computer Science Department, 2008.

[FI93] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1022–1029, 1993.

[FJ00] A. Faÿ and J.-Y. Jaffray. A justification of local conditioning in Bayesian networks. *International Journal of Approximate Reasoning (IJAR)*, 24(1):59–81, 2000.

[FJ09] Andrew Fast and David Jensen. Constraint relaxation for learning the structure of bayesian networks. Technical report, Technical Report 09-18, Computer Science Department, University of Massachusetts, Amherst, 2009.

[GJ06] Christophe Gonzales and N. Jouve. Learning bayesian networks structure using markov networks. In *Workshop on Probabilistic Graphical Models (PGM)*, pages 147–154, 2006.

[Glo90] Fred Glover. Tabu search part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[GP01] Steven B Gillispie and Michael D Perlman. Enumerating markov equivalence classes of acyclic digraph models. In *Uncertainty in Artificial Intelligence (UAI)*, pages 171–177. Morgan Kaufmann Publishers Inc., 2001.

[GVP90] D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks*, 20:507–534, 1990.

[Hec95] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report TR-95-06, Microsoft Research, 1995.

[HGC95] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *The Journal of Machine Learning Research*, 20:197–243, 1995.

[ICR04] J. S. Ide, F. G. Cozman, and F. T. Ramos. Generating random Bayesian networks with constraints on induced width. In *European Conference on Artificial Intelligence (ECAI)*, pages 323–327, 2004.

[JHS10] D. Janzing, P. Hoyer, and B. Schölkopf. Telling cause from effect based on high-dimensional observations. In *The International Conference on Machine Learning (ICML)*, pages 479–486, 2010.

[J.L07] J.Lemeire. *Learning Causal Models of Multivariate Systems and the Value of it for the Performance Modeling of Computer Programs.* PhD thesis, 2007.

[JLO90] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.

[JLZW09] S. Jiang, X. Li, Q. Zheng, and L. Wang. Approximate equal frequency discretization method. In *Global Congress on Intelligent Systems (GCIS)*, pages 514–518, 2009.

[JOL89] Finn V. Jensen, Kristian G. Olesen, and Steffen L. Lauritzen. Bayesian updating in recursive graphical models by local computations. Technical Report R 89-15, Aalborg University (DK), 1989.

[Ker92] R. Kerber. ChiMerge: Discretization of numeric attributes. In *AAAI Conference on Artificial Intelligence*, pages 123–128, 1992.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning.* The MIT Press, 2009.

[KK99] W. Kwedlo and M. Krętowski. An evolutionary algorithm using multivariate discretization for decision rule induction. In *Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 392–397, 1999.

[KKR13] Keiji Kanazawa, Daphne Koller, and Stuart J. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. *CoRR*, abs/1302.4965, 2013.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning (ICML)*, pages 284–292, 1996.

[LB94] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

[LMCL12] J. Lemeire, S. Meganck, F. Cartella, and T. Liu. Conservative independence-based causal structure learning in absence of adjacency faithfulness. *International Journal of Approximate Reasoning (IJAR)*, 53(9):1305–1325, 2012.

[Luo06] W. Luo. Learning Bayesian networks in semi-deterministic systems. In *The Canadian Conference on Artificial Intelligence*, volume 4013 of *Lecture notes in computer science*, pages 230–241, 2006.

[LW89] S.L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17(1):31–57, 1989.

[Mar03] Dimitris Margaritis. *Learning Bayesian network model structure from data.* PhD thesis, US Army, 2003.

[MC98] S. Monti and G.F. Cooper. A multivariate discretization method for learning Bayesian networks from mixed data. In *Uncertainty in Artificial Intelligence (UAI)*, pages 404–413, 1998.

[MC99] S. Monti and G.F. Cooper. A latent variable model for multivariate discretization. In *Association for Information Systems (AIS)*, pages 249–254, 1999.

[Mee95] Christopher Meek. Causal inference and causal explanation with background knowledge. In *Uncertainty in Artificial Intelligence (UAI)*, pages 403–410. Morgan Kaufmann Publishers Inc., 1995.

[MGJCC14] Ahmed Mabrouk, Christophe Gonzales, Karine Jabet-Chevalier, and Eric Chojnaki. An Efficient Bayesian Network Structure Learning Algorithm in the Presence of Deterministic Relations. In *European Conference on Artificial Intelligence (ECAI)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 567–572, August 2014.

[MJ99] A.L. Madsen and F.V. Jensen. LAZY propagation: A junction tree inference algorithm based on lazy inference. *The journal of Artificial Intelligence Research*, 113(1–2):203–245, 1999.

[MRS01] S. Moral, R. Rumi, and A. Salmeron. Mixtures of truncated exponentials in hybrid Bayesian networks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, volume 2143 of *Lecture Notes in Artificial Intelligence*, pages 156–167, 2001.

[Nil98] D Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8(2):159–173, 1998.

[PD04] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *The Journal of Artificial Intelligence Research*, 21:101–133, 2004.

