Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Probabilistic Reasoning Interpretability in Bayesian Networks with Estimation of Distribution Algorithms

Autor: Daniel Zaragoza Pellicer
Tutores: Concha Bielza Lozoya y Pedro Larrañaga Múgica

Madrid, Julio 2024

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Probabilistic Reasoning Interpretability in Bayesian Networks with Estimation of Distribution Algorithms

Julio 2024

Autor: Daniel Zaragoza Pellicer
Tutores: Concha Bielza Lozoya y Pedro Larrañaga Múgica
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Cada vez más la explicabilidad e interpretabilidad están siendo más importantes a la hora de desarrollar modelos de *machine learning*, debido a la necesidad de entender por qué toman las decisiones estos modelos para adoptarlos en distintos sectores. En los últimos años se han desarrollado multitud de técnicas para explicar tanto los modelos como las predicciones, haciendo más fácil la adopción de estos modelos.

Nuestro objetivo en este trabajo es contribuir desarrollando y mejorando técnicas para la interpretabilidad de redes Bayesianas y clasificadores Bayesianos. Para ello propondremos soluciones basadas en algoritmos de estimación de distribuciones.

Existen multitud de técnicas para explicar modelos de *machine learning*, métodos que explican el razonamiento de los modelos, otros que intentan explicar las predicciones o la importancia de las variables. Nosotros nos centraremos en una técnica que busca explicar las predicciones de clasificadores Bayesianos, concretamente en las explicaciones contrafactuales. Estas buscan qué se debe de cambiar en los datos de entrada para obtener una salida deseada. Dentro de este ámbito propondremos una técnica multi-objetivo que busque el contrafactual mínimo, el que más cerca esté de la entrada optimizando varios objetivos. Este método utilizará los algoritmos de estimación de distribuciones para encontrar el mejor contrafactual y lo aplicaremos a un grupo de clasificadores Bayesianos.

En cuanto a las redes Bayesianas, propondremos un algoritmo para resolver el problema de la determinación de la explicación más relevante, que consiste en una técnica que busca la explicación que maximice una métrica a partir de unas evidencias en la red, obteniendo una explicacion buena y concisa. Propondremos un método basado también en algoritmos de estimación de distribuciones, además de implementar otros algoritmos heurísticos para su comparación. También se utilizarán algoritmos existentes para comparar los resultados entre todos los métodos desarrollados.

Abstract

Increasingly, explainability and interpretability are becoming more and more important when developing machine learning models, due to the need to understand why these models make decisions in order to adopt them in different sectors. In recent years a multitude of techniques have been developed to explain both the models and the predictions, making the adoption of these models easier.

Our goal in this work is to contribute by developing and improving techniques for interpretability of Bayesian networks and Bayesian classifiers. We will propose solutions based on estimation of distribution algorithms.

There are many techniques to explain machine learning models, methods that explain the model reasoning, others that try to explain the predictions or the importance of the variables. We will focus on a technique that seeks to explain the predictions provided by Bayesian classifiers, specifically counterfactual explanations. Counterfactuals look for what should be changed in the input data to obtain a desired output. Within this scope we will propose a multi-objective technique that looks for the minimum counterfactual, the one that is closest to the input by optimizing several objectives. This method will use estimation of distribution algorithms to find the best counterfactual and we will apply it to a set of Bayesian classifiers.

Regarding Bayesian networks, we will propose an algorithm to solve the most relevant explanation problem, which consists of a technique that searches for the explanation that maximizes a metric from evidence in the network, obtaining a good and concise explanation. We will propose a method based on estimation of distribution algorithms, in addition to implementing other heuristic algorithms for comparison. Existing algorithms will also be used to compare the results between all the developed methods.

Acknowledgments

To my supervisors, Concha Bielza and Pedro Larrañaga, for their expert advice and guidance.

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

This work has been partially supported by the Spanish Ministry of Science and Innovation through the PID2022-139977NB-I00 and TED2021-131310B-I00 projects, and by the grant to the ELLIS Unit Madrid by the Autonomous Region of Madrid.

