Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Master in Data Science

Master Thesis

Facilitating the Inference Interpretation in Bayesian Networks

Author: Marta Alonso Tubía

Madrid, 07-2023

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

Master Thesis
Master in Data Science

Title: Facilitating the Inference Interpretation in Bayesian Networks
07-2023

Author:	Marta Alonso Tubía
Supervisors:	Concha Bielza Lozoya y Pedro Larrañaga Múgica
	Departamento de Inteligencia Artificial
	ETSI Informáticos
	Universidad Politécnica de Madrid

Resumen

El objetivo de este trabajo es contribuir a salvar un obstáculo importante en la adopción de la inteligencia artificial en la nuestra vida diaria: la falta de transparencia e interpretabilidad de sus modelos. Este asunto es clave en sectores como la justicia o la sanidad. Un sistema inteligente que proporcione al usuario explicaciones sobre cómo llegó a la conclusión, ya sea correcta o incorrecta, no solamente será más confiable, sino que es necesario para cumplir con la legislación europea vigente.

Las redes bayesianas destacan sobre otros paradigmas del *machine learning* por presentar características que las hacen particularmente interpretables: la rápida visualización de las relaciones entre variables en un grafo, y la posibilidad de todo tipo de razonamientos: predictivo, diagnóstico, abductivo, contrafáctico. . . .

Existe una amplia variedad de casuísticas en las cuales las explicaciones pueden ser útiles, así como un extensa categorización de explicaciones. Este trabajo tiene como objetivo presentar nuevas propuestas que puedan mejorar la interpretabilidad en la inferencia probabilística con redes Bayesiana. En primer lugar abordaremos los dos métodos exactos más populares (eliminación de variables y árbol de cliqués), centrándonos en describir el proceso de razonamiento con especial hincapié en el orden de eliminación de variables. En segundo lugar abordaremos los métodos de inferencia aproximada, centrándonos en justificar la convergencia del muestreo de Gibbs.

Seguiremos con otro tipo de inferencia, la inferencia abductiva, para presentar una nueva pregunta que puede ser de interés para el usuario en el caso concreto de datos Gaussianos, y su posible resolución usando algoritmos de estimación de distribución (EDAs). Para ello, en primer lugar se ha adaptado y extendido la librería *EDAspy* y se han aplicado la metodología a dos conjuntos de datos distintos, con su respectiva interpretación de resultados.

Por último, dada la ausencia de software para el cálculo de k explicaciones más relevantes dada una evidencia (k-MRE), se llevará a cabo la implementación, con funcionalidad añadida para una mejor comparativa de las explicaciones más relevantes. En concreto, se lleva a cabo una representación con coordenadas paralelas de las diferentes explicaciones, y se ofrece una métrica para medir la diversidad de las diferentes explicaciones con respecto a la explicación más relevante.

Abstract

The objective of this work is the contribution to overcoming the main obstacle in the adoption of artificial intelligence in our daily life: the lack of transparency and interpretability. This issue is key in sectors such as justice or health. An intelligent system that provides the user with explanations of how the conclusion was reached, whether correct or incorrect, will not only be more reliable, but is necessary to comply with current European legislation.

Bayesian networks stand out from other machine learning paradigms for presenting characteristics that make them particularly interpretable: the visualization of the relationships between variables at a glance in the graph, and the possibility of all kinds of reasoning: predictive, diagnostic, abductive, counterfactual. . . .

There is a wide variety of cases in which explanations can be useful, as well as extensive categorization of explanations. This work aims to present new proposals that can improve the interpretability in probabilistic inference in Bayesian networks. First, we will address the two most popular exact methods (Variable elimination and Junction tree clustering algorithm), focusing on describing the reasoning process with special emphasis on the order of variable elimination. Secondly, we will address the approximate inference methods, focusing on justifying the convergence of Gibbs sampling.

We will continue with another type of inference, the abductive inference. The goal is to present a new question that may be of interest to the user in the specific case of Gaussian data, and its possible resolution using Estimation of Distribution Algorithms (EDAs). Firstly, the *EDAspy* library has been adapted and extended. Secondly, the methodology has been applied to two different data sets, with their respective interpretation of results.

Finally, given the absence of software for calculating the k most relevant explanations given some evidence (k-MRE), the implementation will be carried out, with added functionality for a better comparison of the most relevant explanations. In particular, we carry out a representation with parallel coordinates of the different explanations, and offer a metric to measure the diversity of the different explanations with respect to the most relevant explanation.

Acknowledgments

To my supervisors, Concha Bielza and Pedro Larrañaga, for their expert advice and for promoting a creative and accurate work. Thanks to their instructions I have been able to improve my research skills.

To my colleagues at the Computational Intelligence Group, for the fruitful discussions that helped in the development of my master thesis.

To my parents, for always supporting me in everything I do, and for making all the necessary efforts to give me an education.

This work was partially supported by the Spanish Ministry of Science and Innovation through the PID2019-109247GB-I00 and TED2021-131310B-I00 projects.

Contents

Nomenclature

AI Artificial Intelligence

BN Bayesian Network

$CPDs$ Conditional Probability Distributions

DAG Directed Acyclic Graph

EAs Evolutionary Algorithms

$EDAs$ Estimation of Distribution Algorithms

GAs Genetic Algorithms

GBF Generalized Bayes Factor

MAP Maximum a Posteriori assignment

MPE Most Probable Explanation

MRE Most Relevant Explanation

SHD Structural Hamming Distance

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

We are entering the Fourth Industrial Revolution characterized by the combination of artificial intelligence (AI) and automated machines. This technological advancement has the potential to make an impact on our lives. However, there has been little AI adoption in sectors like health care, justice or finance (Jordan and Mitchell (2015)). In general, a key obstacle against the use of AI-based systems is that they often lack transparency and interpretability: machine learning models can be obscure and difficult for people to understand. Sometimes, AI is designed to achieve a high performance on a given task, at the cost of interpretability. This problem can be tackled using the so called explainable artificial intelligence (XAI), which is the antithesis of black-box systems in AI. XAI (Gunning (2017)) makes progress in the development of systems in which the high level of performance is combined with a proper understanding of its behavior (Figure 1.1).

Figure 1.1: Goals of XAI: Performance and explainability. Source: Gunning (2017).

The need for interpretability arises when for certain problems it is not enough to get the prediction: the model must also explain how it came to the prediction, because a correct prediction only partially solves your original problem. The following reasons

drive the demand for interpretability and explanations (Doshi-Velez and Kim (2017), Molnar (2019)):

- **Human curiosity and learning**, for example when we have unexpected events.
- We want to understand the **inconsistencies** between elements of our **knowledge structures and the reality**. For example, if a machine learning model rejects a loan application, the applicant needs an explanation to understand this unexpected event.
- In real-world tasks that require **safety measures** and testing. If we are confident about the abstraction of the system, we can anticipate to solve the weak points of the reasoning and foresee any possible problem.
- To increase **social acceptance** for the integration of machines and algorithms into our daily lives.
- To allow debugging and audition during the research and development phase as well as after deployment for the interpretation of erroneous predictions.
- To use it as a debugging tool for **detecting bias** in machine learning models.

The new legal restrictions have become an extra motivation for the development of XAI in the European framework. In 2018, a series of guidelines were published to achieve a trustworthy AI (High-Level Expert Group on Artificial Intelligence (2019)), where transparency is a key requirement. Moreover, the General Data Protection Regulation (GDPR), applicable since 2018, establishes the *right to explanation*. This means that individuals need to be provided with a meaningful explanation of the logic behind the AI system, as well as the possible consequences of the data processing. Recently in June 2023, the European Parliament has approved to negotiate the first law on AI in the world, which seeks to strengthen the right of citizens to demand explanations of the decisions of high-risk AI systems (Parliament (2023)).

Many specific machine learning paradigms can be considered interpretable in different ways (classification trees, rule induction, the k-nearest neighbor classifier, logistic regression...). However, Bayesian networks present unique characteristics that make them suitable for reaching a high level of interpretability:

- Bayesian networks use a graphical model that visually represents the probabilistic dependencies between variables, making it easier for humans to comprehend the model.
- Bayesian networks are a suitable tool to model uncertainty using probability theory (Pearl (1988)). By incorporating probabilities, Bayesian networks can provide a measure of confidence associated with their predictions.
- Bayesian network's network structure, both the conditional independence assertions it makes and the associated factorization of the joint distribution, allows tractable inference even in complex networks (Koller and Friedman (2009)).
- Bayesian networks allow for causal reasoning (Jensen and Nielsen (2013)) enabling analysts to understand the cause-and-effect relationships.
- Bayesian networks support incremental learning (Koller and Friedman (2009)), which means they can be updated with new data. This allows the incorporation

of new evidences, and the user can analyse the resulting changes in the network structure or probabilities.

- Bayesian networks can incorporate prior knowledge through the specification of prior probability distributions or prior relationships in the structure (Darwiche (2009), Koller and Friedman (2009)). Therefore any experts can adjust the probabilities and structure of the network, so that they align with their prior knowledge.

In summary, Bayesian networks are useful for making inferences because they allow modeling uncertainty, updating beliefs, reasoning causally, making informed decisions, and addressing complex problems. These characteristics provide the basis for developing interpretable probabilistic reasoning (inference) algorithms.

With everything into account, this work aims to enhance and increase the interpretable solutions in the inference with Bayesian networks.

1.1 Objectives

The objective of this work is to contribute to show that inference in Bayesian networks is interpretable. To accomplish this high level goal, we established different subobjectives:

- Literature review to be aware of the state-of-the-art situation.
- Identify proposals that are both incomplete or that admit improvements, or clear gaps in the literature.
- Review current implementations to extend their functionality with new proposals for interpretability in Bayesian inference.
- Weekly discussion of the new proposals with the supervisors of this work.
- Conclusion about what we have accomplished in this work and what is still to be done in the future.

1.2 Contents

The contributions of this work are organized as follows:

- First of all, we present in this chapter the different motivations that exist to continue working on interpretability in general, and the reasons why we focus on Bayesian networks.
- In Chapter 2, we review the state of the art related to our proposal. In Section 2.1 the fundamental concepts of Bayesian networks are introduced. Section 2.2 explains the basic properties that all explanations must have to be considered a good explanation. Section 2.3 addresses the issue of inference. In the first place, the exact and approximate algorithms are explained to calculate the posterior probability of groups of variables given some evidence, placing special emphasis on the steps that are going to be chosen for the explanations of the model reasoning. Then, section 2.3.3.1 makes a bibliographic review of other types of queries: maximum a posteriori (MAP)/ most probable explanation (MPE) and

most relevant explanation (MRE), identifying the knowledge gaps and highlighting the aspects that have inspired this work.

- Next, in Chapter 3, we develop our new proposals and apply the methodologies to different datasets. Our goal is to improve the interpretability in the different types of inference. Regarding the algorithms that compute conditional probabilities, we focus on extending the *pgmpy* library (the most complete library on Bayesian network inference) to create visualizations that help understand the exact algorithm steps. Regarding approximate algorithms, our contribution focuses on creating a function that: 1. combines *pgmpy* (python) with another library (R) to offer different convergence diagnostics (both numerical and graphic) for the inference with Gibbs sampling, to increase the reliability of the results; 2. allow making inferences with likelihood weighting. Regarding abductive inference, we have posed a question that may be of interest to the user, which combines concepts of sensitive analysis and counterfactual reasoning, and we have solved it using a type of evolutionary algorithm. Finally, we have implemented an approximate algorithm to calculate what is known as the k most relevant explanations (k-MRE) given evidence. The proposal also includes a visualization and a metric to understand and compare the k explanations.
- Finally, in Chapter 4, we draw our conclusions and present our future lines of research that will expand the proposals presented in this work.
- Additionally in the annex, implementation comments are added to clarify the code and the interaction with the user. We add tabular results that are part of the results section.

Chapter 2

State of the art

2.1 Bayesian networks

A Bayesian network (BN) (Koller and Friedman (2009); Pearl (1988)), is a tuple $\mathcal{B} = (\mathcal{G}, \theta)$, where $\mathcal{G} = (V, A)$ is a directed acyclic graph (DAG) with a set of nodes $V = \{X_1, \ldots, X_n\}$ and a set of arcs $A \subseteq V \times V$. A Bayesian network represents the probability distribution, $P(\mathbf{x})$ of a multivariate random variable $\mathbf{X} = (X_1, ..., X_n)$. The set $\theta = \{P(x_i|\mathbf{X}_{Pa_{X_i}})\}$ defines a conditional probability distribution (CPD) for each node of the graph, where Pa_{X_i} is the set of parents of X_i in the graph $\mathcal{G}$. This allows to represent the joint probability distribution $P(\mathbf{x})$ as:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i|\mathbf{X}_{Pa_{X_i}}). \tag{2.1}$$

Bayesian networks provide a sound formalism for probabilistic reasoning under uncertainty.

2.1.1 Gaussian Bayesian networks

A Gaussian Bayesian network (GBN) (Shachter and Kenley (1989), Geiger and Heckerman (1994)) is a BN where the joint probability density associated with the variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a multivariate normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ given by

$$f(\mathbf{x}) = (2\pi)^{-\frac{n}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\}, \tag{2.2}$$

where $\boldsymbol{\mu}$ is n-dimensional mean vector and $\boldsymbol{\Sigma}$ is the $n \times n$ positive definite covariance matrix. The conditional probability density for X_i $(i = 1, \ldots, n)$ that satisfies equation (2.1) is the univariate normal distribution given by

$$f(Xi|Pa_{X_i}) \sim N\left(\mu_i + \sum_{j=1}^{i-1} \beta_{ji}(x_j - \mu_j), v_i\right),$$

where μ_i is the mean of X_i, β_{ji} is the regression coefficient of $X_j \in Pa_{X_i}$ when X_i is regressed on its parents, and v_i is the conditional variance of X_i given its parents.

The covariance matrix $\mathbf{\Sigma}$ can be got from v_i and β_{ji} using the algorithm proposed in Shachter and Kenley (1989):

$$\mathbf{\Sigma} = (\mathbf{I} - \mathbf{B})^{-1}\mathbf{D}(\mathbf{I} - \mathbf{B})^{-1}, \tag{2.3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{D}$ is a diagonal matrix with the conditional variances v_i and $\mathbf{B}$ is a strictly upper triangular matrix with the regression coefficients β_{ji} where X_j is a parent of X_i for the variables in $\mathbf{X}$ with $j < i$.

2.2 Explanation in Bayesian networks

2.2.1 Introduction

First of all, we are going to briefly discuss the meaning and distinction of the terms *explainable* and *interpretable*, since there is a wide debate about whether they are equivalent or not.

Acoording to Miller (2019), the interpretability of a model is defined as the degree to which an observer can understand the cause of a decision. Miller (2019) adopts the assertion that explanation is post-hoc interpretability. Explanation is thus one mode in which an observer may obtain understanding, but there are additional modes that one can adopt. For practical purposes, the author equates *interpretability* with *explainability*.

According to Montavon et al. (2018), an interpretation is the mapping of an abstract concept (e.g., a predicted class) into a domain that the human can make sense of. An example of a domain that is interpretable is images (arrays of pixels). On the other hand, an explanation is the collection of features of the interpretable domain, that have contributed for a given example to produce a decision (e.g., classification or regression). An explanation can be, for example, a heatmap highlighting which pixels of the input image most strongly support the classification decision. Therefore, both concepts are not equivalent.

Due to the intrinsic characteristics of Bayesian networks presented in Section 1, the boundary between both concepts becomes more diffuse, and we will treat them as equivalent.

Moving the focus away from AI to take a more general view of explainability, an explanation can be defined as (Lacave and Díez (2002)):

Definition 2.2.1 *An explanation is the exposition of something in such a way that it is understandable for the receiver of the explanation, which implies that he/she improves his/her knowledge about the object of the explanation; and satisfactory as far as it covers the receiver's expectations.*

In the field of expert systems, the best explanation varies across the different authors. However, regardless of the type, they have to fulfill some properties which are classified into three categories according to Lacave and Díez (2002), and are briefly described below.

2.2.2 Properties of an explanation

The properties of an explanation can be classified into three categories:

Content

It depends on:

1. Focus of explanation:
   There are three basic issues that can be explained: the **model** (consists of displaying, verbally, graphically or in frames, the information contained in the knowledge base. For example, graphical modeling tools such as GeNIe, Elvira and SamIam have funcionalities for visualizing the strength of the probabilistic relations between the variables in a domain.); the **reasoning process** performed by the system to obtain (or not) a conclusion and **the evidence** propagated if any. This last explanation consists of determinig which values of the unobserved variables justify the available evidence. This process is usually called abduction (for example, MPE or k most probable explanation).
2. Purpose: description (the underlying knowledge base, the conclusions or intermediate results) and comprehension (the conclusions and the relations between them).
3. Level of explanations: at micro level (variations produced in a particular node as a consequence of the variation in its neighbors) or macro level (main lines of reasoning that lead from the evidence to a certain conclusion).

Communication

The way in which the explanation is offered to the user. This depends, basically, on the methods of interaction between the user and the system and the manner in which it is presented. We can distinguish how explanations are presented to the user: verbally or graphically or both.

Adaptation

One of the key features of an effective explanation is the ability to address each user's specific needs and expectations, which essentially depends on the knowledge he/she has about the domain and the reasoning method. It is also possible to set the level of detail.

2.3 Inference in Bayesian networks

Inference refers to finding the probability of any variable conditioned on a given evidence or fixed values of some of the variables in the Bayesian network. Inference also refers to finding values of a set of variables that best explain the observed evidence. This is called abductive inference.

The previous inference types can be carried out with exact inference or with approximate inference methods. Exact inference includes the brute-force approach, the variable elimination algorithm (Zhang and Poole (1994)) and the message passing algorithm (Lauritzen and Spiegelhalter (1988a)). Exact inference is tractable for many real-world applications. However, it is limited by its worst-case exponential performance. In such cases approximate inference is the only alternative that will produce any result at all. Probabilistic logic sampling (or forward sampling) (Henrion (1988)), likelihood weighting (Fung and Chang (1990)), and Gibbs sampling (Pearl (1997)) are the most popular approximate methods for inference in Bayesian networks.

2.3.1 Exact inference

2.3.1.1 Variable elimination

The variable elimination algorithm keeps a list of potentials (local distributions involving a variable and its parents) and eliminates all variables one by one following an ordering and operating over the potentials.

Definition 2.3.1 *A factor is a function from a set of random variables to a number.*

When we manipulate the potentials, we obtain a new function called factor. The algorithm works using two fundamental operations over factors: multiplying them and summing out variables.

An ordering involving a reduced number of operations (multiplications and additions) is essential for this method. Finding the optimal (minimum cost) elimination ordering is an NP-hard problem (Bertelè and Brioschi (1972)), and many heuristics have been proposed to find good orderings. However, none stands out above the others (Koller and Friedman (2009)). This project includes a software proposal that integrates the comparison of these heuristics using the *induced-width*. The heuristics, and the *induced-width* will be explained in the following sections.

Elimination as graph transformation

As we have mentioned, the complexity of the algorithm has to do with the number of operations on the factors. This quantity is related to the structure of the underlying graph that induced the set of factors on which the algorithm is run.

First of all we define the notion of an undirected graph associated with a set of factors:

Definition 2.3.2 *Let Φ be a set of factors. We define*

$$Scope[\Phi] = \cup_{\phi \in \Phi} Scope[\phi]$$

to be the set of all variables appearing in any of the factors in Φ. We define $\mathcal{H}_\Phi$ to be the undirected graph whose nodes correspond to the variables in Scope[Φ] and where we have an edge $X_i - X_j \in \mathcal{H}_\Phi$ if and only if there exists a factor $\phi \in \Phi$ such that $X_i, X_j \in Scope[\Phi]$.

In this subsection we want to clarify this concept of *elimination as graph transformation* as operating on an undirected graph $\mathcal{H}$.