[Pea87] Judea Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence Journal*, 32(2):245–257, 1987.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffmann, 1988.

[Pea00] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, Cambridge, England, 2000.

[PP86] J. Pearl and A. Paz. Graphoids: Graph-based logic for reasoning about relevance relations or when would x tell you more about y if you already know z? In *European Conference on Artificial Intelligence (ECAI)*, pages 55–61, Brighton, UK, 1986.

[PV91a] J. Pearl and T.S. Verma. A theory of inferred causation. In *Principles of Knowledge Representation and Reasoning (KR)*, 1991.

[PV$^+$91b] Judea Pearl, Thomas Verma, et al. *A theory of inferred causation*. Morgan Kaufmann San Mateo, CA, 1991.

[Rat03] C. Ratanamahatana. CloNI: Clustering of sqrt(n)-interval discretization. In *nternational Conference on Data Mining Including Building Application for CRM & Competitive Intelligence*, 2003.

[RdMAC08] S. Rodrigues de Morais, A. Aussem, and M. Corbex. Handling almost-deterministic relationships in constraint-based Bayesian network discovery: Application to cancer risk factor identification. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 101–106, 2008.

[RZS06] Joseph Ramsey, Jiji Zhang, and Peter Spirtes. Adjacency-faithfulness and conservative causal inference. In *Uncertainty in Artificial Intelligence (UAI)*, pages 401–408, 2006.

[RZWL13] C. Ruichu, H. Zhifeng, W. Wen, and W. Lijuan. Regularized Gaussian mixture model based discretization for gene expression data association mining. *Applied intelligence*, 39(3):607–613, 2013.

[S$^+$78] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[SBK05] T. Sang, P. Beame, and H. A. Kautz. Performing Bayesian inference by weighted model counting. In *AAAI Conference on Artificial Intelligence*, 2005.

[Scu09] Marco Scutari. Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*, 2009.

[SEHK11] D. Song, C.H. Ek, K. Huebner, and D. Kragic. Multivariate discretization for Bayesian network structure learning in robot grasping. In *The*

International Conference on Robotics and Automation (ICRA), pages 1944–1950, 2011.

[SGS01] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Bradford Book, 2nd edition, 2001.

[Sha86] R.D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

[Sha96] G. Shafer. *Probabilistic expert systems*. SIAM, 1996.

[She97] P.P. Shenoy. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning (IJAR)*, 17(1):1–25, 1997.

[SP90] Ross D. Shachter and Mark A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligence (UAI)*, UAI '89, pages 221–234, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[SSM$^+$96] P. Spirtes, R. Scheines, C. Meek, T. Richardson, C. Glymour, H. Hoijtink, and A. Boomsma. *TETRAD 3: Tools for Causal Modeling – User's Manual*, 1996.

[ST99] Harald Steck and Volker Tresp. Bayesian belief networks for data mining. In *Workshop on Data Mining and Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme*, pages 145–154. Citeseer, 1999.

[SW11] P.P. Shenoy and J.C. West. Inference in hybrid Bayesian networks using mixtures of polynomials. *International Journal of Approximate Reasoning (IJAR)*, 52(5):641–657, 2011.

[TAS03] Ioannis Tsamardinos, Constantin F Aliferis, and Alexander Statnikov. Time and sample efficient discovery of markov blankets and direct causal relations. In *ACM SIGKDD International conference on Knowledge Discovery and Data Mining*, pages 673–678. ACM, 2003.

[TASS03] Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale markov blanket discovery. In *Florida Artificial Intelligence Research Society Conference (FLAIRS)*, volume 2, 2003.

[TBA06] I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *The Journal of Machine Learning Research*, 65(1):31–78, 2006.

[TK05] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 584–590, Edinburgh, Scottland, UK, July 2005.

[vDvdGT03a] Steven van Dijk, Linda C. van der Gaag, and Dirk Thierens. A skeleton-based approach to learning bayesian networks from data. In *Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 132–143, 2003.

[vDVDGT03b] Steven van Dijk, Linda C Van Der Gaag, and Dirk Thierens. A skeleton-based approach to learning bayesian networks from data. In *Knowledge Discovery in Databases: PKDD*, pages 132–143. Springer, 2003.

[WL04] Man Leung Wong and Kwong Sak Leung. An efficient data mining method for learning bayesian networks using an evolutionary algorithm-based hybrid approach. *IEEE Transactions on Evolutionary Computation*, 8(4):378–404, 2004.

[XG08] Xianchao Xie and Zhi Geng. A recursive method for structural learning of directed acyclic graphs. *J. Mach. Learn. Res.*, 9:459–483, jun 2008.

[YM05] Sandeep Yaramakala and Dimitris Margaritis. Speculative markov blanket discovery for optimal feature selection. In *International conference on Data Mining (ICDM)*, pages 4–pp. IEEE, 2005.

[ZRR98] D.A. Zighed, S. Rabaséda, and R. Rakotomalala. FUSINTER: a method for discretization of continuous attributes. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(03):307–326, 1998.