Contents

1 Introduction **1**
1.1 Objectives 2
1.2 Contents 3

2 State of the Art **5**
2.1 Bayesian networks 5
2.1.1 Inference 5
2.1.1.1 Exact methods 6
2.1.1.2 Approximate methods 6
2.1.2 Learning 7
2.1.2.1 Parameter estimation 7
2.1.2.2 Structure learning 7
2.2 Explainability 8
2.2.1 Counterfactuals 8
2.2.2 Bayesian networks explainability 9
2.2.2.1 Most relevant explanation 10
2.3 Estimation of distribution algorithms 13

3 Counterfactual Explanations **15**
3.1 Proposal 15
3.2 Experiments 16
3.2.1 Implementation and algorithms 16
3.2.2 Datasets 17
3.2.3 Results with the best model 18
3.2.4 Results with the two best models 19
3.2.5 Results with all models 19
3.2.6 Counterfactual example 20
3.2.7 Counterfactuals in a real dataset 21
3.3 Conclusion 24

4 Most Relevant Explanation **25**
4.1 Proposal 25
4.2 Experiments 26
4.2.1 Implementation and algorithms 26
4.2.2 Bayesian networks used 27
4.2.3 Twelve target nodes 27
4.2.4 Twenty target nodes 28
4.2.5 Brute force comparison 30

4.2.6 Execution example 31
4.3 Conclusion 31

5 Conclusions and Future Work 33
5.1 Conclusions 33
5.2 Future work 33
5.3 Scientific dissemination 34

Bibliography 35

Appendix 39
A MRE example complete results 39

Chapter 1

Introduction

Each year artificial intelligence and machine learning models become increasingly popular and powerful, but at the same time their complexity renders them less interpretable. Explainable artificial intelligence has gained protagonism, enhancing interpretable models and providing post-hoc explanations for black box models. This interpretability is becoming an indispensable requirement for people to trust these models, which is why within this field there is a large variety of methods and models developed to find explanations (Holzinger et al. (2022), Molnar (2022), Linardatos et al. (2020)). Dwivedi et al. (2023) describe the different types of techniques in which explainability can be approached depending on the objective pursued, such as explaining the reasoning of the model, explaining the importance of different variables or explaining specific instances.

Interpretability usually comes with a drop of the model performance, therefore it is necessary to evaluate if in the specific use case it is necessary or not. But at the same time there are a lot of problems for which it is not enough to know the correct prediction, being essential to know why. Some reasons of the importance of interpretability and explainability are (Doshi-Velez and Kim (2017),Molnar (2022)):

- Human curiosity and learning: humans have a mental model of their environment, and this model is updated each time we find something unexpected or new. As humans we want to know why the machine learning model makes a prediction, either to understand the model or to reaffirm our knowledge in the area.
- Safety: when we have a task that entails risk it is important to be able to implement safety measures into the model and test it. So explaining the decisions is essential in order to reveal the possible failures and why they occur.
- Bias: interpretability can be helpful for detecting biases, and importantly why and when these biases occur. Knowing this simplifies the task of changing the training process or the training data to avoid a biased model.
- Social acceptance: in many areas when you try to get people to use a machine learning model, they are skeptical because they do not know if the model's answer is correct or not.
- Debugging: knowing why the model predicts something permits to debug the model easily, not only in the development phase but also after development to

fix mistakes.

All these factors allow for fairer, private, robust and reliable models, features increasingly requested not only by model users but also by the new legislation on artificial intelligence (Selbst and Barocas (2018)).

Having seen the importance of explainability, one may wonder what an explanation is and what characteristics a good explanation has. Miller et al. (2017) conducted a survey in search of answers to these questions. As to what an explanation is, he concludes that it must answer a why-question, which can be answered simply or can require more questions or knowledge of the area. The important questions to find explanations in machine learning are the first ones, since they allow anyone to trust the model or its results.

On the other hand, they found several characteristics present in the best explanations:

- Explanations are contrastive, since many times we do not ask ourselves why a prediction has been made but we want to know why that prediction and not another. This means that users will care not only about explaining the actual predictions but also about explaining why others not.
- Explanations are concrete: it is not expected that the explanation can explain all the causes of an event but that concrete causes are explained. Therefore it is better to give short solutions with few reasons, improving the comprehension of the explanation.
- Explanations are social: depending on the target audience the explanation should be different, being simpler or more complex.
- Explanations focus on the abnormal: people give an explanation of the events that occur to abnormal causes. Therefore, models should include in their explanations the variables that present strange and abnormal data.
- Explanations are truthful and general: good explanations must be reliable and faithful, so if in one case something occurs and the model gives it as an explanations it should be true for more instances.