The algorithm input is the set of factors Φ and the undirected graph $\mathcal{H}$ obtained from $\mathcal{G}$ (the only relevant information from the BN structure in this algorithm is the undirected graph). When a variable X is eliminated, several operations take place. First, we create a single factor Ψ that contains X and all of the variables Y with which it appears in the factors. Then, we eliminate X from Ψ, replacing it with a new factor τ that contains all of the variables Y but does not contain X. Let Φ_X be the resulting set of factors that will be used in the subsequent elimination steps. How does the undirected graph $\mathcal{H}_{\Phi_X}$ differ from $\mathcal{H}_\Phi$? The step of constructing Ψ generates edges between all of the variables Y. Some of them were present in $\mathcal{H}_\Phi$, whereas others are introduced due to the elimination step and are called *fill edges*. The step of eliminating X from Ψ to construct τ has the effect to removing X and all of its incident edges from the graph (Figure 2.1).

Definition 2.3.3 *A graph $\mathcal{G}$ is said to be complete when all nodes are pairwise linked.*

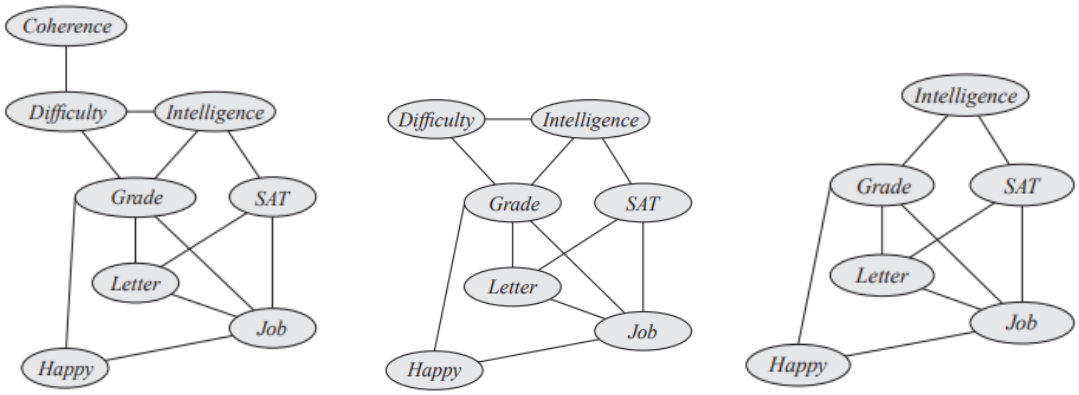


Figure 2.1: On the left, the undirected graph associated to the Student Bayesian network (Koller and Friedman (2009)). In the middle, graph transformation after eliminating variable *Coherence*. On the right, graph transformation after eliminating *Difficulty*.

Definition 2.3.4 *A subgraph over a set of vertices that is complete and maximal (it is not a subset of another complete set) is called a clique.*

We call *induced graph* to the union of all of the graphs resulting from the different steps of the variable elimination algorithm.

Every factor that appears in one of the steps in the algorithm corresponds to a complete subgraph of the induced graph, and is therefore a clique in the graph (Theorem 9.6 in Koller and Friedman (2009)). Therefore, there is a direct correspondence between the maximal factors generated by our algorithm and cliques in the induced graph, and both depend strongly on the elimination ordering. Thus, we can reformulate our goal of finding an elimination ordering so that the largest clique in the resulting graph is as small as possible. These considerations will help us later to establish a connection with the junction tree algorithm.

Definition 2.3.5 *We define the **width** of an induced graph to be the number of nodes in the largest clique in the induced graph minus 1. We define the **induced width** of an ordering relative to a graph $\mathcal{H}$ (directed or undirected) to be the width of the graph $\mathcal{I}_{\mathcal{H}}$ induced by applying variable elimination to $\mathcal{H}$ using the ordering. We define the **tree-width** of a graph to be its minimal induced width.*

We will use the induced width to compare between the different heuristic ordering-selections that are going to appear in the following section.

How to compute ordering elimination

Our goal is to construct an ordering that induces cliques as small as possible. While we cannot find an ordering that achieves the global minimum, we can eliminate variables one at a time, so that each step tends to lead to a small blow-up in size. At each point, the algorithm evaluates each of the remaining variables in the networks based on its heuristic cost function:

- **Min-neighbors**: The cost of a vertex is the number of neighbors it has in the current graph.
- **Min-weight**: The cost of a vertex is the product of weights (domain cardinality)

of its neighbours.

- **Min-fill**: The cost of a vertex is the number of edges that need to be added to the graph due to its elimination.
- **Weighted-min-fill**: The cost of a vertex is the sum of weights of the edges that need to be added to the graph due to its elimination, where a weight of an edge is the product of weights of its constituent vertices.

It can be shown (Koller and Friedman (2009)) that none of these criteria is universally better than others. Since the computational cost of the heuristic ordering-selection algorithms is usually negligible relative to the running time of the inference itself, it is worthwhile to run several heuristic algorithms in order to find the best ordering obtained by any of them.

2.3.1.2 Junction tree clustering algorithm

This work aims to illustrate each of the steps of this algorithm to make it more interpretable. This entails the creation of the junction tree (moralise, triangulate the moral graph, obtain the cliques, create the junction tree and its separators, compute the junction tree parameters), and the message passing algorithm for the calculation of probabilities.

The first step of the algorithm consists of the *moralization* of the network structure. This means that all variables with a common child are linked, after which all directions on the arcs are deleted. The second step of the algorithm is the so-called *triangulation* of the moral graph. A graph is triangulated if every minimal loop in the graph is of length tree. The basic technique for triangulation repeatedly eliminates nodes following an elimination ordering. Eliminating node X means: (1) adding edges so that all nodes adjacent to X become pairwise adjacent (if they are not), and (2) deleting node X and its adjacent edges. The added edges are called **fill-in-edges**. The addition of edges can lead to the loss of independence information implied by the graph structure, and hence an optimal triangulation should yield the fewest fill-in edges. Cliques can be retrieved from the fill-in process because, when eliminating a node X, edges are added to make the subgraph given by X and its neighbours complete. Lots of approaches, most of which are **based on heuristics**, have been proposed to find good triangulations. Here we present those that we will use through the *pgmpy* library:

Let $\boldsymbol{X}$ be the set of variables and X_i denotes the i-th variable. If we call

- S_i - The size of the clique created by deleting the variable.
- E_i - Cardinality of variable X_i.
- M_i - Maximum size of cliques given by X_i and its adjacent nodes.
- C_i - Sum of size of cliques given by X_i and its adjacent nodes.

Then, the heuristics are defined in the following way (Cano and Moral (1995)):

- H1 - Delete the variable with minimal S_i.
- H2 - Delete the variable with minimal S_i/E_i.
- H3 - Delete the variable with minimal $S_i - M_i$.

- H4 - Delete the variable with minimal $S_i - C_i$.
- H5 - Delete the variable with minimal S_i/M_i.
- H6 - Delete the variable with minimal S_i/C_i.

In the last step, a junction tree is built. A common and efficient approach for building the tree is the *maximum spanning tree algorithm*.

These first steps of the model (moralization, triangulation and junction tree building) are explained to the user by depicting the network (Chapter 3). In this work, this visual exploration is organized according to different layouts.

Finally, each potential in the BN is attached to a clique that contains its domain. Whenever there is more than one potential attached to the same clique, the potential at this clique is the product of those potentials. If a clique has no attached potential, we attach the constant function 1.

To conclude, once a clique tree is constructed, it can be used for inference using the message passage algorithm on top of the junction tree.

As can be read in Jensen (1996),

> *Since any elimination sequence will produce a triangulation it may not seem as a problem, but for the propagation algorithm it is. In probability propagation, the cliques in the junction graph shall have joint probability tables attached to them. The size of the table is the product of the number of states of the variables. So, the size increases exponentially with the size of the clique. A good triangulation, therefore, is a triangulation yielding small cliques; or to be more precise, yielding small probability tables.*

Wen (1991) demonstrated that the search for an optimal triangulation is NP-hard.

Larrañaga et al. (1997) proposed a way to search for the optimal node elimination sequence based on genetic algorithm. As we want to obtain small probability tables after the triangulation, our objective is to minimize the following function $w(G)$, defined as

$$w(\mathcal{G}) = log_2 \sum_{C} \prod_{v_i \in C} n_i, \tag{2.4}$$

where $\mathcal{G}$ is the graph, n_k is the number of states of vertex $v_k \in V$, which belongs to clique C. This metric will be used in this work to compare the elimination orders obtained by different heuristics.

When this process concludes, each clique contains the marginal probability over the variables in its scope. We can compute the marginal probability over a particular variable X by selecting a clique whose scope contains X, and eliminating the redundant variables in the clique. A key point is that the result of this process does not depend on the clique we have selected. That is, if X appears in two cliques, both must agree on its marginal. This state which can be reached using belief propagation, and we say that the clique tree is **calibrated**. This process is graphically shown to the user.

To conclude, given the analogy that we have presented between the elimination of variables and the transformation on the graph, it is interesting to compare both algorithms, variable elimination and junction tree algorithm. In principle, they both use the same basic operations of multiplying factors and summing out variables, and

both seek variable elimination ordering / triangulation leading to small cliques. So both have the same computational complexity. In practice, however, they offer different trade-offs. The junction tree clustering algorithm allows to perform a fraction of the operations in advance, and not during inference for any query: the choice of triangulation/elimination ordering and the product of the CPDs in a single clique. It also provides answers to multiple queries, at the expense, however, of increasing the space required to store all intermediate messages, which variable elimination does away with.

2.3.2 Approximate methods

The framework for this class of methods is as follows. Consider some distribution $P(\mathbf{X})$ over a set of variables $\mathbf{X}$ and assume we want to estimate the probability of some event $\mathbf{Y} = \mathbf{y}$ relative to P, for some $\mathbf{Y} \subseteq \mathbf{X}$ and some value assignment $\mathbf{y}$. We approximate this probability by generating a set of M instantiations to all or some of the variables in the network.

2.3.2.1 Forward sampling

Definition 2.3.6 *Let $\mathcal{G} = (V, A)$ be a graph. An ordering of the nodes $< X_1, ..., X_n >$ is a topological ordering relative to $\mathcal{G}$ if, whenever we have the directed arc $X_i \to X_j \in A$ then $i < j$ in the ordering.*

It is the simplest approach for the generation of samples. Here, we sample the nodes using the topological ordering of $\mathbf{X}$ in the BN, so that by the time we sample a node, we have values for all of its parents. We can then sample from the distribution defined by the CPD and by the chosen values for the node's parents.

The approximation of marginal probabilities like $P(\mathbf{y})$ is

$$\hat{P}(\mathbf{y}) = \frac{1}{M} \sum_{m=1}^{M} I(\mathbf{y}[m] = \mathbf{y}),$$

where we use $\mathbf{y}[m]$ to denote the assignment to $\mathbf{Y}$ in the m-th sample, and $I(\mathbf{y}[m] = \mathbf{y})$ is the indicator function.

In general, however, we are interested in conditional probabilities of the form $P(\mathbf{y}|\mathbf{E} = \mathbf{e})$. One approach is the method called *rejection sampling*. We generate samples $\mathbf{x}$ from $P(\mathbf{X})$ with *forward sampling*. We then reject any sample that is not compatible with $\mathbf{e}$. The problem, of course, is that the number of unrejected samples can be quite small.

2.3.2.2 Likelihood weighting

Regarding conditional probabilities $P(\mathbf{y}|\mathbf{E} = \mathbf{e})$, it seems much more sensible to simply force the samples to take on the appropriate values at observed nodes. That is, when we come to sampling a node E_i whose value has been observed, we simply set it to its observed value. The problem with this approach is that it fails to account for the fact that, in the standard forward sampling process, a sampled node Y_i parent of an evidence variable E_j could have a different conditional distribution when the information $E_j = e_j$ is known. If we follow for sampling the topological order, we are ignoring this key information.

When we have multiple observations and we want our sampling process to set all of them to their observed values, we need to consider the probability that each of the observation nodes, had it been sampled using the standard forward sampling process, would have resulted in the observed values. This is done by assigning weights to the samples. If we call $\mathbf{u}_i$ the value assignment to the parents of the evidence node E_i, $Pa(E_i)$, then the weigth for each evidence node is $P(e_i|\mathbf{u}_i)$.

The sampling events for each node in forward sampling are independent, and hence the weight for each sample should be the product of the weights induced by each evidence node separately

$$w = \prod_{e_i \in \mathbf{E}} P(e_i|\mathbf{u}_i).$$

The approximation $\tilde{P}(\mathbf{y}|\mathbf{e})$ to $P(\mathbf{y}|\mathbf{e})$ is

$$\tilde{P}(\mathbf{y}|\mathbf{e}) = \frac{\sum_{m=1}^{M} w[m] I(\mathbf{y}[m] = \mathbf{y})}{\sum_{m=1}^{M} w[m]},$$

where we use $\mathbf{y}[m]$ and $w[m]$ to denote the assignment to $\mathbf{Y}$ and weight in the m-th sample, respectively.

2.3.2.3 Gibbs sampling algorithm

One of the limitations of likelihood weighting is that an evidence node affects the sampling only for nodes that are its descendants. The effect on nodes that are nondescendants is accounted for only by the weights. In cases where much of the evidence is at the leaves of the network, we are essentially sampling from the prior distribution, which is often very far from the desired posterior.

We now present an alternative sampling approach that generates a sequence of samples. This sequence is constructed so that, although the first sample may be generated from the prior, successive samples are generated from distributions that provably get closer and closer to the desired posterior. It is called *Gibbs sampling algorithm*, and is classified as a Markov chain Monte Carlo method. This method starts out by generating a sample of the unobserved variables from some initial distribution. Starting from that sample, we then iterate over each of the unobserved variables, sampling a new value for each variable given our current sample for all other variables. This process allows information to "flow" across the network as we sample each variable. As we repeat this sampling process, the distribution from which we generate each sample gets closer and closer to the posterior $P(\mathbf{X}|\mathbf{e})$. This argument can be formalized using the framework called *Markov chain Monte Carlo (MCMC)*.

A key question is how many iterations we should perform before we can collect a sample as being (almost) generated from the posterior.

An approach to solve the convergence issue to the posterior distribution is through diagnostic tools of the output produced by the algorithm. Many diagnostic methods are known (Cowles and Carlin (1996)). One of the most popular among the statistical community is the Gelman and Rubin diagnosis. It is a univariate diagnosis for each of the variables.

Multiple chains sampling the same distribution should, upon convergence, all yield similar estimates. In addition, estimates based on a complete set of samples collected

from all of the chains should have variance comparable to variance in each of the chains. More formally, assume that K separate chains are each run for $T + M$ steps starting from a diverse set of starting points. After discarding the first T samples from each chain, let $\mathbf{X}_k[m]$ denote a sample from chain k after iteration $T + m$. We can now compute the B (between-chains) and W (within-chain) variances for variable $X^j, \quad j = 1, \ldots, n$. If we call $X_j^k[m]$ the instantiation of variable j in chain k after iteration $T + m$ (Koller and Friedman (2009)), then

$$\begin{aligned}
\bar{X_j^k} &= \frac{1}{M}\sum_{m=1}^{M} X_j^k[m], \\
\bar{X_j} &= \frac{1}{K}\sum_{k=1}^{K} \bar{X_j^k}, \\
B_j &= \frac{M}{K-1}\sum_{k=1}^{K}(\bar{X_j^k} - \bar{X_j})^2, \\
W_j &= \frac{1}{K}\frac{1}{M-1}\sum_{k=1}^{K}\sum_{m=1}^{M}(X_j^k[m] - \bar{X_j^k})^2.
\end{aligned}$$

The between chain variance relative to the within-chain variance should be about the same if all chains have settled. If we define $Z_j = \frac{M-1}{M}W_j + \frac{1}{M}B_j$, one measure of disagreement between chains is given by $R_j = \sqrt{\frac{Z_j}{W_j}}$. It is called the *potential scale reduction factor*. If this value is close to 1, the chains have all converged to the true distribution. A value higher than 1.1 is *cause for concern*.

There exist several multivariate versions of Gelman and Rubin's diagnostic. In Brooks and Gelman (1996) it is proposed a generalization of the univariate shrink factor that may be used as an approximate upper bound to the maximum of the univariate statistics over all variables.

Overall, the use of multiple chains demands a higher computational cost, but on the other side, it allows us to evaluate the extent to which our chains are mixing.

Geweke (1992) proposed another convergence diagnostic for Markov chains based on a test for equality of the means of the first and last part of a Markov chain. If the samples are drawn from the stationary distribution of the chain, the two means should be close and Geweke's statistic has an asymptotically standard normal distribution. The test statistic is a standard Z-score: the difference between the two sample means divided by its estimated standard error.

More convergence diagnostic tools are available in the R package *coda*.

In this project, we generate the samples using the python package *pgmpy* and test the convergence using both test of diagnosis, implemented in the available R package called *coda*. To communicate R with python I use as intermediary the *rpy2* package.

2.3.3 Queries in Bayesian networks

In addition to the two types of queries presented so far, marginal probabilities $P(\mathbf{x})$ and the posterior probability of a set of variables $\mathbf{Y} \subseteq \mathbf{X}$ given $\mathbf{e}$, $P(\mathbf{Y}|\mathbf{e})$, there are

other very interesting queries.

An important type of query is abduction, which consists of finding the most plausible explanation of certain variables of interest, $\mathbf{Y}_u \subseteq \mathbf{X}$ for evidence $\mathbf{e}$, i.e. find the value assignment $\mathbf{y}_u$ that maximises the posterior probability given the evidence:

$$\mathbf{y}_u = argmax_{\mathbf{Y}_u} P(\mathbf{Y}_u|\mathbf{e}).$$

The set $\mathbf{Y}_u \subseteq \mathbf{X}$ can be the complete set of unobserved variables, and we talk in this case about most probable explanation (MPE) or total abduction. However, if we are interested in only a subset of variables, this type of query is called partial abduction (MAP).

The explanations found by MAP or MPE are often overspecified because irrelevant target variables may also be included. MRE (see Section 2.3.3.2) is a method developed to address the limitations of MAP and MPE. Its main idea is to find a partial instantiation of target variables that maximizes the generalized Bayes factor as the explanation.

Both MPE and MRE are covered in more detail in the following sections.

2.3.3.1 Abductive inference

We have made a bibliographic review that focuses on the interpretability proposals and the approximate algorithms for abductive queries: contrastive-counterfactual explanations (Koopman (2020)), evolutionary algorithms and sensitive analysis. These ingredients have inspired the proposal of this work. Our work focuses on studying what knowledge gaps exist and what new proposals can help the user to better understand the result and gain confidence about it. Based on this, we develop the methodology of our proposal in the next chapter.

Contrastive-counterfactual explanations

Recently, Koopman (2020) elaborated another possibility to add explainability, slightly different from counterfactuals, called *contrastive-counterfactual explanations*. First of all, a contrastive-counterfactual explanation contains a *sufficient* explanation: given the observations for the variables in this explanation, the other variables could have an arbitrary value without changing the most probable value of the target. Secondly, the contrastive-counterfactual explanation should have a *counterfactual* explanation: this explanation states what minimal changes should be applied to the observations for the target to result in the expected value of the user. The combination of the sufficient and the counterfactual explanation give a contrast between the expected and most probable value of the target. In addition the author makes a proposal of how this could be computed from a Bayesian network.

Evolutionary algorithms (EAs) for abductive inference

In general, solving a problem, can be perceived as a search through a space of potential solutions. Since usually we are after *the best* solution, we can view this task as an optimization process. For large spaces, AI techniques must be employed and evolutionary algorithms are among such techniques.