All these characteristics about explanations will be considered during the development of this work and applied to obtain better explanations. We will focus on explanations based on counterfactuals, which look for what change is necessary in an input to obtain a desired class with a Bayesian classifier, and explanations of evidence in general Bayesian networks according to the concept of most relevant explanation.

1.1 Objectives

The main objective of this work is within the framework of finding explainability and interpretability in machine learning models. The developed algorithms will be applied to Bayesian networks and Bayesian classifiers using estimation of distribution algorithms and compared with existing algorithms.

Within the general explainability, we will seek to develop an algorithm for the computation of counterfactuals with a multi-objective function based on estimation of distribution algorithms. The proposal will be compared with the implementation of genetic algorithms and tested with a set of Bayesian classifiers.

For this purpose, an algorithm based on estimation of distribution algorithms is proposed to solve the most relevant explanation problem in Bayesian networks. The aim is to improve both the search for the best explanation and the speed of the existing algorithms and to implement other heuristic algorithms to obtain a more comprehensive comparison.

1.2 Contents

The document is organized as follows:

- Chapter 2 explains and reviews the state of the art related to our proposal. First, Bayesian networks are introduced in Section 2.1. Subsequently, the latest advances in explainability are discussed in general, specifying in their subsections the methods related to our proposal. Explanations in Bayesian networks are presented, focusing on most relevant explanation and on counterfactual explanations. Finally, estimation of distribution algorithms will be introduced.
- Chapter 3 presents our proposal for finding counterfactual explanations. We introduce the minimization function used, the algorithms applied and the set of models. After this, the experiments carried out and the different results obtained are shown.
- Chapter 4 shows our proposal to solve the most relevant explanation problem, presenting the algorithms used and the different modifications made to improve explanations and computational speed. Then, the experiments and results obtained are shown.
- The conclusions obtained and possible improvements and modifications for future work are outlined in Chapter 5.
- Additionally in the Appendix, more information of the most relevant explanation example is shown.

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 $\mathbf{X} = (X_1, \ldots, X_n)$. The set $\theta = \{P(x_i|\mathbf{Pa}_{X_i})\}$ defines a conditional probability distribution (CPD) for each node of the graph, where $\mathbf{Pa}_{X_i}$ is the set of parents of X_i in graph $\mathcal{G}$. This allows to represent the joint probability distribution $P(\mathbf{x})$ as,

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

Although there are different types of Bayesian networks, such as ones that allow continuous data, hybrid networks or dynamic networks, we will focus on discrete Bayesian networks.

In addition to the explicitly represented conditional probabilities, a BN also implicitly represents conditional independence assertions. Let $X_1, X_2, \ldots, X_n$ be an (ancestral) enumeration of all the nodes in a BN such that each node appears after its parents, and let $\mathbf{Pa}_{X_i}$ be the set of parents of a node X_i. So each variable X_i is conditionally independent of the variables in $\{X_1, X_2, \ldots, X_{i-1}\}$ given its parents.

2.1.1 Inference

Inference refers to finding the probability of any variable conditioned on a given evidence or fixed values for some of the variables in the BN, i.e. $P(X_i|\mathbf{e})$. Inference also refers to finding values of a set of variables that best explain the observed evidence, called abductive inference.

Referring to the first definition of inference, we can distinguished between exact inference and approximate inference methods.

2.1.1.1 Exact methods

Exact inference algorithms are designed to give an exact answer to the probabilistic query. While these methods are tractable for many real world applications, they are limited by its worst case exponential performance and they are *NP*-hard (Cooper (1990)). The brute force approach is conceptually simple but computationally complex, often very inefficient and intractable in real problems. Therefore other methods have been proposed:

- Variable elimination (Zhang and Poole (1994)). The idea behind this method is to successively remove variables from a BN while maintaining its ability to answer the query of interest. Variable elimination works with local computations, giving importance to the elimination ordering, but finding an optimal ordering is *NP*-hard (Bertele and Brioschi (1972)).