Evolutionary computation is a discipline that has been emerging for many decades and has reached a stage of some maturity. It has been widely used in science and en-

gineering. All methods within this discipline, as population-based metaheuristics, are characterized by maintaining a set of possible solutions (individuals) to make them successively evolve to fitter solutions generation after generation. The best known algorithms in this class include genetic algorithms, evolutionary programming, evolution strategies and genetic programming (Dasgupta and Michalewicz (1997)). In particular, the behavior of some evolutionary computation algorithms (for example, genetic algorithms (GAs) (Holland (1975))), depends on several parameters associated with them (crossover and mutation operators, probabilities of crossover and mutation, size of the population, rate of generational reproduction, number of generations, etc.).

Abductive inference is known to be NP-hard (Koller and Friedman (2009)), so exact computation is not always possible. Genetic algorithms have been successfully applied to give an approximate algorithm for both types of abductive inference problems: total (Gelsema (1995, 1996); Rojas-Guzman and Kramer (1996)) and partial (de Campos et al. (1999)). In Gelsema's algorithm, a chromosome is a configuration of the unobserved variables ($\mathbf{Y}_u = X \backslash \mathbf{E}$), i.e., a string of integers (associated to the discrete values), and we want to maximize $P(\mathbf{Y}_u, \mathbf{E})$. Dealing with partial abductive inference ($\mathbf{Y}_u \subset \mathbf{X}$) using GAs appears easier than that of dealing with total abduction, because the size of the search space in the partial case is considerably smaller than in the total abductive inference problem. However, this is not the case, because of the increasing complexity of the evaluation function: in the partial case, we have to marginalize the variables that are not present in the variables of interest and $\mathbf{E}$. For this reason, in de Campos et al. (1999) the fitness $P(\mathbf{y}_u|\mathbf{E})$ of a chromosome $\mathbf{y}_u$ is computed by using probability propagation over a clique tree.

Traditional EAs present an additional disadvantage: they do not explicitly consider dependencies among the variables involved, and hence, solutions reproduced by the algorithm are not able to exploit the information found in the data. This limitation is overcome by using another type of EAs called estimation of distribution algorithms (EDAs) (Mühlenbein and Paass (1996)). In EDAs there are neither crossover nor mutation operators. Instead, the new population of individuals is sampled from a probability distribution, which is estimated from a dataset containing selected individuals from the previous generation. In EDAs the interrelations are explicitly expressed through the structure of the BN that factorizes the joint probability distribution associated with the individuals selected at each iteration. In Campos et al. (2002), both GA and EDAs are compared with respect to different parameters.

In EDAs, three steps,

1. selecting some individuals from a population of individuals,
2. learning the joint probability distribution of the selected individuals and
3. sampling from the learnt distribution,

are repeated until a -previously established- stopping criterion is met (for example, when a fixed number of generations or a fixed number of different evaluated individuals are achieved, no improvement with regard to the best individual obtained in the previous generation, etc.). We can apply EDAs to carry out abductive inference, specifically, to calculate the MPE given evidence. This technique will be key to developing our proposal in the next chapter.

Depending on the type of variables that the optimization problem deals with, a discrete or a continuous EDA is used. When dealing with continuous variables, it is common to assume an underlying probability distribution. The usual choice is a Gaussian distribution. As discrete EDAs were studied before continuous EDAs, most of the first continuous EDAs were adaptations of discrete EDAs to continuous environments. An example of these algorithms is the estimation of Gaussian networks algorithm (EGNA) (Larrañaga (2002)), which adapts the discrete EDA estimation of Bayesian network algorithm (EBNA), to continuous environments using GBNs.

EDAspy is a Python package that implements EDAs (Soloviev (2020)). In Soloviev et al. (2022), the package was used to solve industrial optimization problems.

Sensitivity analysis

Castillo and Kjaerulff (2003) deal with the problem of sensitivity analysis in GBNs. Chan and Darwiche (2006) raise the question *How much change in a single network parameter can we afford to apply while keeping the MPE unchanged?*. Chan and Darwiche (2002) characterize the changes in the parameters that are relevant or not, and lower and upper bounds are given on the queries of interest $P(\mathbf{y}|\mathbf{e})$ depending on the changes in the parameters. Leonelli and Riccomagno (2022), pose the situation in which the modeler has some idea of what the value of the probability of interest should be (assume a value δ) and ask what the changes in the conditional probabilities should be done so that the probability takes the value δ. In addition, Leonelli et al. (2021) illustrate the use of the R package *bnmonitor* to carry out sensitivity analysis on Bayesian networks. In Görgen and Leonelli (2020) the authors warn that, for Gaussian graphical models, variations on the parameters often lead to an inconsistent representation of the conditional independences structures of the model (i.e. the perturbation may break its conditional independences). Therefore the authors propose a new class of limited perturbations of Gaussian vectors, called model-preserving.

Knowledge gap detected and proposal introduction

During the previous review of existing interpretability proposals in abductive queries, a knowledge gap was identified: using EDAs to check how the Bayesian network has to change (structure and parameters) so that the MPE probability exceeds a certain value fixed by the user, and the comparison of the original network and the network obtained after the optimization process. This new proposal therefore combines *expert knowledge* with ideas inspired by sensitivity analysis and evolutionary algorithms. In Leonelli et al. (2021), a sensitivity analysis has been carried out to modify the probability of the MPE to the requirements of a user. However, we do not want to restrict ourselves to the disturbance of a specific parameter within the limits that allow us to maintain the conditional independence of the model, but rather a greater flexibility in the disturbance. Now both the structure and the parameters can change, and we want to give meaning to the changes in the model that satisfy the probability change required by the user. In other words, we will analyze the qualitative relationships that have appeared and disappeared in the network (after the network perturbation), as well as the quantitative variations in the mean and the covariance matrix.

Furthermore, most of the efforts regarding calculating the MPE using EAs, and in sensitivity analysis, have focused on discrete datasets. In our proposal we are going

to focus on Gaussian data. In chapter 3 we settle on the details of our new approach.

2.3.3.2 Most relevant explanation (MRE)

The approaches to finding abductive inference explanations MAP and MPE may generate explanations that are either too simple (underspecified) or too complex (overspecified). It is desirable to find the most relevant contributing factors. This motivates the following definitions.

Definition 2.3.7 *Given a set of target variables $\boldsymbol{M}$ with explanatory interest in a Bayesian network and partial evidence $\boldsymbol{e}$ in the remaining variables, an explanation for the evidence is a joint instantiation $\boldsymbol{x}$ of a non-empty subset $\boldsymbol{X}$ of the target variables, i.e. $\emptyset \subset \boldsymbol{X} \subseteq \boldsymbol{M}$.*

Various pruning techniques have been used to avoid overly complex explanations: pre-pruning and post-pruning. Pre-pruning methods use the context-specific independence relations represented in Bayesian networks to prune irrelevant variables before applying methods such as MAP to generate explanations. However, these independence relations are too strict and are unable to prune marginally or loosely relevant target variables. In contrast, post-pruning methods first generate explanations using methods such as MAP or MPE and then prune variables that are not important. A variable is regarded as unimportant if its removal does not reduce the likelihood of an explanation. In Yuan et al. (2011b) some of these methods are applied on several Bayesian networks to find an explanation that contains only the most relevant variables, showing that the methods usually fail to identify them. In addition, they depend on tunable parameters subjected to human errors.

In order to enhance the quality of the explanation, it is necessary to state what is a good explanation. An explanation is good if it has two basic properties: **precision** and **concision**. As a way to measure these properties, we need a proper metric able to not only evaluate the explanatory power of an explanation but also favor more concise explanations so that only the most relevant variables are included in an explanation. In Yuan et al. (2011b) several popular relevance measures are discussed, pointing their drawbacks to motivate the need of a new measure, called the generalized Bayes factor.

Definition 2.3.8 *The generalized Bayes factor (GBF) of an explanation $\boldsymbol{x}$ for a given evidence $\boldsymbol{e}$ is defined as*

$$GBF(\boldsymbol{x}; \boldsymbol{e}) \equiv \frac{P(\boldsymbol{e}|\boldsymbol{x})}{P(\boldsymbol{e}|\bar{\boldsymbol{x}})},$$

where $\bar{\boldsymbol{x}}$ denotes the set of all alternative hypothesis of $\boldsymbol{x}$.

We do not really compute $\bar{\mathbf{x}}$ directly in calculating GBF($\mathbf{x}$; $\mathbf{e}$). Instead we compute

$$GBF(\mathbf{x}; \mathbf{e}) = \frac{P(\mathbf{x}|\mathbf{e})(1 - P(\mathbf{x}))}{P(\mathbf{x})(1 - P(\mathbf{x}|\mathbf{e}))}.$$

Definition 2.3.9 *The conditional Bayes factor (CBF) of explanation $\boldsymbol{y}$ for given evidence $\boldsymbol{e}$ conditioned on explanation $\boldsymbol{x}$ is defined as*

$$GBF(\boldsymbol{y}; \boldsymbol{e}|\boldsymbol{x}) \equiv \frac{P(\boldsymbol{e}|\boldsymbol{y}, \boldsymbol{x})}{P(\boldsymbol{e}|\bar{\boldsymbol{y}}, \boldsymbol{x})}.$$

Yuan et al. (2011b) show that the GBF has several theoretical properties that enable to automatically identify the most relevant target variables in forming an explanation for a partial evidence. We present below some of them. The method that relies on this metric (GBF) is called MRE.

Handling extreme values

The belief that the probability of an event is equal to 1 means that one is absolutely sure that the event will occur. Therefore, any increase in probability from less than one to one or decrease from non-zero to zero is extremely significant, no matter how small the change is. A metric consistent with this interpretation such as GBF, assigns much more weight to probabilities in ranges close to 0 and 1.

Monotonicity of GBF with regard to other measures

The measures we take into consideration for the monotonicity are: the difference between the posterior and prior probability, and the belief update ratio.

It is commonly believed that the same amount of difference in probability in ranges close to zero or one is much more significant than in other ranges. Figure 2.2 shows a plot of GBF against the prior probability when the difference between the posterior and prior probabilities is fixed. The figure clearly shows that GBF assigns much higher scores to probability changes close to zero and one.

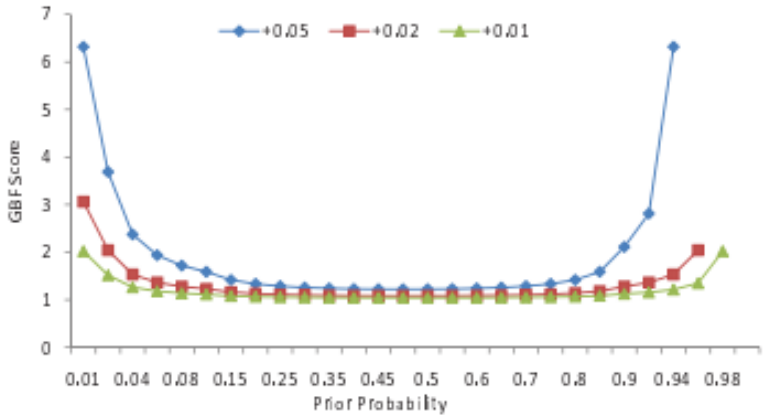


Figure 2.2: The GBF as a function of the prior probability given a fixed increase in the posterior probability from the prior. The different curves correspond to different probability increases. For example, "+0.01" means the difference between the posterior and prior probabilities is 0.01. Source: Yuan et al. (2011b).

The belief update ratio is defined as follows

Definition 2.3.10 *Assuming $P(\boldsymbol{x}) \neq 0$, the belief update ratio of $\boldsymbol{x}$ given $\boldsymbol{e}$, $r(\boldsymbol{x}; \boldsymbol{e})$, is defined as*

$$r(\boldsymbol{x}; \boldsymbol{e}) \equiv \frac{P(\boldsymbol{x}|\boldsymbol{e})}{P(\boldsymbol{x})}. \tag{2.5}$$

With regard to the belief update ratio, we have the following theorem.

Theorem 2.3.1 *(Yuan et al. (2011b)) For an explanation* $\boldsymbol{x}$ *with a fixed belief update ratio* $r(\boldsymbol{x};\boldsymbol{e}) > 1.0$*, GBF(*$\boldsymbol{x}$*,*$\boldsymbol{e}$*) is monotonically increasing as the prior probability P(*$\boldsymbol{x}$*) increases.*

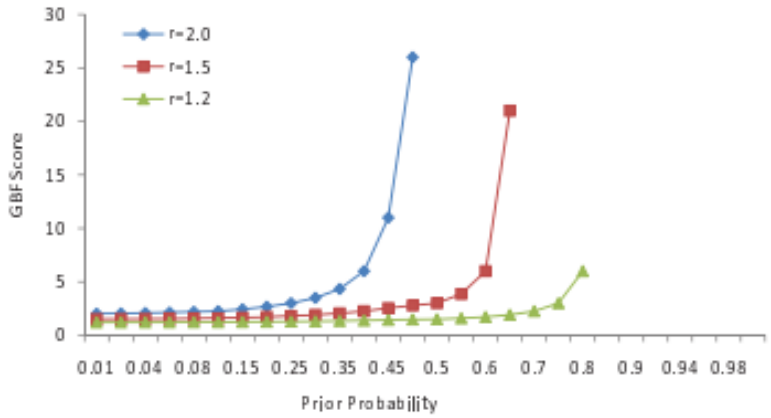


Figure 2.3: The GBF as a function of the prior probability when the belief update ratio is fixed. The different curves correspond to different belief update ratios. Source: Yuan et al. (2011b).

Figure 2.3 plots GBF as a function of the prior probability while fixing the belief update ratio. As the prior probability of an explanation increases, under the same belief update ratio of probability the GBF increases and becomes more and more significant. Therefore, explanations that cannot be distinguished by the belief update ratio can be ranked using the GBF.

Since lower-dimensional explanations typically have higher probabilities, GBF has the intrinsic capability to penalize more complex explanations.

Achieving conciseness in explanation

The key property of the GBF is that it is able to weigh the relative importance of multiple variables and only include the most relevant variables in explaining the given evidence.

Theorem 2.3.2 *(Yuan et al. (2011b)) Let* $\boldsymbol{x}$ *be an explanation with* $r(\boldsymbol{x};\boldsymbol{e}) > 1$ *and* Y *a variable that is conditionally independent from* $\boldsymbol{E}$ *given* $\boldsymbol{x}$ *, then for any state y of Y, we have*

$$GBF(\boldsymbol{x}, y; \boldsymbol{e}) < GBF(\boldsymbol{x}; \boldsymbol{e}).$$

The theorem captures the intuition that variables that are independent of the evidence variable given the explanation add no additional information to an explanation in explaining the evidence.

Theorem 2.3.3 *(Yuan et al. (2011b)) Let* $\boldsymbol{x}$ *be an explanation with r(*$\boldsymbol{x}$*;* $\boldsymbol{e}$*) > 1, and y be the state of a variable Y such that* $P(y|\boldsymbol{x},\boldsymbol{e}) < P(y|\boldsymbol{x})$*, then we have*

$$GBF(\boldsymbol{x}, y; \boldsymbol{e}) < GBF(\boldsymbol{x}; \boldsymbol{e}).$$

This is again an intuitive result; the state of a variable whose posterior probability decreases given the evidence should not be part of an explanation for the evidence.

The theoretical properties already presented show that the GBF is able to automatically identify the most relevant target variables in finding an explanation. We are now in the position to give a concrete definition of MRE in terms of this metric.

Definition 2.3.11 *Let* $\boldsymbol{M}$ *be a set of target variables, and* $\boldsymbol{e}$ *be the partial evidence on the remaining variables in a Bayesian network. MRE is the problem of finding an explanation* $\boldsymbol{x}$ *for* $\boldsymbol{e}$ *that has the maximum GBF score, GBF (*$\boldsymbol{x}$*;* $\boldsymbol{e}$*), i.e.,*

$$MRE(\boldsymbol{M}; \boldsymbol{e}) \equiv argmax_{\boldsymbol{x}, \emptyset \subset \boldsymbol{X} \subseteq \boldsymbol{M}} \quad GBF(\boldsymbol{x}; \boldsymbol{e}).$$

Although MRE is general enough to be applied to any probabilistic distribution model, MRE's properties make it especially suitable for Bayesian networks. The concise representation of a Bayesian network is also beneficial to MRE in the following ways. First, MRE can utilize the conditional independence relations modeled by a Bayesian network to find explanations more efficiently. Such independence relations can be identified through the graphical structure of a Bayesian network and may significantly improve the efficiency of the search for an explanation.

2.3.3.3 k-MREs

In many decision making problems, decision makers typically would like multiple competing options to choose from. This becomes especially useful when there are multiple top solutions that are almost equally good. In that case, we want to know not only which solution is the best, but also how much better it is than the other solutions. It allows decision makers to determine the degree of confidence in the quality of the best solution. A simple approach to finding the top k-MRE solutions is to select the explanations with the highest GBF scores. However, this strategy may find explanations that are supersets of other top explanations. It is often critical to achieve **diversity** when the goal is to find multiple solutions. To this aim, Yuan et al. (2011b) define two dominance relations among the candidate solutions of MRE. The first relation is strong dominance.

Definition 2.3.12 *An explanation* $\boldsymbol{x}$ *strongly dominates another explanation* $\boldsymbol{y}$ *if and only if* $\boldsymbol{x} \subset \boldsymbol{y}$ *and* $GBF(\boldsymbol{x}) \geq GBF(\boldsymbol{y})$.

If $\mathbf{x}$ strongly dominates $\mathbf{y}$, $\mathbf{x}$ is clearly a better explanation than $\mathbf{y}$, because it not only has a higher or equal explanatory score but is also more concise. We only need to include $\mathbf{x}$ in the top explanation set. The second relation is *weak dominance*.

Definition 2.3.13 *An explanation* $\boldsymbol{x}$ *weakly dominates another explanation* $\boldsymbol{y}$ *if and only if* $\boldsymbol{x} \supset \boldsymbol{y}$ *and* $GBF(\boldsymbol{x}) > GBF(\boldsymbol{y})$.

In this case, $\mathbf{x}$ has a larger GBF score than $\mathbf{y}$, but $\mathbf{y}$ is more concise. It is possible that we can include them both and let the decision maker to decide whether she prefers conciseness or a higher score. However, we believe that we only need to include $\mathbf{x}$, because its higher GBF score indicates that the extra variables in $\mathbf{X}$ but not in $\mathbf{Y}$ are important in explaining the given evidence.

Therefore, the k-MRE algorithm works as follows (Yuan et al. (2012)). Whenever we generate a new explanation, we check its score against the best solution pool. If it is lower than the worst score in the pool, reject the new explanation. If there are fewer than K best solutions or if the score of the new explanation is higher than the worst score in the pool, we consider adding the new explanation to the top pool.

We first check whether the new solution is strongly or weakly dominated by any of the top explanations. If so, reject the new explanation. Otherwise, we add the new explanation to the top pool. However, we then need to check whether there are existing top explanations that are dominated by the newly added explanation. If yes, these existing explanations should be excluded. Otherwise we delete the top explanation with the least score.

2.4 Comparison between Bayesian networks

Definition 2.4.1 *An acyclic graph containing both directed and undirected edges is called a partially directed acyclic graph or PDAG.*

The structural Hamming distance (SHD) between two PDAGs is the number of the following operators required to make the PDAGs match: add or delete an undirected edge, and add, remove, or reverse the orientation of an arc (see Algorithm 1). It penalizes all operations by the same amount (Tsamardinos et al. (2006)).

Algorithm 1: SHD algorithm

Input: PDAG $\mathcal{H}$, PDAG $\mathcal{G}$

```
shd = 0;
for every edge E different in H than G do
    if E is missing in H then
        shd+ = 1 ;
    if E is extra in H then
        shd+ = 1 ;
    if E is incorrectly oriented in H then
        shd+ = 1 ;
```

A higher value indicates more dissimilarity between the two PDAGs to be compared.

On theoretical grounds, PDAG measures should be preferred over DAG measures (some edges are statistically indistinguishable). However, in general the measurements performed either on DAGs or PDAGs are similar (de Jongh and Druzdzel (2009)).

In this work we are interested in comparing DAGs, and we will make a natural extension of this metric to compare explanations $\mathbf{x}$ given evidence (Section 3.3.2).

Chapter 3

Contributions and experiments

The ultimate goal of this work is to improve the interpretability of the different queries in Bayesian networks exposed in Section 2.3.3. In each case, aspects to be explained to the user, the methodology and the application on different datasets will be detailed.

3.1 Posterior probabilities

The two main exact inference methods are chosen for interpretation (variable elimination and junction tree clustering algorithm). In both cases the purpose is to describe the intermediate results in the reasoning process.

To someone unfamiliar with Bayesian networks and probabilistic inference, both algorithms may seem very complex at first sight. Faced with this situation, one can choose to explain the mathematical calculations that take place, or to graphically illustrate the sequence of steps that make up the algorithm. This work chooses for the second option.

Variable elimination is based on the systematic and ordered removal of variables until only the variables of interest remain. In this algorithm the order of elimination of the variables plays a relevant role (Bertelè and Brioschi (1972)) since it is the basis for the procedure efficiency. We wanted to justify the choice of the order of elimination of variables and show the union of factors that give rise to new factors in the elimination of each variable.

The junction tree clustering algorithm is a more advanced and technical method. We think that, despite ignoring the mathematical details, it is possible to reach an understanding about the reasoning process and the relevant variables taking action. It allows the user to get closer to an algorithm that is not specifically designed for the interpretability of the results. The elimination order that enables triangulation (Larrañaga et al. (1997)) and the subsequent creation of the junction tree has a paramount importance. The steps that go from the triangulation of the moral graph to the message passing scheme between cliques within the junction tree, and the final pruning will be shown.

For approximate inference, the main objective is to extend *pgmpy* to show the user the process of convergence towards the stationary distribution that Gibbs sampling requires (Pearl (1997)), as it increases the degree of confidence in the result really

reflecting the posterior distribution. To this end, the following tasks have been carried out:

- Study the different options to study the convergence in Gibbs sampling. Finally we decided to use statistical tests for convergence diagnostics.
- The next step was to get to know the functionalities and limitations of all the Bayesian inference libraries that offer approximate methods (*bnlearn*, *pgmpy* and *pomegranate*).

  bnlearn is probably the most mature project in the Bayesian network community. However, it may be complex to add extensions to the package (Atienza et al. (2022)). Furthermore, it only makes approximate inferences with forward sampling and likelihood weighting. On the other side, *pomegranate* does not diagnose convergence of Gibbs sampling.

  We decided to continue using *pgmpy* to perform approximate inference (Ankan and contributors (2022)) and look for another specialized library in convergence diagnostic tests to the equilibrium distribution of the Markov chain, in order to to extend *pgmpy* functions.
- Find the way to communicate R with python and the transformation of objects from one library to another. Finally we decided to use as intermediary the *rpy2* package.
- Perform the implementation.
- As a secondary goal, we have extended *pgmpy* to be able to make approximate inferences with likelihood weighting.

In addition, knowing when the convergence is reached allows increasing the computational efficiency by stopping the generation of samples.

3.1.1 Results

The objective of the different functions defined in the previous section is to enrich the numerical result of the query with additional information that accounts for the internal mechanisms of the algorithm, in order to get a better understanding of the process. For illustrative purposes, I use the well-known Asia Bayesian Network, mentioned in Lauritzen and Spiegelhalter (1988b)), which can be seen in Figure 3.1. Each

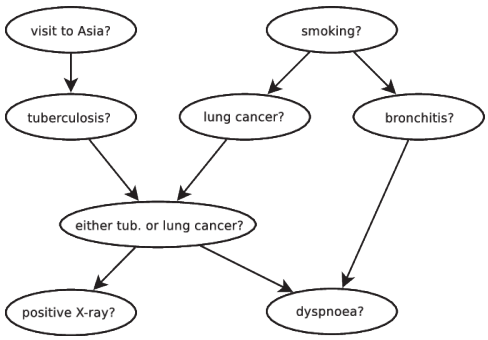


Figure 3.1: Asia BN. Source: Lauritzen and Spiegelhalter (1988b).

node represents a facet that may relate to the patient's condition. From now on we

will use the following notation: "smoke" indicates whether is a confirmed smoker, "asia" shows if he/she recently visited Asia, "dysp" indicates if the patient suffers from shortness of breath, "lung" means the presence of lung cancer, "xray" indicates abnormalities on an x-ray of the lungs, "tub" refers to tuberculosis, "bronc" is bronchitis and "either" means that the patient can suffer from either bronchitis or lung cancer.

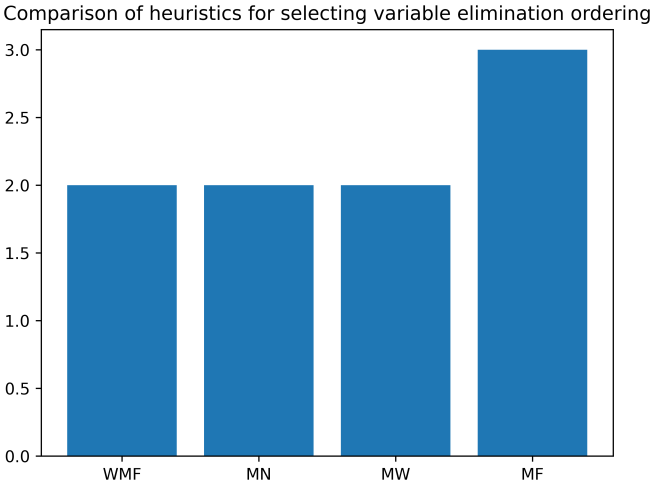


Figure 3.2: Comparison of the different variable elimination costs. On the X axis, the different heuristics, where WMF is weighted-min-fill, MN is min-neighbors, MW is min-weight and MF is min-fill (see Section 2.3.1.1). On Y axis, the induced-width.

Since the ordering of variables that the elimination process follows in Variable elimination is key to efficiency, we want to offer the user the opportunity to choose the best one with an adequate justification. We provide a comparative plot of the variable elimination cost due to the different heuristics (Figure 3.2).

In addition, for optimization processes, it is advisable to identify irrelevant variables. Some variables may have no impact on the variables of interest or the final inference result. By identifying and eliminating them from the computation graph, we can significantly reduce the computational complexity and improve efficiency. For example, if we compute the joint probability $(\mathbf{Y}, \mathbf{e})$, we can remove any leaf node if it does not belong to $\mathbf{Y} \cup \mathbf{e}$. If we recursively prune unobserved leaf nodes, we get Figure 3.3.

Regarding the junction tree clustering algorithm, the user can follow the sequence of steps of the algorithm. First of all, the construction of the Clique tree is carried out (Figure 3.5). We give special detail to the order of variable elimination ordering in the triangulation (see Figure 3.4), and to the calibration of the junction tree. This last step is the one that requires the most prior knowledge on the part of the user, given the nomenclature used for clique beliefs $\beta(C)$, sepset beliefs $\mu(C_i, C_j)$, and the auxiliar terms $\sigma(C_i, C_j)$ defined ad hoc for this algorithm. One of the properties that must have an explanation, according Lacave and Díez (2002), is adaptation. This implies establishing the degree of detail according to the knowledge of the user and his expectations. The degree of interpretability only affects the calibration step; a high level give more details about the message passing (Figure 3.6); a low level can

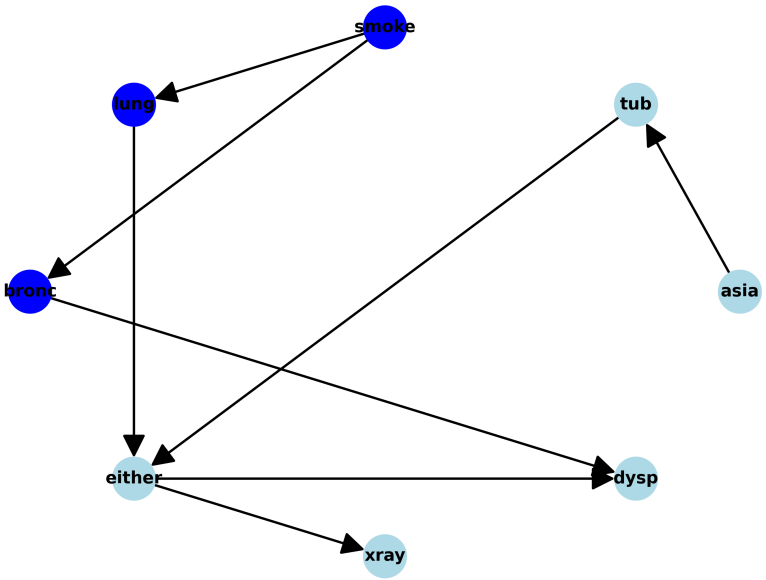


Figure 3.3: Reduced model for variable elimination (optimization). In dark blue, the remaining nodes after pruning. In light blue, the pruned node. Query variable: bronc. Evidence: Lung = 'Yes'.

be seen in Figure 3.7.

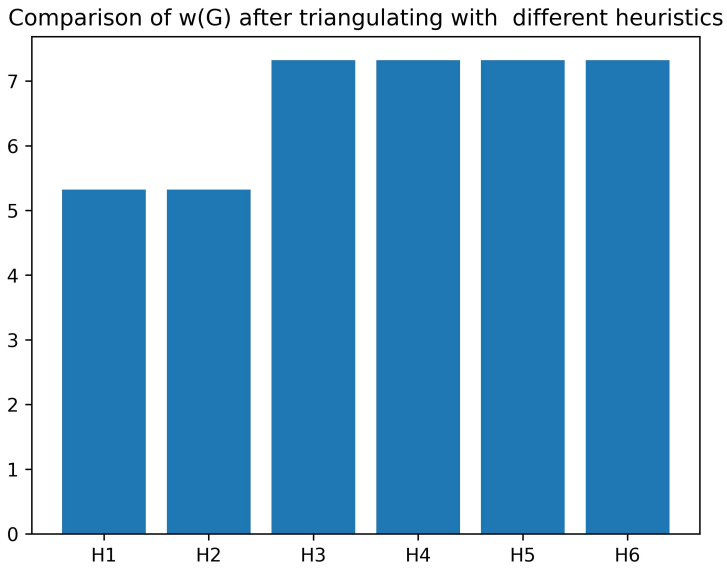


Figure 3.4: Comparison of the different variable elimination costs for triangulation. On the X axis, the different heuristics in Section 2.3.1.2. On Y axis, the metric defined in equation (2.4).

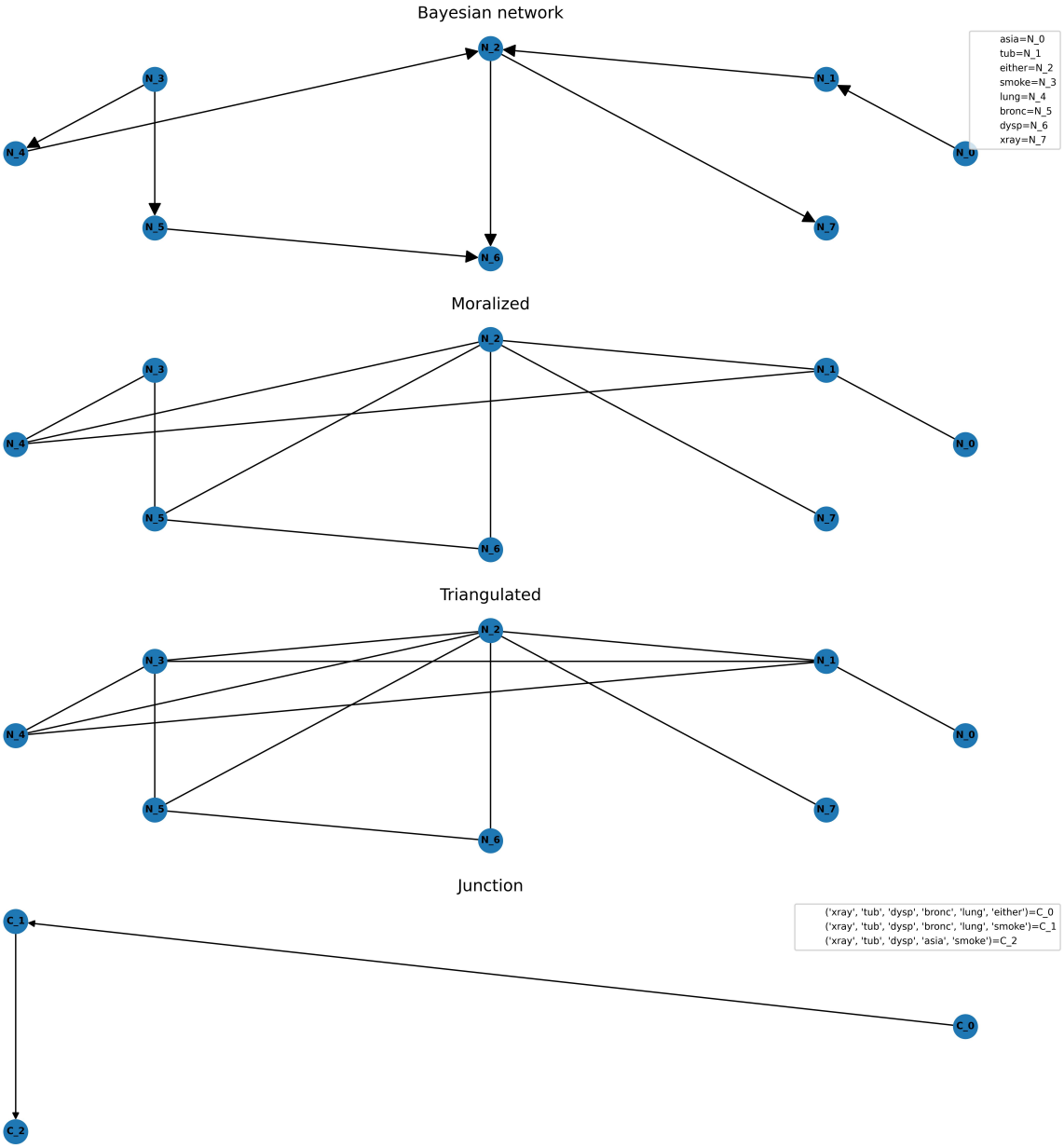


Figure 3.5: Sequence of steps in the construction of the junction tree (node names in the legend).

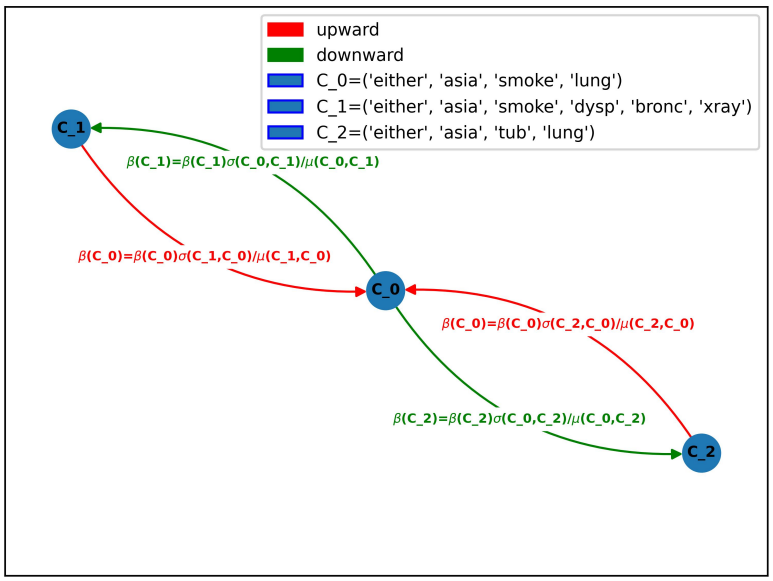


Figure 3.6: Calibration step in the junction tree clustering algorithm. Degree of interpretability: high.

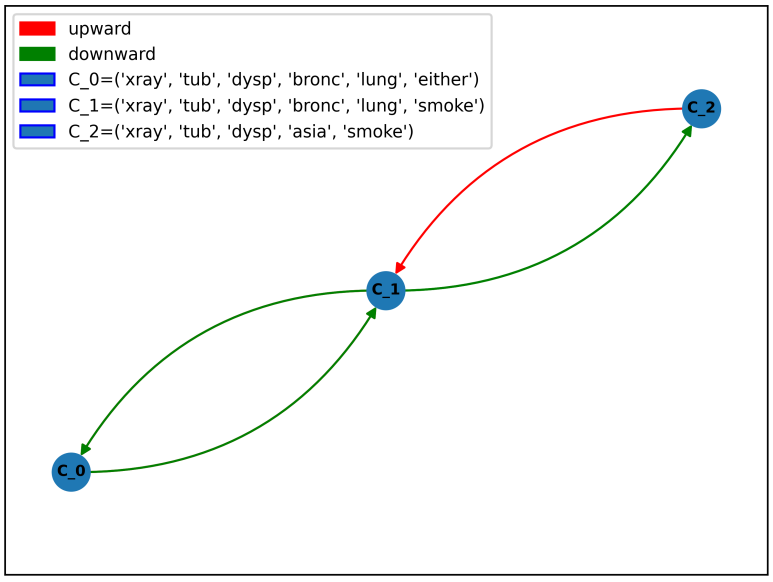


Figure 3.7: Calibration step in the junction tree clustering algorithm. Degree of interpretability: low.

With regard to the approximate methods, due to the lack of flexibility that rpy2 allows when combining python objects with R objects, we have fixed to 3 the number of Markov chains when Gibbs sampling is requested. In the following example, we make the query $P(bronc)$, using Gibbs sampling, and set the size of the chain to 15000 samples. For example, in Table 3.1 we can confirm that after 15000 iterations, the shrink factor of variable *Smoke* takes the value 1.03, which is a sign of convergence. A potential problem with Gelman's test is that it may misdiagnose convergence if the shrink factor happens to be close to 1 by chance (Plummer et al. (2006)). By calculating the shrink factor at several points in time and providing a plot of the evolution, it shows if the shrink factor has really converged, or whether it is still fluctuating. Figures like the one in Figure 3.8, allows to confirm the convergence of the variable *Smoke* shrink factor: the between chain variance relative to the within-chain variance are about the same and the chains have all converged to the true distribution. It is the case of a univariate convergence diagnosis.

	Point est
asia	1.00
tub	1.32
either	1.85
dysp	1.04
smoke	1.03
bronc	1.00
lung	1.73
xray	1.55

Table 3.1: Gelman and Rubin shrink factors.

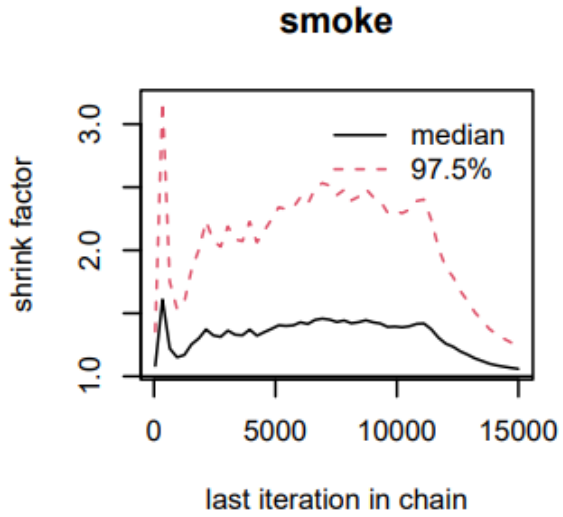


Figure 3.8: Evolution of the shrink factor in time. Variable: *Smoke*.

```
1 Multivariate psrf
2
3 1.44
```

We also obtain a multivariate value of the Gelman Rubin shrink factor equal to 1.44.

It gives us an approximation of the global convergence of all the variables. Although the convergence of some variables is not cause for concern, it is not the case of all the variables, and the multivariate factor warns that we have not reached the true posterior distribution (Multivariate psrf > 1.1).