- Junction tree. This method entails the creation of the junction tree (moralise, triangulate the moral graph, obtain the cliques, create the junction tree and its separators and compute the junction tree parameters) and the message passing algorithm (Lauritzen and Spiegelhalter (1988)) for the calculation of probabilities.

2.1.1.2 Approximate methods

Approximate inference algorithms are designed to give an approximate answer to the probabilistic query, since exact methods are intractable with large and dense networks. These methods use the network to generate a large number of cases from the network distribution, estimating $P(X_i|\mathbf{e})$ from the generated cases by counting observed frequencies in the samples. As the number of cases increases, the estimated probability converges to the exact one, but Dagum and Luby (1993) demonstrated that approximate inference in BNs within an arbitrary tolerance or accuracy is also *NP*-hard. The main methods are:

- Forward sampling. It is the simplest approach for generating samples, where the method samples the nodes using the topological ordering of $\mathbf{X}$ in the BN. With this, each time we sample a node we have values for all of its parents, so then we can sample from the distribution defined by the CPD and by the chosen values for the node's parents.

- Likelihood weighting (Shachter and Peot (1990)). It takes into account the values of the observed nodes, forcing them to these values. That means that when we come to sample an observed node, we simply set it to its observed value. When we consider multiple observations and we want our sampling process to set all of them to their observed values, it is necessary to consider the probability that each of the observation nodes would have in the observed values. This is done by assigning weights to the samples depending of this probability.

- Gibbs sampling (York (1992)). This method generates a sequence of samples which is constructed so that, although the first sample may be generated from the prior, successive samples are generated from conditional distributions that probably get closer and closer to the desired posterior.

2.1.2 Learning

The problem of fitting BNs requires learning both the structure of the graph $\mathcal{G}$ and the parameters θ. To learn the structure of the network we need to find the underlying conditional independence relationships between our variables. This in turn fixes the number of parameters of our model and, after making some assumptions on the distributions of our variables, allows us to find the most appropriate values for θ. Learning BNs is a very broad topic that depends on the specific goal and structure searched (Daly et al. (2011)).

2.1.2.1 Parameter estimation

Let us assume that we already have a graph structure $\mathcal{G}$ and a dataset $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m\}$, where $\mathbf{d}_i = (x_1^i, x_2^i, \ldots, x_n^i)$ is a row in our dataset with the values of all variables $\mathbf{X} = (X_1.X_2, \ldots, X_n)$ in our graph. In this scenario, we want to learn the network parameters θ defined by $\mathcal{G}$ that best fit our data $\mathcal{D}$. We can typically perform this automatically from data via maximum likelihood estimation or Bayesian estimation.

Maximum likelihood estimation

Maximum likelihood estimation (MLE) is one of the most common methods for parameter estimation. With this method, our aim is to obtain the parameter set θ by using a point estimate based on all instances of $\mathcal{D}$. To know whether a set θ fits some data properly we use the likelihood function, but in practice it is often convenient to use the the log-likelihood function. MLE will search for the set of parameters $\hat{\theta}$ that maximizes the log-likelihood of the data:

$$\hat{\theta} = \underset{\theta}{max}\ l(\theta : \mathcal{D}) \tag{2.2}$$

This poses a high dimensional optimization problem, even for BNs with a low number of nodes, since we need to optimize over all the CPDs in the network. But there exists a factorization decomposing it into a summation of independent terms, one for each CPD in the network, and then combine these individual solutions to get the MLE.

Bayesian estimation

These methods estimate the parameters by using a point, such as the MLE, now there is a measure of uncertainty, and prior knowledge can be incorporated into the learning process. The prior knowledge is introduced via a prior distribution over the parameters, and uncertainty is reflected in its posterior distribution. The posterior distribution encodes updated beliefs once prior knowledge and data have been taken into consideration. For a fixed structure $\mathcal{G}$, the posterior distribution of the parameters is given with the Bayes theorem. Then Bayesian estimation gives a decomposition, similar to MLE, for obtaining the best set of parameters for the data.

2.1.2.2 Structure learning

The task of learning the graph structure of a BN is a complex problem where the search space of possible DAGs grows super-exponentially with the number of nodes.