If the user is familiar with z-score tests, can request another type of convergence diagnosis such as Geweke's, which is also univariate, and allows observing the evolution of the statistic when successively discarding a greater number of initial iterations of the chain (Plummer et al. (2015)).

3.2 Abduction queries

3.2.1 Datasets

The scope in this section is multivariate Gaussian data. The main difficulty is that it is practically impossible to find real (not synthetic) datasets where the random variables follow a multivariate Gaussian distribution. Our approach consists, therefore, in assuming Gaussianity in the variables and modeling them as if they were. This causes the generated results to be less accurate than desired. We have chosen two datasets to illustrate the methodology that is going to be developed.

3.2.1.1 Electric motor temperatures

These are measurements from various sensors within an electric motor over time. The dataset, extracted from Kaggle (available at `https://www.kaggle.com/datasets/wkirgsn/electric-motor-temperature`), is quite large, with tens of thousands of rows and 10 variables. It is clean and free of missing values.

All recordings are sampled at 2 Hz. The data set consists of multiple measurement sessions in which the engine is stressed to bring it to high temperatures. They can be distinguished from each other by column *profile id*. A measurement session can be between one and six hours long. For our purpose we assume that the data rows do not have a temporal relationship.

The motor is excited by hand-designed driving cycles denoting a reference motor speed and a reference torque. Currents in d/q-coordinates (columns i_d and i_q) and voltages in d/q-coordinates (columns u_d and u_q) are a result of a standard control strategy trying to follow the reference speed and torque. Columns *motor speed* and *torque* are the resulting quantities achieved by that strategy, derived from currents and voltages.

The most interesting target features are rotor temperature (*pm*), stator temperatures (*stator**) and *torque*. Being able to have strong estimators for the rotor temperature helps the automotive industry to manufacture motors with less material and enables control strategies to utilize the motor to its maximum capability. A precise torque estimate leads to more accurate and adequate control of the motor, reducing power losses and eventually heat build-up.

The experiments have been carried out focusing on a measurement session, that is, on a specific value of the *profile id* column. Also, there is a high correlation between current and voltage with motor speed, and the 99% of motor speed variability can be linearly explained by current and voltage (R squared coefficient). Therefore current and voltage variables have been removed.

To better understand the meaning of the variables, it is necessary to put some context in the operation of a PM motor (Jaszczolt (2017)).

A PM motor is an alternating current (ac) motor that uses magnets imbedded into or attached to the surface of the motor's rotor. A rotating field is generated by the stator winding and induces a current in the rotor bars. The interaction between the field and the current produces the driving force. The variable motor speed refers to the angular rate at which the motor's rotor rotates in response to the application of electrical power. As the motor speed increases in a PM motor, there are increases in

resistance losses (generated due to current flowing through the windings and electrical resistance of the conductors), mechanical losses, and difficulty in dissipate heat, which can result in an increase in rotor temperature. In a PM motor, the relationship between the rotor temperature (pm) and the motor speed can be influenced by several factors: motor design, load conditions, cooling mechanisms, and thermal characteristics (heat dissipation for example).

As we already mentioned, it is rare to find real data whose joint distribution is multivariate Gaussian. This is confirmed by performing a multivariate Gaussianity hypothesis test on our data set. In particular, the Henze-Zirkler test (Henze and Zirkler (1990)) rejects the null hypothesis of Gaussianity. A univariate exploration of the variables using histograms also confirms the lack of Gaussianity in the marginal densities.

In the domain of interest we are dealing with, it is vital to accurately monitor the stator temperature to prevent a temperature rise and to ensure that motor performance does not deteriorate. Maintaining cost-effective motors is crucial. It is therefore possible that the engineer, faced with the need to make decisions, may be interested in taking his system to a behavior state in which he can know the most probable state with greater reliability. Hence, the motivation of this experiment, is not to predict future values, but to model a Bayesian network with a higher MPE probability than the minimum value requested by the user from the original system. If the user has less uncertainty about the most probable situation given evidence, he/she can, for example, know the explanation for an unexpected behavior of the motor and plan with greater security the repair of the manufacturing plants, achieving a more efficient maintenance.

3.2.1.2 COVID-19

The COVID-19 data (Constantinou et al. (2020)) captures the development of the pandemic in the UK over a period of 2.5 years, with a focus on viral tests, infections, hospitalisations, deaths, vaccinations, policy interventions, and population mobility. Data were collected from different public sources, with the majority of information coming from official UK government websites for COVID-19.

Our goal is not predictive. Again, the goal is to understand the variables relationships in a hypothetical reality in which the health authorities would have predicted the most likely situation with greater reliability. For our purpose we assume that the data rows do not have a temporal relationship.

The dataset is free of missing values. As our scope is continuous data, we removed categorical variables: Majority COVID-19 variant, Lockdown, Face masks, Transportation activity..., losing important information about UK policy interventions regarding lockdowns and mask mandates and indices about mobility (flights, buses, trains and transit stations, visits to parks, school activities, ...).

We describe and categorise the data variables by the type of information they capture (Constantinou et al. (2023)):

- **Viral tests**: A single variable fits this category; the *Tests across all four pillars*. Testing conducted under Pillar 1 represent tests carried out by the National Health Service and the UK Health Security Agency, under Pillar 2 by the

UK government COVID-19 testing programme, under Pillar 3 involves antibody serology testing, and under Pillar 4 involve testing for national surveillance.

- **Infections**: Three variables make up this category; namely *Positive tests*, *New infections*, and *Reinfections*. These variables capture information about daily cases, and whether those cases were new infections or reinfections.

- **Hospitalisations**: This category consists of the variables *Hospital admissions*, *Patients in hospital*, and *Patients in MVBs*. These three variables capture information about the number of patients admitted to hospital with COVID-19, the number of patients in hospitals with COVID 19, and the number of patients in mechanical ventilator beds (MVBs) with COVID-19 respectively.

- **Vaccinations**: A single variable, called *Second dose uptake*, that captures information about the 2nd dose vaccine uptake, and which represents the proportion of the eligible population who received the vaccine.

- **Deaths**: two variables, called *Deaths with COVID on certificate* and *Excess mortality*, that capture the number of deaths with COVID-19 listed on the death certificate, and overall excess mortality.

3.2.2 Methodology

As a first approximation to our proposal in the area of abductive inference, we begin with a brief summary of the steps that make up our work:

1. Identify existing proposals in the literature regarding abductive inference.

2. Identify knowledge gaps and interesting questions that remain to be answered and that can add value to the user's understanding ('Knowledge gap detected and proposal introduction' in Section 2.3.3.1) .

   The general goal is to find a Gaussian Bayesian network that describes a hypothetical reality in which the most probable situation has a higher probability (than the lower threshold set by user).

   Although at first, this question may sound confusing, we are going to verify its usefulness by applying it to two datasets from very different areas.

3. Define a methodology based on the use of EDAs and the analysis of the differences (qualitative and quantitative) between the original BN and the BN after running the EDA to draw conclusions (Figure 3.9).

As we have already mentioned, the methodology relies heavily on the use of EDAs. Next, we are going to detail the cost function and the stopping criteria designed to achieve the stated objective.

Suppose the vector $\mathbf{X}$ is partitioned into two components $\mathbf{X} = (\mathbf{Y}, \mathbf{E})$, where $\mathbf{Y}$ are the variables of interest to the user and $\mathbf{E}$ are the evidence variables. In our approach we consider a flexible idea of evidence (which we will refer to as 'soft evidence') in the following way:

The goal is to minimize the cost function C_l, $l = 1, 2, 3 \ldots$

$$C_l(\mathbf{x}) = \begin{cases} log(f_l(\boldsymbol{\mu}_l)) - log(f_l(\mathbf{x})) & \|\tilde{\mathbf{e}} - \mathbf{e}\|_\infty \leq \epsilon \\ 99999 & \text{other case} \end{cases} \tag{3.1}$$

where l is the iteration number, $\mathbf{x} = (\mathbf{y}, \tilde{\mathbf{e}})$, and

$$f_l(\mathbf{x}) = (2\pi)^{-n/2} |\boldsymbol{\Sigma}_l|^{-1/2} exp - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_l)' \boldsymbol{\Sigma}_l^{-1} (\mathbf{x} - \boldsymbol{\mu}_l).$$

The cost function is a piecewise function that assigns:

- A very high cost to those individuals whose evidence variables are not within the range $[e_i - margin_i, e_i + margin_i]$ (where $margin_i$ is the maximum disagreement between variable X_i and the evidence e_i).
- The difference between the loglikelihood of the mode, and the individual to be evaluated.

The selected stopping criterion is

$$\text{death iter} \geq N_{iter} \quad \text{or} \quad \frac{f_l(\mu_l)}{f_0(\mu_0)} \geq \alpha,$$

where α is the probability density growth factor and N_{iter} is the maximum number of iterations without a cost function improvement (both decided by the user), μ_0 is the mean of the original dataset, and μ_l is the mean of the population in each iteration l. The second stopping criterion accounts for the increase in the mode density function in iteration l with respect to the initial value

For implementing the EDAs, we use the python package *EDAspy*, with slight modifications to:

- Allow the definition of cost functions that dynamically vary during the iterative process,
- Allow a softer concept of evidence: evidence variables may differ to some extent from the fixed value, and
- Customize the stopping criterion.

These code modifications will be clarified in the *Implementation* section (Appendix B).

To carry out the experiments, some decisions were to be made: size of the population, maximum number of iterations during runtime, number of iterations with no improvement after which the algorithm finishes, percentage of population selected to update the probabilistic model, percentage of previous population selected to add to the new generation, constraints over the network structure, growth probability factor α ... Most of them are hand tuned by trial and error to get the best (in the sense of little cost and big probability dense function increase) and fastest results, and $\alpha = 1.5$ (just for illustrative purpose).

Once we have the results, the original and optimized Bayesian networks will be analyzed: the qualitative relationships that have appeared and disappeared in the network will be compared, as well as the quantitative variations in the mean and the covariance matrix.

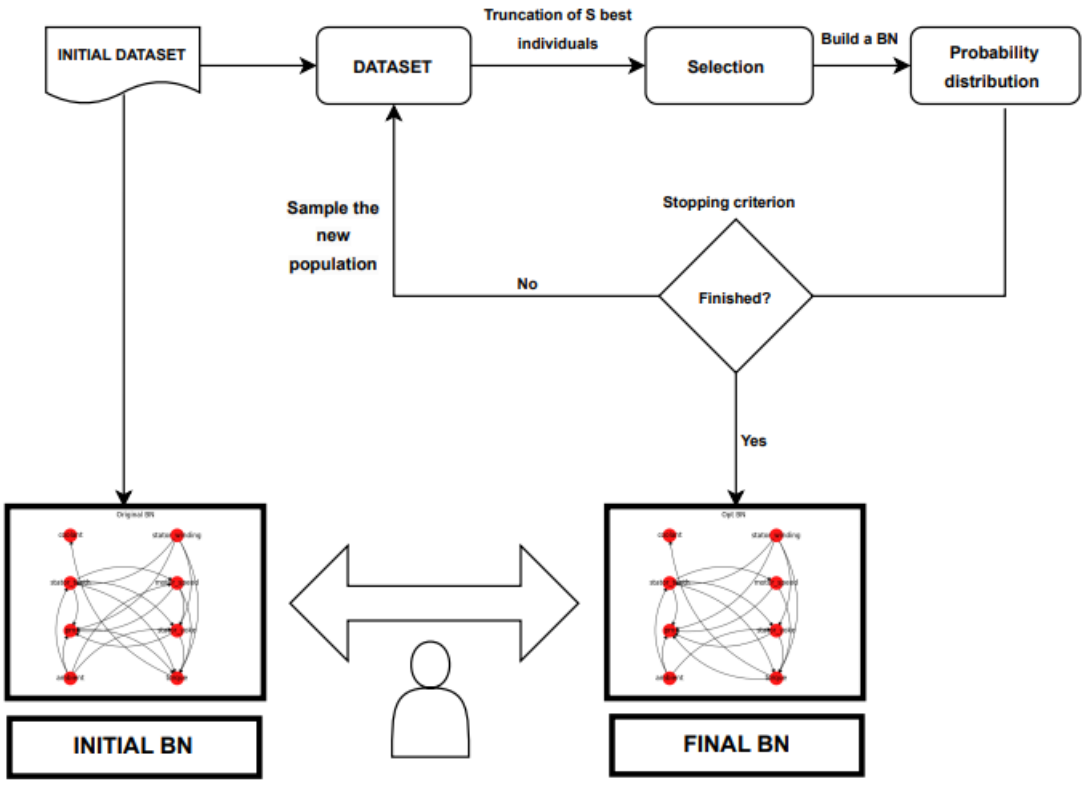


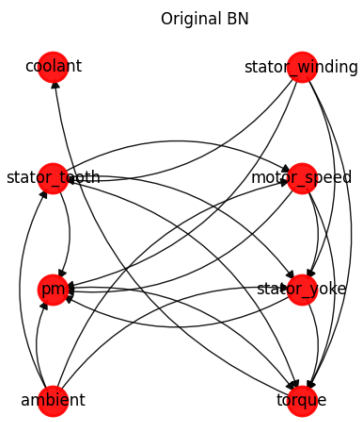
Figure 3.9: Methodology for answering our proposal in MPE.

3.2.3 Results

We want to compare both the qualitative part of the Bayesian network (DAG) and the density function. As in this case it is a multivariate Gaussian, we provide a comparison of the covariance matrix of the original Bayesian network and after the optimization process.

3.2.3.1 Electric motor temperature

Original BN



Optimized BN

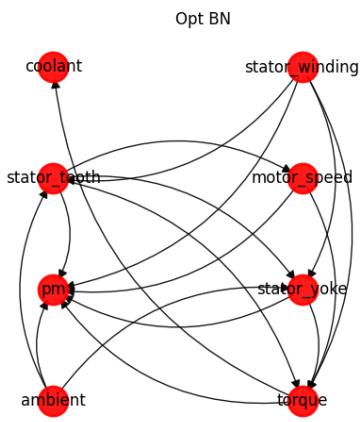


Table 3.2: Evidence: torque=25 and coolant=18.9. We have set the margins of disagreement with the evidence at 0.5 and 0.1 respectively.

For illustrative purposes, we want to increase the value of the density function in the MPE by a factor greater than 1.5. After the optimization process we successfully obtain a probability density function increase of 1.638. The next step is to interpret the qualitative and quantitative changes (Table B.3 in Appendix) in the Bayesian networks (Figure 3.2).

In the first place we have as distinct edges (ambient, motor speed) and (motor speed, stator yoke). Now, regarding the initial situation, motor speed

- is conditionally independent of ambient and stator york,
- decreases its correlation by more than $\frac{1}{4}$ with stator tooth, stator winding and stator yoke,
- decreases its correlation with *pm*.

Therefore, the new reality is a situation in which the variables of greatest interest (rotor temperature -pm- and stator temperature -stator_*-) have less dependence on motor speed.

Taking into account the engine operation described in previous Section 3.2.1.1, a decrease in the dependence of the rotor temperature on the motor speed, ultimately translates into an adequate design of the cooling system and following the manufacturer's instructions, to isolate as much as possible one effect from another.

If we isolate (to a certain extent) some effects from others (in this case motor speed and stator/rotor temperature), it is easier to predict the evolution or causes of engine

damage. Therefore, the modeled hypothetical reality responds to the objective of increasing confidence about what is actually happening.

3.2.3.2 COVID-19

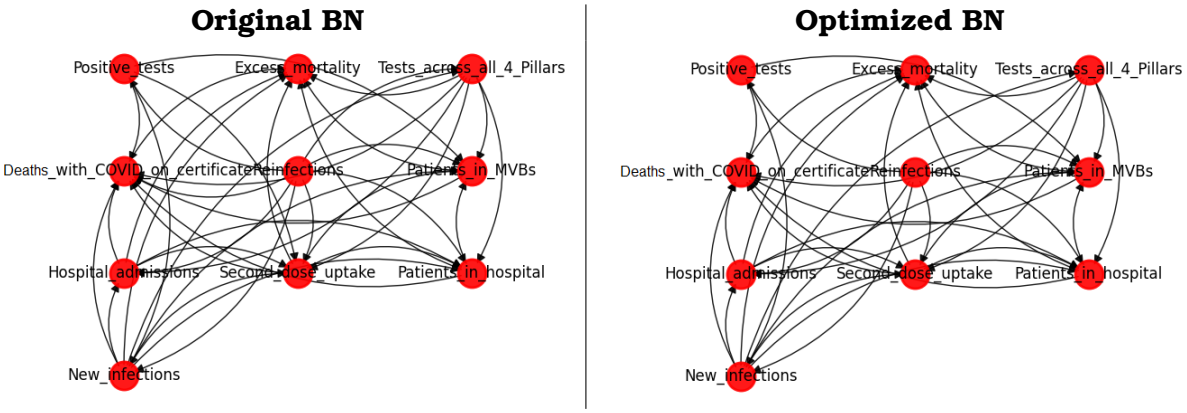


Table 3.3: Evidence: Second dose uptake=45, Excess mortality=9. The allowed disagreement in the evidence is set to 5.

We successfully obtain a probability density function increase of 1.592 (over the 1.5 requested). In a similar way to what was done with the previous dataset, we analyzed the results by comparing the Bayesian networks (Figure 3.3). We have as different edges: (*Reinfections*, *Deaths with COVID on certificate*), (*Reinfections*, *Tests across all 4 Pillars*), (*Positive tests*, *Second dose uptake*). In this case the quantitative changes in the parameters (mean and covariance matrix) are more remarkable and we highlight them in Table 3.4.

	Positive tests	Excess mort	Tests	Death with COVID	Reinfections	Patient in MVBs	Hospital admission	Second dose	Patient in hosp	New infections
Positive tests	1,28E+00	2,49E+00	1,43E+00	1,24E+00	1,35E+00	1,15E+00	1,17E+00	1,03E+00	1,26E+00	1.28E+00
Excess mort	2,49E+00	9,33E-02	1,14E-01	2,48E+00	1,66E-02	-9,64E-01	2,39E-01	2,22E-02	2,44E-01	-2.19E+01
Tests	1,43E+00	1,14E-01	9,69E-02	1,76E-02	-1,98E-02	2,02E-02	2,31E-02	-1,26E-02	1,91E-02	1.70E+00
Death with COVID	1,24E+00	2,48E+00	1,76E-02	1,72E-01	1,34E-03	2,65E-01	7,48E-02	7,08E-03	7,64E-02	1.55E+01
Reinfections	1,35E+00	1,66E-02	-1,98E-02	1,34E-03	8,25E-03	6,79E-04	-6,55E-03	4,17E-03	-4,90E-03	1.35E+00
Patient in MVBs	1,15E+00	-9,64E-01	2,02E-02	2,65E-01	6,79E-04	2,68E-01	5,41E-02	5,33E-03	6,04E-02	1.12E+01
Hospital adm	1,17E+00	2,39E-01	2,31E-02	7,48E-02	-6,55E-03	5,41E-02	9,52E-02	5,05E-03	8,90E-02	1.66E+00
Second dose	1,03E+00	2,22E-02	-1,26E-02	7,08E-03	4,17E-03	5,33E-03	5,05E-03	4,13E-03	5,53E-03	9.89E-01
Patient in hosp	1,26E+00	2,44E-01	1,91E-02	7,64E-02	-4,90E-03	6,04E-02	8,90E-02	5,53E-03	1,12E-01	1.78E+00
New infections	1.28E+00	-2.29E+01	1.70E+00	1.55E+01	1.35E+00	1.12E+01	1.66E+00	9.89E-01	1.78E+00	1.54E+00

Table 3.4: Ratio $covariance_{ij}^{original}/covariance_{ij}^{opt}$ between the covariances of the original BN and the optimized BN (data available in Appendix, Section 3.4). In yellow, remarked decreases in this metric.

The direct dependencies *Reinfections-Death with COVID*, *Reinfection-Test cross* and *Positive test-Second dose* in the optimized network disappear. Furthermore, if we compare the mean vectors (data in the Appendix, Section 3.4) the number of reinfections represents a larger factor with respect to the total new infections. With these statements and looking at the comparative Table 3.4 of covariance matrices, we conclude that in the new hypothetical reality, we have a higher proportion of reinfections and these are less correlated with hospital admission, and the second dose is inoculated less. The new infections are not directly related to death with COVID-19. The optimized Bayesian network models a reality in which there is greater herd immunity, since there are more people who have already suffered from the disease in the past, have developed antibodies and may not need to get a second vaccine. Therefore, it is a more controlled and predictable situation, with less drastic changes, and

it makes sense that the MPE is more likely. Specifically the increase in probability is the change in volume below the surface of the multivariate probability density around the mean. A higher value of the density function, a higher probability.

In Figure 3.10, we can see the change in density function for each variable. In the optimized model, we have less uncertainty in the values (although not always, as happens in the first and last graph). In Figure 3.11 we can visualize notable changes in the relationships between the two variables (*Reinfections* and *Hospital admissions*): not only we have a higher density function value in the mode, but also a less flattened curve.

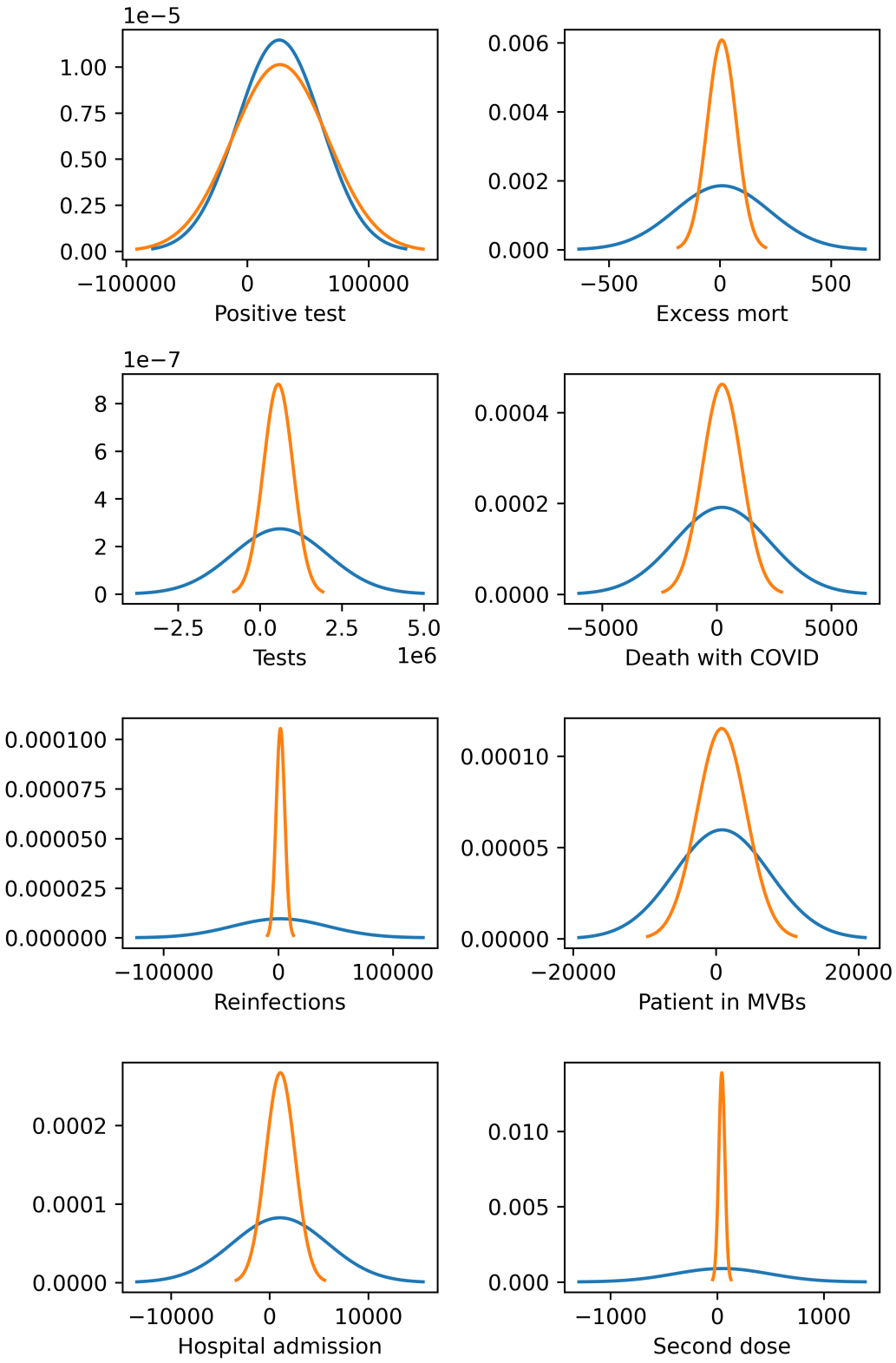


Figure 3.10: Comparison, for each variable, of the density functions in the original model (blue) and the optimized model (orange).

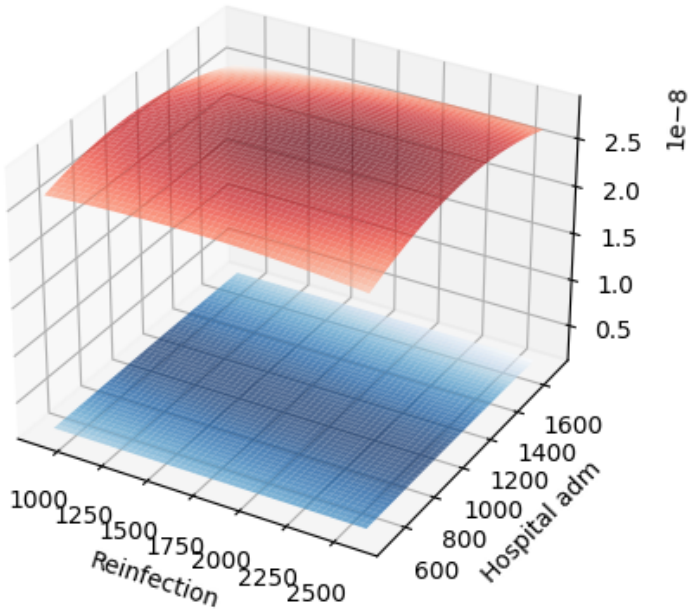


Figure 3.11: Two-dimensional probability density function change for *Reinfections* and *Hospital admissions*. In blue, the original data. In red, the data from the optimized generation.

3.3 Most relevant explanation

We have identified a gap of available software for the task of finding the MRE and k-MRE. The objective of this section is to design the methodoly to perform the tasks with discrete datasets and enrich the comparison in the k-MRE list.

3.3.1 Methodology

Since MRE is NP^{PP} (Yuan et al. (2011a)), approximation methods may be the only feasible approach to solving MRE in large Bayesian networks. In this work we use an adaptation of a local search method, the forward search, to find the MRE.

In forward search, we first generate one or more starting solutions. Then for each starting solution, we greedily improve the solution by either adding an additional variable with some state or changing the state of one variable in the current solution. In the meanwhile, MRE needs to select the states for these variables in maximizing the GBF score as an explanation.

Algorithm 2: Forward search algorithm for MRE

Input: Bayesian network $\mathcal{B}$ with a set of target variables $\boldsymbol{X}$, and a set of evidence variables $\boldsymbol{E}$.
Output: An MRE solution over **X**.

```
Initialize the starting solution set I with an initialization rule
Initialize the current best solution y_best = ∅
for s in I do
    y = s
    while y does not stop updating do
        Find the neighboring solution set N of y by either adding one target
         variable with any state or changing one variable to another state
        Compute GBF score for each solution in N
        Update y if the best solution in N has higher GBF score
    if GBF(y) > GBF(y_best) then
        y_best = y
return y_best
```

The solution space of MRE has a lattice structure similar to the graph in Figure 3.12 for three binary variables A, B and C. The two possible values for each variable appear with the same letter as the name of the variable, in lowercase and uppercase. Each node in the graph contains a potential solution, and the edges show the connectivity between the solutions. Two solutions are connected if they differ only by one variable: either the state of the variable is different, or one solution has an extra variable than the other. The solutions range from singletons to full configurations of the target variables. In forward search, we are searching the graph in Figure 3.12 top down while allowing horizontal moves. A forward search algorithm is outlined in Algorithm 2.

Our method for obtaining the MRE, relies on GBF in finding explanations for a given evidence. This measure is not directly applicable to continuous data, because continuous variables have an infinite number of possible values. Therefore the GBF is

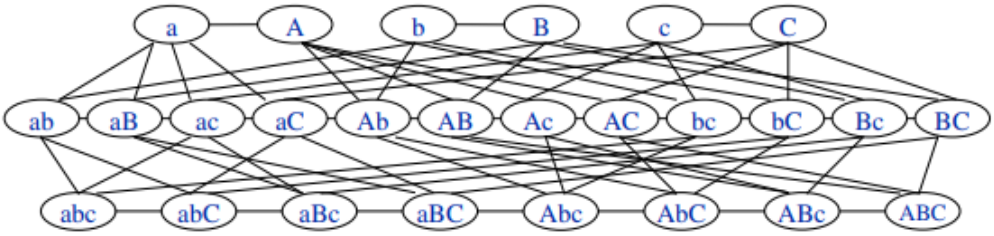


Figure 3.12: Solution space of MRE for three binary variables. Source: Yuan et al. (2011a).

specifically designed for discrete data.

The datasets we have considered until this moment, are continuous, and the algorithm we have implemented is discrete-data dependent. Therefore the dataset's usage requires a previous discretization. This can have several disadvantages, including loss of information, arbitrariness of binning... leading to a sensitivity to binning methods and a new source of variability and subjectivity (Nojavan A. et al. (2017)). Therefore, the suitability of discretization depends on the nature of the data, the problem at hand and the requirements of our analysis. To prevent this obstacle from staining the analysis of our implementation, we use a well-known benchmark dataset for evaluating Bayesian networks: the *ALARM* dataset (Beinlich et al. (1989)). It consists of categorical variables.

The k-MRE algorithm (Section 2.3.3.3), which is summarized in Algorithm 3, takes into account two dominance relationships in order to achieve diversity among the explanations. These relationships make sure that no explanation is contained in another one in the list. In addition, we can measure the amount of diversity by comparing with some metric all the explanations with the most relevant one (the first). To this purpose we propose to use a Hamming distance between explanations, that is, the number of variable states in which the two explanations differ. This metric could be useful if two explanations are close in GBF, but one is much more diverse with respect to the MRE than the other. In this case, the user may prefer the most diverse one.

3.3.2 Results

ALARM is a diagnostic application that implements an alarm message system for patient monitoring in a hospital intensive care unit (ICU). The goal is to provide specific text messages advising the user (the doctor) of possible problems.

For illustrative purposes, we run the k-MRE algorithm with $k = 3$ and evidence of low lung ventilation on top of the ALARM dataset. The objective is to facilitate the comparison between the k most relevant explanations. For that purpose we use a parallel coordinate plot (Figure 3.13 and Table 3.5) to visualize the differences between the different explanations. The continuous color scale for the lines represents the GBF value.

These explanations, with GBF belonging to the range $10 - 30$, can be considered as strong explanations (Yuan et al. (2011b)). In addition they are consistent with the

Algorithm 3: k-MRE algorithm

Input: A list of different Most Relevant Explanations.
Output: k-MRE list (the pool).

```
Initialize an empty list for the k-MRE (pool)
for x in MRE list do
    if GBF(x) < min(GBF_pool) then
        Continue
    if GBF(x) > min(GBF_pool) then
        if x is strongly/weakly dominated by other instances in the pool then
            Continue
        else
            Add x to the pool
            if Other top explanations are dominated by x then
                Remove those explanations from the pool
```

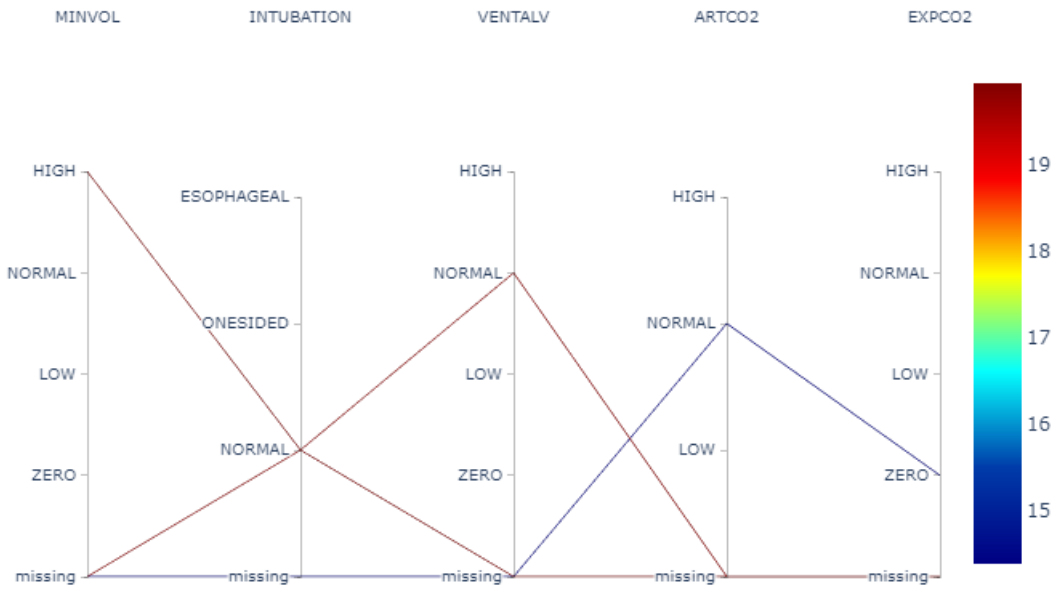


Figure 3.13: 3-MRE with ALARM dataset. Evidence: VENTLUNG (lung ventilation) = LOW. Variables: MINVOL (minimum volume), ARTCO2 (arterial CO2), VENTALV (pulmonary alveoli ventilation), EXPCO2 (expelled CO2).

corresponding MAP query:

```
inference.map_query(variables=['MINVOL', 'INTUBATION', 'VENTALV', 'ARTCO2', 'EXPCO2'],evidence
    ={'VENTLUNG': 'LOW'})
 {'MINVOL': 'HIGH', 'INTUBATION': 'NORMAL', 'VENTALV': 'NORMAL', 'ARTCO2': 'NORMAL', 'EXPCO2':
```

MINVOL	INTUBATION	VENTALV	ARTCO2	EXPCO2	GBF	Hamming distance
HIGH	NORMAL				19.93	0 (MRE)
	NORMAL	NORMAL			19.79	2
			NORMAL	ZERO	14.39	4

Table 3.5: Tabular results of 3-MRE in ALARM dataset. The last two columns allow a bivariate comparison taking into account both GBF and diversity.

```
        'ZERO'}
```

Let's see what information it provides in medical terms. The most relevant explanation is high minimum volume and normal intubation. Minimum volume refers to the minimum volume of air that must be exchanged in the lungs for adequate breathing. Essentially, the result indicates that despite the normal intubation, a high minimum volume of ventilation may be a contributing factor to the low lung ventilation observed. We may think that the data is from a patient whose air volume needs are high and current ventilation does not meet his/her needs.

From the second most relevant explanation we can see that MINVOL and VENTALV are almost equally relevant variables, with similar explanatory power. In the case of a normal VENTALV, it can be inferred from this explanation that the patient is resting or under sedative medications that decrease his/her pulmonary ventilation.

With regard to the diversity of explanations, we can observe that in the 3-MRE list, no explanations are contained in one another, which has been achieved by taking into account the relationships of strong dominance and weak dominance. In addition, we can quickly measure the diversity incorporated by the second and third explanations regarding MRE with the 'Hamming distance' column. Although the third has a lower GBF value, it incorporates more diversity than the second.

Chapter 4

Conclusions and future work

4.1 Conclusions

We have increased the interpretability of exact inference algorithms by visualizing the different steps. This can help support the decision-making by

1. Sheding light on how calculations are made at each stage.
2. Detecting unexpected results that could indicate problems in the inference process.
3. Validate the results.
4. Enhancing the communication: visual representations can be more accessible than purely mathematical descriptions. In addition, we can also detect which variables have an impact on the result and which ones are irrelevant.

With regard to approximate inference, we have added to the *pgmpy* library the possibility of diagnosing convergence when making inferences with Gibbs Sampling. The diagnosis of convergence is important for several reasons: it helps determine if the Gibbs Sampling algorithm has reached a stable sampling distribution. If the algorithm has not converged, the results obtained may not adequately reflect the desired posterior distribution, hence decreasing the precision of the results, and indicates that the results are likely to vary in different executions of the algorithm; additionally, visualizing the evolution of the convergence factor with the iterations allows us to approximate the number of additional samples needed to achieve convergence.

We have addressed a new question related to the most probable explanation, which has not been answered in the existing literature until now. It is about finding a hypothetical reality modelled by a Bayesian network, in which the most probable explanation has a higher probability. This reality can serve as a reference to guide our actions towards a situation in which we can act under less uncertainty. In the case of motor dataset, the new knowledge allows a successful maintenance and repair of the manufacturing plants; in the case of COVID-19 dataset, it is not always possible to take the current reality to a more favorable state, but some variables are more controllable than others and by knowing what their configuration is in the new hypothetical reality, the health authorities can try to redirect the situation. In both cases, the initial situation must be compared with the final model, looking for the

direct relationships that have appeared or disappeared, and the quantitative changes in the vector of means and the covariance matrix. This enables to advise experts in the field of research who want to answer the question: *what should I change to be able to predict the explanation of evidence with greater certainty?*

The limitation of our approach consists in assuming Gaussianity in the variables and modeling them as if they were. This causes the generated results to be less accurate than desired.

We have implemented an approximate algorithm, based on local search (forward search), to find MRE. Its implementation is supported by the *pgmpy* library. This work would not only add extra functionality to *pgmpy*, but is one of the first open access implementations known. Additionally, we have made a proposal for the presentation of results that can help to compare the different explanations, both visually using parallel coordinates and through a new metric, a Hamming distance between explanations.

All the proposed explanations meet the following requirements in Definition 2.2.1: they are understandable, improve the knowledge about the object of the explanation. We think they also answer interesting questions that can satisfy the wishes of the person requesting them.

Finally, given the classification of an explanation according to its properties (Chapter 2), we do the same with the proposals of this work (Figure 4.1).

FIELD OF EXPLANATION	PROPERTIES					TYPE OF DATASET
	CONTENT			COMMUNICATION	ADAPTATION	
	FOCUS	PURPOSE	LEVEL			
INFERENCE ALGORITHMS	Reasoning process	Comprehension	Macro	Visual	Yes (in the case of junction tree)	Discrete
ABDUCTIVE INFERENCE	Evidence	Comprehension	Macro	Visual (BN graphs) and textual (tables, text)	No	Gaussian
MRE	Evidence	Comprehension	Macro	Visual (parallel coordinates) and textual (table)	No	Discrete

Figure 4.1: Classification of explanations developed in this master thesis.

All the implementations are available in `https://github.com/martatubia/TFM`

4.2 Future work

During the development of this master thesis, questions, difficulties and new work lines arose. Although they were out of the scope of our work, either due to the limitation in time or because they went beyond the pre-established goals, they motivate the following proposals:

- As far as we know, the open/free access libraries used to make approximate inferences do not have integrated the possibility of diagnosing the convergence of a chain of Gibbs samples. We have circumvented the difficulty by combining a python library with an R library. Executing R code through *rpy2* is slower compared to directly executing R code in a native R environment. Possible causes are: translation and communication between Python and R, the additional function call overhead due to invoking R functions from Python... The execution time is an area to be improved.