In order to find appropriate structures, many authors have proposed different structure learning algorithms, but we will focus on the score-based method.

Score-based method

Score-based algorithms define the task of finding the best graph structure as an optimization problem. If we define a score that identifies how well does a graph fit some data, then we can maximize this score to find an optimal structure. However, the problem of finding the optimal network is NP-hard (Cooper (1990)) and algorithms have to resort to heuristics in the search phase to find the best possible structure instead of the optimal one. These methods are divided into two phases:

- Score phase, consists of a score that will be maximized. Different scores have been used such as the log-likelihood, Akaike information criterion or Bayesian information criterion.
- Search phase, consists of searching the space of possible networks as an optimization problem. The common procedure is to apply some heuristic to search for good candidate networks, such as hill-climbing or tabu.

2.2 Explainability

Explainability and interpretability can be achieved in different ways depending on our objective, type of data or model. The different approaches can be divided into categories, where each technique is classified into one type in each of them (Linardatos et al. (2020),Molnar (2022)):

- Model specific or model agnostic: model specific methods can be applied to a single model or group of models, taking advantage of the capabilities of that model and its characteristics, while model agnostic methods can be applied to any model.
- Local or global: global methods search to explain the overall model while local ones explain a single prediction.
- Intrinsic or post-hoc: intrinsic methods provide an intrinsic explainability, resulting in interpretable models, while post-hoc methods analyze the model after training.
- Result of an interpretability method: the interpretability methods can be differentiated given what they provide. Some methods give a feature summary, either statistic or visual, showing their interpretation and significance. Other methods interpret the model internals, like the learned tree structure of decision trees, or give interpretable models as a result, where the model itself can be interpreted, like BNs. Some methods explain data points, giving explanations to given instances or modifying them to gain interpretability.

2.2.1 Counterfactuals

Among the techniques explaining the reasoning of the model, we will focus on counterfactual explanations within supervised classification. They answer the question of which minimal change is needed in the input data to obtain a desired output. Obtaining good counterfactual explanations may not be straightforward, hence many methods have been developed (Verma et al. (2020), Guidotti (2022)). Heuristic search based approaches usually involve minimizing a cost function accounting for the sought objective(s). Wachter et al. (2017) is one of the first works to propose to optimize a function, combining the distance between the input and the generated counterfactual, and the counterfactual estimated output class probability. Dhurandhar et al. (2018) add plausibility in their cost function and Mothilal et al. (2020) focus on plausibility and solution diversity, penalizing similar solutions. As for the heuristic search used, Lash et al. (2017) use genetic algorithms and local search, Moore et al. (2019) apply gradient-descent methods and Lucic et al. (2020) use Monte Carlo simulation. Many existing methods combine different objectives into a single one to fulfil the desired properties, at the expense of losing information when combined. Other methods pose the problem as a multi-objective optimization one.

Different authors have defined counterfactual explanations from a variety of points of view. For example, Guidotti (2022) formalized the problem with the objective of minimizing the change of the input variables for achieving a different prediction.

Definition 1 ***(Guidotti (2022))*** *Given a classifier ϕ that outputs the decision $c = \phi(\boldsymbol{x})$ for an instance $\boldsymbol{x}$, a counterfactual explanation consists of an instance $\boldsymbol{x}'$ such that the decision for ϕ on $\boldsymbol{x}'$ is different from c, i.e., $\phi(\boldsymbol{x}') \neq c$, and such that the difference between $\boldsymbol{x}$ and $\boldsymbol{x}'$ is minimal.*

To find the best possible explanation, a counterfactual explainer will be used, which will be in charge of finding $\mathbf{x}'$ that meets the constraints of Definition 1. Thus, the only thing that remains to be defined is what it means for the difference between $\mathbf{x}$ and $\mathbf{x}'$ to be minimal. Each method for calculating counterfactuals has defined which objectives are important to obtain that minimal difference while being a high quality counterfactual. The objectives that we consider most important are:

- *Validity*: the classification output has to be different from the original one.
- *Minimality*: the number of variables or the distance between $\mathbf{x}'$ and $\mathbf{x}$ should be as small as possible. Each counterfactual explainer will have its own aim.
- *Plausibility*: $\mathbf{x}'$ should be coherent with an observation population, i.e., $\mathbf{x}'$ can occur with the given data.