- The interpretability proposal for the MPE has been implemented on top of *EDAspy*, which has functionality for Gaussian variables, binary variables (UMDAd) and independent categorical variables (categorical EDA). It would be interesting to implement our proposal for categorical variables and hybrid datasets, without any constraint.

- The GBF definition is specially designed for discrete variables. If we want to generalize it to the case of continuous variables, we would have to talk about ranges of values. Also, *pgmpy* does not support continuous variables (Table 4.2). Therefore, other libraries would have to be used. So far, with open source software, there is only possibility to do inference with Gaussian BNs, and conditional Gaussian BNs, using *bnlearn* or *LGNpy*. However they have not an extensible design. We find ourselves with a double obstacle to generalize the MRE to continuous variables or hybrid networks: the need to redefine the GBF and the need to implement it using other libraries.

Feature	PyBNesian	bnlearn	pcalg	pgmpy	pomegranate	LGNpy
Extensible design	✓	X	✓[a]	✓	X	X
Representation support						
Discrete BNs	✓	✓	X	✓	✓	X
Continuous BNs						
Gaussian BNs	✓	✓	X	X	X	✓
Kernel density estimation BNs	✓	X	X	X	X	X
Semiparametric BNs	✓	X	X	X	X	X
Hybrid BNs						
Conditional linear Gaussian BNs	✓	✓	X	X	X	X
Hybrid semiparametric BNs	✓	X	X	X	X	X
Conditional BNs	✓	X	X	X	X	X
Dynamic BNs	✓	X	X	✓	X	X
Learning support						
Parameter learning						
Maximum likelihood estimate	✓	✓	X	✓	✓	✓
Bayesian estimate	X	✓	X	✓	X	X
Structure learning algorithms						
Greedy hill-climbing	✓	✓	X	✓	✓[b]	X
PC	✓	✓	✓	✓	X	X
Max–min parent children	✓	✓	X	✓	X	X
Max–min hill-climbing	✓	✓	X	✓	X	X
Learning score fucntions						
K2	X	✓	X	✓	X	X
Bayesian Dirichlet equivalent	✓	✓	X	✓	X	X
Bayesian Information Criterion	✓	✓	X	✓	✓	X
Bayesian Gaussian equivalent	✓	✓	X	X	X	X
Predictive log-likelihood	✓	✓	X	X	X	X
Conditional independence tests						
χ^2	✓	✓	X	✓	X	X
Mutual information	✓[c]	✓	X	X	X	X
Partial linear correlation	✓	✓	✓	✓	X	X
Inference support						
Exact inference	X	X	X	✓	X	✓
Approximate inference	X	✓	X	✓	✓	X

Figure 4.2: Summary of the functionalities implemented by BN library. Source: Atienza et al. (2022)

- Integrate the comparison of queries in other tools that have graphic user interface (GUI).

One of the most powerful tools to construct model-based decision support systems is Elvira. Figure 4.3 shows an explanation option in Elvira, that helps in building med-

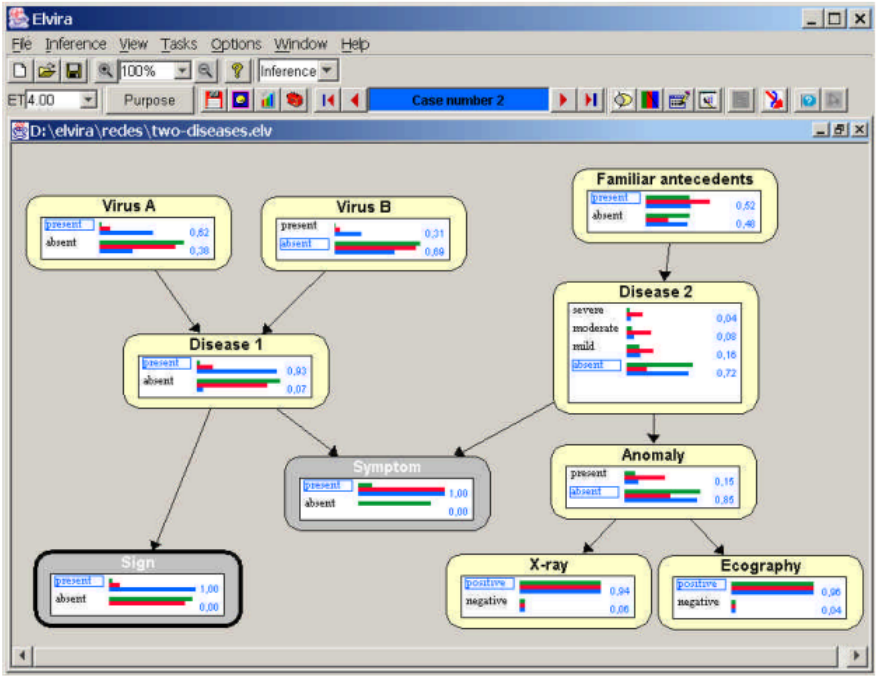


Figure 4.3: Elvira main window in inference mode. A BN for the differential diagnosis of two diseases. Source: Lacave et al. (2007).

ical applications: the simultaneous display of several evidence cases. For instance, it shows how the probabilities of two diseases increase or decrease when several findings are introduced. The finding that the patient has a Symptom increases the probability of both *Disease 1* and *Disease 2* (see red bar in nodes *Disease 1* and *Disease 2*). The presence of a certain *Sign* confirms *Disease 1* and reduces our suspicion of *Disease 2* (see the blue bar in nodes *Disease 1* and *Disease 2*) (Díez (2004)).

We believe that it would be useful to be able to compare queries of interest, and not just posterior probabilities of single variables. As an example, in Figure 4.4 we have made a prototype with the COVID-19 dataset presented above. On the X axis we have the possible combinations of the variables of interest ordered in the form *Reinfections*/*Patients in Hospital*. On the y axis, the probabilities associated with the query $P(\textit{Reinfections}, \textit{Patients in Hospital}|\mathbf{e})$, where the evidence takes the possible values described in the legend.

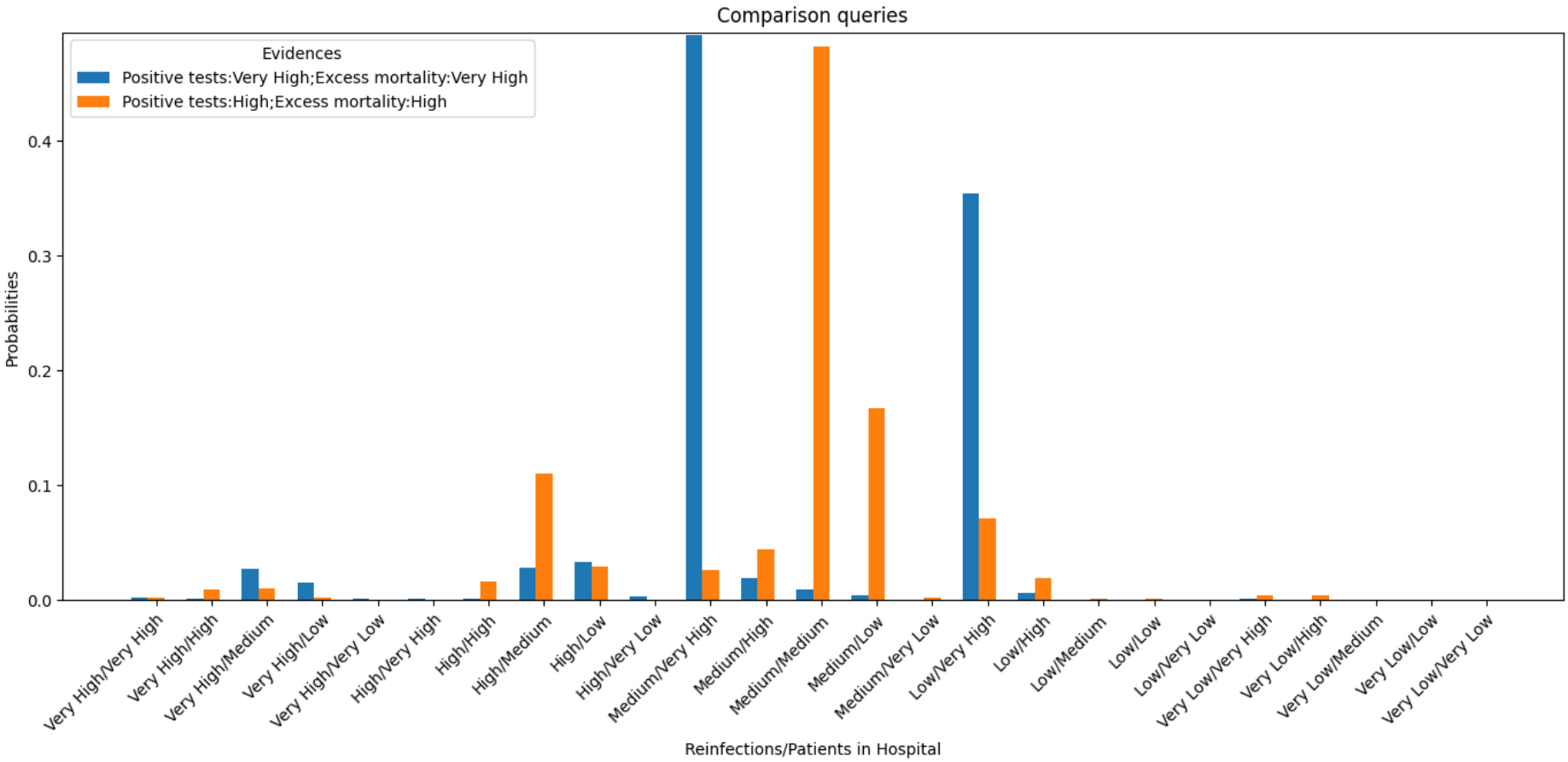


Figure 4.4: Comparison of the queries with different evidences using grouped bar charts.

Bibliography

Ankan, A. and contributors (2022). Pgmpy (version 0.1.22). Computer Software.

Atienza, D., Bielza, C., and Larrañaga, P. (2022). PyBNesian: An extensible python package for Bayesian networks. *Neurocomputing*, 504:204–209.

Beinlich, I. A., Suermondt, H. J., Chavez, R. M., and Cooper, G. F. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In Hunter, J., Cookson, J., and Wyatt, J., editors, *AIME 89, Second European Conference on Artificial Intelligence in Medicine, London, UK, August 29th-31st 1989. Proceedings*, volume 38 of *Lecture Notes in Medical Informatics*, pages 247–256. Springer.

Bertelè, U. and Brioschi, F. (1972). *Nonserial Dynamic Programming*. Academic Press.

Brooks, S. P. and Gelman, A. (1996). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7:434–455.

Campos, L. M., Gámez, J. A., Larrañaga, P., Moral, S., and Romero, T. (2002). Partial abductive inference in Bayesian networks. an empirical comparison between GAs and EDAs. In *Estimation of distribution algorithms*, volume 2 of *Genetic Algorithms and Evolutionary Computation*, pages 323–341. Springer.

Cano, A. and Moral, S. (1995). Heuristic algorithms for the triangulation of graphs. In Bouchon-Meunier, B., Yager, R. R., and Zadeh, L. A., editors, *Advances in Intelligent Computing — IPMU '94*, pages 98–107, Berlin, Heidelberg. Springer Berlin Heidelberg.

Castillo, E. and Kjaerulff, U. (2003). Sensitivity analysis in Gaussian Bayesian networks using a symbolic-numerical technique. *Reliability Engineering & System Safety*, 79:139–148.

Chan, H. and Darwiche, A. (2002). When do numbers really matter? *Journal of Artificial Intelligence Research*, 17:265–287.

Chan, H. and Darwiche, A. (2006). On the robustness of most probable explanations. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 63–71.

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

Constantinou, A. C., Liu, Y., Chobtham1, K., Guo, Z., and Kitson, N. K. (2020). The Bayesys data and Bayesian network repository. [Online].

Cowles, M. K. and Carlin, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904.

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

Dasgupta, D. and Michalewicz, Z. (1997). *Evolutionary Algorithms in Engineering Applications*, chapter 1. Springer.

de Campos, L., Gámez, J., and Moral, S. (1999). Partial abductive inference in Bayesian belief networks using a genetic algorithm. *Pattern Recognition Letters*, 20:1211–1217.

de Jongh, M. and Druzdzel, M. J. (2009). A comparison of structural distance measures for causal Bayesian network models. In *Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science Series*, pages 443 – 456. Academic Publishing House EXIT.

Díez, F. J. (2004). Teaching probabilistic medical reasoning with the Elvira software. *Yearbook of Medical Informatics*, 13:175–180.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning.

Fung, R. and Chang, K.-C. (1990). Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Uncertainty in Artificial Intelligence*, pages 209–219. North-Holland.

Geiger, D. and Heckerman, D. (1994). Learning Gaussian networks. In *Uncertainty Proceedings*, pages 235–243.

Gelsema, E. S. (1995). Abductive reasoning in Bayesian belief networks using a genetic algorithm. *Pattern Recognition Letters*, 16:865–871.

Gelsema, E. S. (1996). Diagnostic reasoning based on a genetic algorithm operating in a Bayesian belief network. *Pattern Recognition Letters*, 17:1047–1055.

Görgen, C. and Leonelli, M. (2020). Model-preserving sensitivity analysis for families of Gaussian distributions. *Journal of Machine Learning Research*, 21(84):3257–3288.

Gunning, D. (2017). Explainable artificial intelligence (xai), darpa/i2o.

Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Uncertainty in Artificial Intelligence*, volume 5, pages 149–163. North-Holland.

Henze, N. and Zirkler, B. (1990). A class of invariant consistent tests for multivariate normality. *Communications in Statistics - Theory and Methods*, 19(10):3595–3617.

High-Level Expert Group on Artificial Intelligence (2019). Ethics guidelines for trustworthy AI.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Jaszczolt, C. (2017). Understanding permanent magnet motors. https://www.controleng.com/articles/understanding-permanent-magnet-motors/.

Jensen, F. V. (1996). *An Introduction to Bayesian Network*. Springer.

Jensen, F. V. and Nielsen, T. D. (2013). *Bayesian Networks and Decision Graphs*. Springer.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *American Association for the Advancement of Science*, 349.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. The MIT Press.

Koopman, T. (2020). Computing Contrastive, Counterfactual Explanations for bayesian Networks. Master's thesis, Utrecht University.

Lacave, C. and Díez, F. J. (2002). A review of explanation methods for Bayesian networks. *The Knowledge Engineering Review*, 17:107–127.

Lacave, C., Luque, M., and Diez, F. J. (2007). Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(4):952–965.

Larrañaga, P. (2002). A review on estimation of distribution algorithms. In *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, pages 57–100. Springer.

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

Lauritzen, S. L. and Spiegelhalter, D. J. (1988a). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50:157–224.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988b). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224.

Leonelli, M., Ramanathan, R., and Wilkersonc, R. L. (2021). Sensitivity and robustness analysis in Bayesian networks with the bnmonitor R package. *arXiv:2107.11785v1*.

Leonelli, M. and Riccomagno, E. (2022). A geometric characterization of sensitivity analysis in monomial models. *International Journal of Approximate Reasoning*, 151:64–84.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Molnar, C. (2019). *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

Nojavan A., F., Qian, S. S., and Stow, C. A. (2017). Comparative analysis of discretization methods in bayesian networks. *Environmental Modelling Software*, 87:64–71.

Parliament, E. (2023). Artificial intelligence act. *EU Legislation in Progress*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2 edition.

Pearl, J. (1997). Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32:245–257.

Plummer, M., Best, N., Cowles, K., and Vines, K. (2006). Coda: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11.

Plummer, M., Best, N., Cowles, K., Vines, K., Sarkar, D., Bates, D. M., Almond, R. G., and Magnusson, A. (2015). Output analysis and diagnostics for mcmc.

Rojas-Guzman, C. and Kramer, M. (1996). An evolutionary computing approach to probabilistic reasoning in Bayesian networks. *Evolutionary Computation*, 4:57–85.

Shachter, R. D. and Kenley, C. R. (1989). Gaussian influence diagrams. *Management Science*, 35(5):527–550.

Soloviev, V. P. (2020). EDAspy (version 1.0.2). https://github.com/VicentePerezSoloviev/EDAspy. Accessed: 2023-05-09.

Soloviev, V. P., Larrañaga, P., and Bielza, C. (2022). Estimation of distribution algorithms using Gaussian Bayesian networks to solve industrial optimization problems constrained by environment variables. *Journal of Combinatorial Optimization*, 44:1077–1098.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 4(65):31–78.

Wen, W. X. (1991). Optimal decomposition of belief networks. *Uncertainty in Artificial Intelligence*, 6:209–224.

Yuan, C., Lim, H., and Littman, M. L. (2011a). Most relevant explanation: computational complexity and approximation methods. *Annals of Mathematics and Artificial Intelligence*, 61:159–183.

Yuan, C., Lim, H., and Lu, T.-C. (2011b). Most relevant explanation in Bayesian networks. *Journal of Artificial Intelligence Research*, 42:309–352.

Yuan, C., Liu, X., Lu, T.-C., and Lim, H. (2012). Most relevant explanation: Properties, algorithms, and evaluations. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 631–638. UAI Press.

Zhang, N. L. and Poole, D. (1994). A simple approach to Bayesian network computations. In *Proceedings of the Canadian Artifical Intelligence Conference*, pages 171–178. UAI Press.

Appendix A

Inference algorithms

A.1 Variable elimination

In order to compare the different available heuristics and choose the best one, the *best_heuristic* function is defined. It calculates the induced width of all the different elimination orderings obtained by the different heuristics and plots the results in a bar chart.

Once we have already chosen the elimination ordering, we extend the query function of the *Variable elimination* class (from *pgmpy*) to plot the optimized subtree after pruning unnecessary variables. In addition to the results in Section 3.1.1, we show both the factors from the list that participate in the construction of the new factor, and the resulting factor (Figure A.1). Everything mentioned above is integrated into a report in pdf format.

A.2 Junction tree clustering algorithm

The different heuristics to decide the variables deletion order for the triangulation are compared using the metric defined in Equation (2.4). This task is performed by the functions *cost_triangulate* and *best_heuristic*.

Once we have the elimination order to do the triangulation, the next step is to construct the junction tree. The sequence of steps that go from the moralization of the network to the junction tree can be visualized with the aid of the function *dibujar*. This function is supported by the *networkx* package, a software for network analysis.

Finally, the algorithm ends with the calibration, and consists of using belief propagation, with an upward and downward pass. In the upward pass we first pick a root and send all messages to the root. When this process is complete, the root can send the appropriate message to all of its children. This algorithm continues until the leaves of the tree are reached, at which point no more messages need to be sent (downward pass). We represent the sending of messages in the upward and downward pass with arrows of two different colors, and labels on each arrow. According to the user's knowledge, more or less detail can be attached to the labels, representing the calculation of the messages. The code for the representations can be seen in function $calibrate\_junction\_tree$. To conclude, in a similar way to the *dibujar* function,

```
Variable to eliminate smoke
Factors involved

+------------+--------------+
| smoke      |   phi(smoke) |
+============+==============+
| smoke(yes) |       0.1000 |
+------------+--------------+
| smoke(no)  |       0.0100 |
+------------+--------------+
+------------+--------------+
| smoke      |   phi(smoke) |
+============+==============+
| smoke(yes) |       0.5000 |
+------------+--------------+
| smoke(no)  |       0.5000 |
+------------+--------------+
+------------+------------+--------------------+
| bronc      | smoke      |   phi(bronc,smoke) |
+============+============+====================+
| bronc(yes) | smoke(yes) |             0.6000 |
+------------+------------+--------------------+
| bronc(yes) | smoke(no)  |             0.3000 |
+------------+------------+--------------------+
| bronc(no)  | smoke(yes) |             0.4000 |
+------------+------------+--------------------+
| bronc(no)  | smoke(no)  |             0.7000 |
+------------+------------+--------------------+
```

(a) List of the potentials involving the variable (smoke) to be eliminated.

```
Phi
 +------------+--------------+
| bronc      |   phi(bronc) |
+============+==============+
| bronc(yes) |       0.0315 |
+------------+--------------+
| bronc(no)  |       0.0235 |
+------------+--------------+
```

(b) Final factor.