2.2.2 Bayesian networks explainability

In the area of BNs, specific methods have been developed to provide explanations. Unlike many machine learning methods that are mostly predictive methods, BNs can be used for both prediction and explanation, with a good representation of a domain. Explanations in BNs can be classified into explanation of reasoning, explanation of model and explanation of evidence (Lacave and Díez (2002)). The objective of the explanation of reasoning is to explain the reasoning process used to produce results, so you can believe the obtained result. The explanation of model seeks to show the knowledge encoded in the network in a understandable form, such as visual aids. The goal of the explanation of evidence is to explain why some observed variables are in their particular states using the other variables available.

We will focus on explanation of evidence, searching what nodes explains the evidence.

A multitude of methods have been developed to find these explanations, where some of them simplify the problem and focus on singleton explanations. For example, it is often assumed that the fault variables are mutually exclusive and collectively exhaustive, and there is conditional independence of evidence given any hypothesis (Heckerman et al. (1995)). These explanations are not capable of fully explaining the evidence, since in many cases their explanation requires multiple causes. Therefore, methods capable of finding multivariate explanations have been developed. Maximum a posteriori assignment (MAP) finds a complete instantiation of a set of target variables that maximizes the joint posterior probability given evidence on the other variables (partial abduction). Most probable explanation (MPE) (Pearl (1988)) is similar to MAP except that MPE defines the target variables to be all the unobserved variables (total abduction). The main drawback of these methods is that they often produce hypotheses that are overspecified and may contain irrelevant variables in explaining the given evidence. Due to this, various pruning techniques have been used to avoid overly complex explanations, where these methods can be divided into two categories: pre-pruning and post-pruning. Pre-pruning methods use the context specific independence relations represented in BNs to prune irrelevant variables. Post-pruning methods first generate explanations using methods such as MAP or MPE and then prune variables that are not important. Although various improvements and methods have been proposed, many of them continue to produce explanations that are either too simple or too complex.

2.2.2.1 Most relevant explanation

Yuan et al. (2011b) propose the most relevant explanation (MRE) method which finds a partial instantiation of the target variables that maximizes the generalized Bayes factor (GBF), this being the best explanation given the evidence.

MRE uses the GBF because it fulfills two indispensable characteristics of a good explanation, which are precision and consistency. This implies that an explanation should be precise which means that it should reduce the surprise about the explanandum as much as possible, while consistency means that the explanation should contain only the most relevant variables to explain the evidence, avoiding irrelevant variables.

The Bayes factor is the ratio between the likelihoods of a hypothesis and an alternative hypothesis (Fisher (1935), Jeffreys (1961)). The Bayes factor assigns large values when the probabilities approach certainty. The problem with the Bayes factor is that it is difficult to use it to compare more than two hypotheses and therefore it is necessary to make pairwise comparisons between multiple hypotheses. Because of this limitation, the Bayes factor is generalized to be able to compare different hypotheses.

Definition 2 *Generalized Bayes factor (GBF) of an explanation* $\boldsymbol{x}$ *for a given evidence* $\boldsymbol{e}$ *is defined as*

$$GBF(\boldsymbol{x}; \boldsymbol{e}) \equiv \frac{P(\boldsymbol{e}|\boldsymbol{x})}{P(\boldsymbol{e}|\bar{\boldsymbol{x}})} \tag{2.3}$$

where $\bar{\boldsymbol{x}}$ *means the set of all alternative hypotheses of* $\boldsymbol{x}$*.*

It should be noted that it is not necessary to compute $P(\mathbf{e}|\bar{\mathbf{x}})$ directly when calculating

$GBF(\mathbf{x};\mathbf{e})$. Instead we compute

$$GBF(\mathbf{x};\mathbf{e}) = \frac{P(\mathbf{x}|\mathbf{e})(1 - P(\mathbf{x}))}{P(\mathbf{x})(1 - P(\mathbf{x}|\mathbf{e}))} \tag{2.4}$$

Yuan et al. (2011b) present different theoretical properties of the GBF that allow us to identify the most relevant target variables for the explanation.