Figure A.1: One step in the variable elimination process.

we have another function *dibujar_subtree* that represents the subtree of the junction tree on which the elimination of variables is done.

Everything mentioned above is integrated into a report in pdf format.

A.3 Approximate inference

The added functionality focuses on improving the interpretability of the Gibbs sampling algorithm, showing the convergence of the sampling process. The *pgmpy* library allows to generate Gibbs samples, but does not diagnose its convergence. If we restrict ourselves to open-source software, to this purpose we decided to use the R package *coda*, which provides functions for diagnostic tests of convergence to the stationary distribution of the Markov chain. The user will be able to choose between the Gelman and Robin's, Geweke's or Heilde's test, and in the Gelman case, choose

between the multivariate or univariate shrink factor.

In addition, we want to increase as much as possible the user's flexibility by allowing him to choose between the different sampling algorithms: if there is evidence in the query, choose between 'rejection' and 'likelihood' sampling; if there is no evidence, choose between 'Gibbs' and 'forward' sampling (this sampling functions do not allow evidence in *pgmpy*). *pgmpy* use rejection sampling in presence of evidence, and forward sampling in abscence of evidence.

The way to communicate both libraries, *pgmpy* and *coda*, is through *rpy2*, which acts as an intermediary between R and python. In particular, we use *robjects* to represent any R object in python.

The *GibbsSampling_int* class extends the *pgmpy GibbsSampling* class, adding the *sample_convergencia* function. This function receives as arguments the sizes of the Markov chains, the chosen convergence diagnosis method, and a boolean argument with the Gelman Rubin multivariate option. This function generates three chains (a hyperparameter that had to be fixed due to the lack of flexibility when *pgmpy* communicates with *rpy2*) and saves them in different csv files. Then, it creates *robject* with the necessary R syntax to read the csv files and execute the convergence diagnostics (from *coda*). After that, we have extended the *ApproxInference* class with a function called *query_GibbsChoice*. This function is used to allow making approximate inferences using Gibbs sampling (and is supported by the function *sample_convergencia*).

Appendix B

MPE

B.1 Implementation

The implementation is based on introducing new functionalities to *EDAspy*. Therefore, only these details will be commented, and not all the programming of the EDAs itself.

The *create_widgets* method creates the main window with checkboxes for selecting columns from the DataFrame and an entry field for entering the numeric value 'Increase in density function'. If everything is valid, the *open_values_window* method is called. It opens a new window to enter the values for each evidence variable, and its 'soft margins'.

Once we have all this information, we are ready to instantiate the class myEDA_evidence and the function *minimize* to start the EDA. This class extends the EDA class of the EDAspy library, with the cost function defined in Equation 3.1. In addition, we have implemented an automation of the white list to speed up the procedure.

White list automation

The software offers the user the possibility of

- Defining a *white list* and a *black list*, and
- Automating these lists dynamically during successive iterations, when the results are good enough. In our case, we save the edges for the next iteration if the cost is less than 1, and the increase in the value of the density in mode is greater than 'Increase in density function'. Experimentation tells us that this procedure speeds up and facilitates obtaining good results in successive iterations of the algorithm.

B.2 Initial interaction with the user

In order to facilitate the introduction of input arguments to our programs, we have created a graphical interface with the *tkinter* library (Figure B.1).

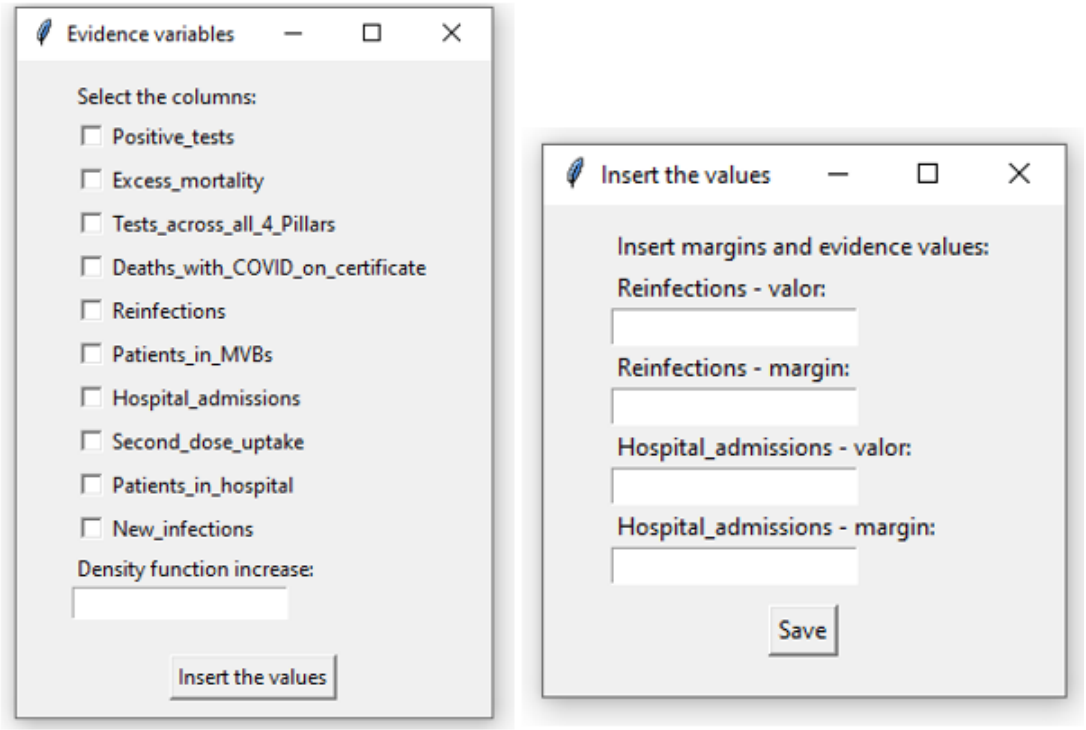


Figure B.1: Interaction with the user: on the left, window to select the variables for which the evidence is known and the increase in the probability density function. On the right, window to select the values of the evidence and the margins (the 'soft evidence' described above in Section 3.2.2).

B.3 Results

B.3.1 Electric motor temperature

For the electric motor temperature dataset, the covariance matrix of the original BN, the covariance matrix of the optimized BN, the mean of the original BN and the mean of the optimized BN can be seen in Tables B.1, B.2, B.3, B.4, respectively.

	Coolant	**Stator wind-ing**	**Stator tooth**	**motor speed**	**pm**	**stator yoke**	**ambient**	**torque**
coolant	0.76	0.96	0,59	0.83	5.84	0,1	-0.05	128.33
stator winding	0.96	126.93	85.99	519.79	84.86	49.46	0	226.83
stator tooth	0.59	85.99	59.82	401.31	63.19	35.01	1.28	138.35
motor speed	0.83	519.79	401.31	60057.82	437.38	239.41	51.33	193.62
pm	5.84	84.98	63.19	437.38	784.66	57.73	4.69	1370.96
stator yoke	0.1	49.46	35.01	239.42	57.73	54.66	1.25	23.02
ambient	-0.05	0	1.28	51.33	4.69	1.25	1.41	-12.41
torque	128.33	226.83	138.35	193.62	1370.96	23.02	-12.41	30093.95

Table B.1: Covariance matrix of the original BN.

	Coolant	**Stator wind-ing**	**Stator tooth**	**motor speed**	**pm**	**stator yoke**	**ambient**	**torque**
coolant	0.43	0.98	0.59	0.46	4.87	0.00	-0.06	53.46
stator winding	0.98	131.68	89.31	129.62	88.33	51.50	0.00	230.23
stator tooth	0.59	89.31	62.23	90.32	65.91	36.54	1.33	139.76
motor speed	0.46	129.62	90.32	17968.65	73.28	53.03	1.92	108.31
pm	4.87	88.33	65.91	73.28	1659.06	57.15	4.91	1145.94
stator yoke	0.00	51.50	36.54	53.03	57.15	60.25	1.31	0.01
ambient	-0.06	0.00	1.33	1.92	4.91	1.31	1.45	-13.31
torque	53.46	230.23	139.76	108.31	1145.94	0.01	-13.14	12586.90

Table B.2: Covariance matrix of the optimized BN.

Coolant	**Stator winding**	**Stator tooth**	**motor speed**	**pm**	**stator yoke**	**ambient**	**torque**
18.89	60.66	50.13	4987.03	58.81	37.25	23.94	25.68

Table B.3: Mean of the original BN.

Coolant	**Stator winding**	**Stator tooth**	**motor speed**	**pm**	**stator yoke**	**ambient**	**torque**
18.94	60.61	53.52	5051.80	61.5121	39.15	24.018	25.20

Table B.4: Mean of the optimized BN.

B.3.2 COVID-19

For the COVID-19 dataset, the covariance matrix of the original BN, the covariance matrix of the optimized BN, the mean of the original BN and the mean of the optimized BN can be seen in Tables B.6, B.7, B.8, B.9, respectively.

	Coolant	Stator wind-ing	Stator tooth	motor speed	pm	stator yoke	ambient	torque
Coolant	1,77E+00	9,80E-01	1,00E+00	1,80E+00	1,20E+00		8,33E-01	2,40E+00
Stator winding	9.80E-01	9.64E-01	9.63E-01	4.01E+00	9.61E-01	9.60E-01		9.85E-01
Stator tooth	1.00E+00	9.63E-01	9.61E-01	4.44E+00	9.59E-01	9.58E-01	9.62E-01	9.90E-01
motor speed	1.80E+00	4.01E+00	4.44E+00	3.34E+00	5.97E+00	4.51E+00	2.67E+01	1.79E+00
pm	1.20E+00	9.62E-01	9.59E-01	5.97E+00	4.73E-01	1.01E+00	9.55E-01	1.20E+00
stator yoke		9.60E-01	9.58E-01	4.51E+00	1.01E+00	9.07E-01	9.54E-01	2.30E+03
ambient	8.33E-01		9.62E-01	2.67E+01	9.55E-01	9.54E-01	9.72E-01	9.32E-01
torque	2.40E+00	9.85E-01	9.90E-01	1.79E+00	1.20E+00	2.30E+03	9.44E-01	2.39E+00

Table B.5: Ratio $covariance_{ij}^{original}/covariance_{ij}^{opt}$ between the covariances of the original BN and the optimized BN. In green, remarked increases in this metric.

	P test	Excess mort	Tests across	Death w COVID	Reinfections	Patient in MVBs	Hospital admission	2nd dose	Patient in H	New inf
P tests	1.21E+09	-2.73E+04	8.87E+09	3.18E+05	1.03E+08	1.68E+06	1.42E+07	5.41E+05	9.42E+07	1.10E+0.9
Excess mort	-2.73E+04	4.61E+04	1.99E+07	2.51E+04	-7.37E+05	-2.73E+05	1.21E+05	-1.20E+04	1.07E+06	-1.73E+04
Tests across	8.87E+09	1.99E+07	2.12E+12	2.18E+09	-5.62E+10	9.35E+09	5.98E+09	-4.57E+08	5.77E+10	8.21E+09
Death w COVID	3.18E+05	2.51E+04	2.18E+09	4.34E+06	-6.77E+07	1.16E+07	9.72E+06	-9.30E+05	9.63E+07	3.94E+06
Reinfections	1.03E+08	-7.37E+05	-5.62E+10	-6.77E+07	1.73E+09	-2.85E+08	-1.82E+08	1.41E+07	-1.76E+09	9.17E+0.7
Patient in MVBs	1.68E+06	-2.73E+05	9.35E+09	1.16E+07	-2.85E+08	4.47E+07	3.78E+07	-3.45E+06	3.65E+08	2.00E+06
Hospital adm	1.42E+07	1.21E+05	5.98E+09	9.72E+06	-1.82E+08	3.78E+07	2.34E+07	-2.05E+06	2.34E+08	1.32E+07
2nd dose	5.41E+05	-1.20E+04	-4.57E+08	-9.30E+05	1.41E+07	-3.45E+06	-2.05E+06	2.00E+05	-2.12E+07	4.90E+05
Patient in H	9.42E+07	1.07E+06	5.77E+10	9.63E+07	-1.76E+09	3.65E+08	2.34E+08	-2.12E+07	2.16E+09	8.77E+07
New inf	1.10E+09	-1.73E+04	8.21E+09	3.94E+05	9.17E+07	2.00E+06	1.32E+07	4.90E+05	8.77E+07	1.01E+09

Table B.6: Covariance matrix of the original BN.

	P test	Excess mort	Tests across	Death w COVID	Reinfections	Patient in MVBs	Hospital admission	2nd dose	Patient in H	New inf
Positive tests	1.55E+09	-6.78E+04	1.27E+10	3.95E+05	1.38E+08	1.93E+06	1.65E+07	5.58E+05	1.19E+08	1.41E+09
Excess mort	-6.78E+04	4.30E+03	2.27E+06	6.23E+04	-1.22E+04	2.63E+05	2.88E+04	-2.67E+02	2.60E+05	3.97E+05
Tests	1.27E+10	2.27E+06	2.05E+11	3.83E+07	1.11E+09	1.89E+08	1.38E+08	5.78E+06	1.10E+09	1.40E+10
Death with COVID	3.95E+05	6.23E+04	3.83E+07	7.44E+05	-9.06E+04	3.08E+06	7.27E+05	-6.59E+03	7.36E+06	6.11E+06
Reinfections	1.38E+08	-1.22E+04	1.11E+09	-9.06E+04	1.43E+07	-1.94E+05	1.19E+06	5.88E+04	8.62E+06	1.24E+08
Patient in MVBs	1.93E+06	2.63E+05	1.89E+08	3.08E+06	-1.94E+05	1.20E+07	2.05E+06	-1.84E+04	2.20E+07	2.24E+07
Hospital adm	1.65E+07	2.88E+04	1.38E+08	7.27E+05	1.19E+06	2.05E+06	2.23E+06	-1.03E+04	2.09E+07	2.18E+07
Second dose	5.58E+05	-2.67E+02	5.78E+06	-6.59E+03	5.88E+04	-1.84E+04	-1.03E+04	8.24E+02	-1.17E+05	4.85E+05
Patient in hosp	1.19E+08	2.60E+05	1.10E+09	7.36E+06	8.62E+06	2.20E+07	2.09E+07	-1.17E+05	2.43E+08	1.56E+08
New infections	1.41E+09	3.97E+05	1.40E+10	6.11E+06	1.24E+08	2.24E+07	2.18E+07	4.85E+05	1.56E+08	1.55E+09

Table B.7: Covariance matrix of the optimized BN.

P test	Excess mort	Tests across	Death w COVID	Reinfections	Patient in MVBs	Hospital admission	2nd dose	Patient in H	New inf
26278.38	9.53	608611.09	230.63	1506.21	857.68	1044.15	45.30	9656.13	24910.40

Table B.8: Mean of the original BN.

P test	Excess mort	Tests across	Death w COVID	Reinfections	Patient in MVBs	Hospital admission	2nd dose	Patient in H	New inf
26920.59	8.67	556539.81	238.04	1931.50	819.49	1088.38	43.50	10209.25	25081.32

Table B.9: Mean of the optimized BN.

Appendix C

MRE

C.1 Implementation

The MRE implementation presents a modular structure. The **main** function is the main entry point of the program called *myEDA* and is responsible for

1. Calling the file functions preprocessing_* to load the data, and do the preprocessing. This function will be customized for each dataset.
2. Creating the main window to select the columns on which you want to declare the evidence using a checkbox, and save those variables in a dictionary. It then creates a new window where the dropdown menus will be displayed. Additionally, when the "Save Selection" button is clicked the selected values from the dropdown menus are retrieved, creating a DataFrame *selected_df* using the selected values.
3. Generating the k-MRE list by calling the functions defined in **functions** (see below).
4. Generating the parallel coordinate graphs to visualize the results obtained. The funciones_plot file is responsible for generating the parallel coordinate graphs from the k-MRE list.

The file **functions** contains all the necessary implementation to generate a Bayesian network model from the data, to calculate the MRE for the inserted evidence, following the forward search algorithm exposed above (Section 3.3.1), and to generate a list with the k-MRE using the algorithm based on strong and weak dominance.

The $initialize$ function is used to initialize the local lookup process. A random variable and a random state are selected to begin the search.

The $neighbors\_add$ function generates neighbors for the current state by adding different values to unobserved variables.

The $neighbors\_change$ function generates neighbors for the current state by changing the values of observed variables.

The GBF function calculates the GBF for the evidence explanation. It uses exact inference to calculate the necessary conditional probabilities.

The get_MRE function compares the GBFs of the neighbors and returns the neighbor with the highest GBF.

The $strong\_dom$ function checks if an explanation is strongly dominated by any instance in the dataframe in terms of explanations and GBF.

The $weak\_dom$ function checks if an explanation is weakly dominated by any instance in the dataframe in terms of explanations and GBF.

The get_kMRE function performs the MRE search algorithm to generate a list of most relevant explanations. It uses the $initialize$, $neighbors\_add$, $neighbors\_change$, get_MRE, GBF, $strong\_dom$ and $weak\_dom$ functions to perform the lookup process. Then, it filters the explanations found by removing those that are strongly or weakly dominated by other explanations. Finally, it returns the three best explanations.

The *learn_model* function induces a Bayesian model from a dataframe using the Hill-Climb search algorithm and the K2 score.

C.2 Initial interaction with the user

The user introduces the input arguments (the number of explanations (k) and the evidence) through a graphical interface (Figure C.1).

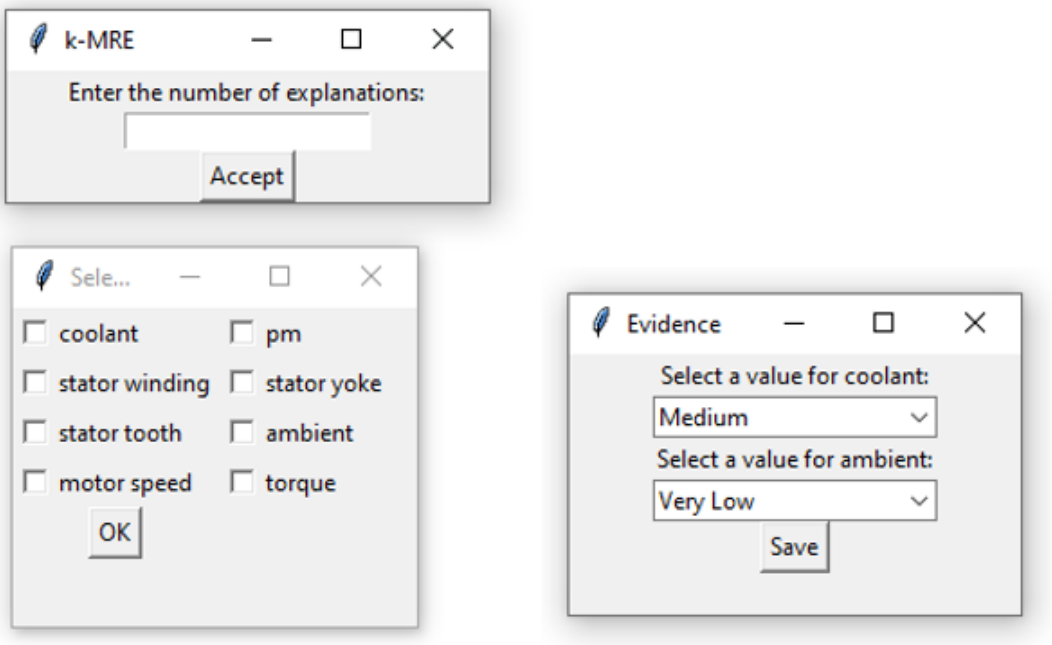


Figure C.1: Selection of the number of explanations, evidence variables and their values.