Handling extreme values

GBF assigns much more weight to probabilities in ranges close to 0 and 1, since these values ensure certainty about the event. Therefore, any change in the probability from a value down to 0 or up to 1 will imply a much higher weight even if the change is minimal. Some special cases are also taken into account:

- If $P(\mathbf{x}) = 0.0$, then $P(\mathbf{x}|\mathbf{e})$ will be 0 as well and the GBF will be 0.
- If $P(\mathbf{x}) = 1.0$ and $P(\mathbf{x}|\mathbf{e}) = 1.0$, the GBF will be 0, since the explanation will be true whether the evidence is there or not, therefore it does not provide information.
- If $P(\mathbf{x}) < 1.0$ and $P(\mathbf{x}|\mathbf{e}) = 1.0$, the GBF will be infinite.

Monotonicity of GBF

Monoticity is studied by taking into account the difference between the posterior and prior probabilities and the belief update ratio. For the first one, 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 2.1 shows the GBF against the prior probability when the difference between the posterior and prior probabilities is fixed. It is clearly seen how the GBF is higher when the probability changes close to 0 and 1.

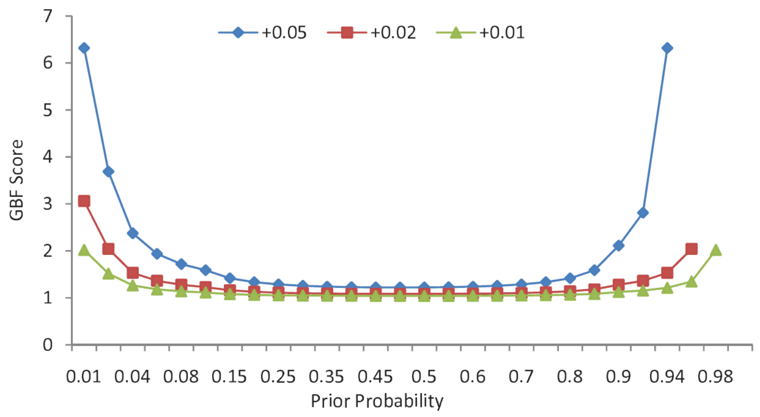


Figure 2.1: The GBF as a function of the prior probability given a fixed increase in the posterior probability from the prior. The different curves correspond to different probability increases (Yuan et al. (2011b))

The belief update ratio is defined as follows,

Definition 3 *Assuming $P(\boldsymbol{x}) \neq 0$, the belief update ratio of **x** given **e**, r(**x**;**e**), is defined as*

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

Regarding the belief update ratio, we have the following theorem,

Theorem 1 *(Yuan et al. (2011b)) For an explanation **x** with a fixed belief update ratio r(**x**;**e**) > 1.0, GBF(**x**;**e**) is monotonically increasing as the prior probability P(**x**) increases.*

Figure 2.2 shows the 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, the GBF provides more discriminant power than the belief update ratio.

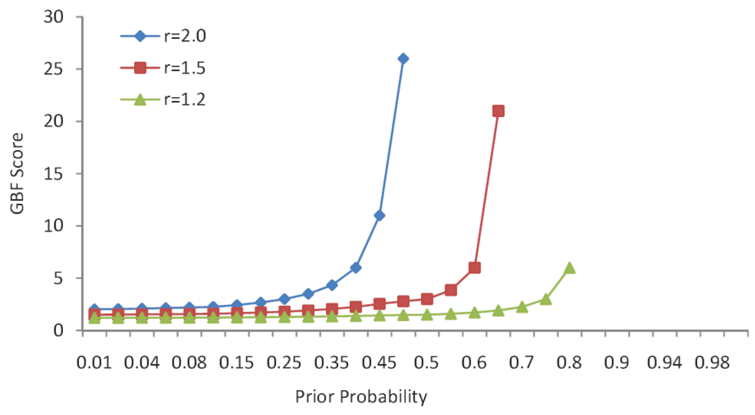


Figure 2.2: The GBF as a function of the prior probability when the belief update ratio is fixed. The different curves correspond to different belief update ratios (Yuan et al. (2011b))

Achieving conciseness in explanations

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

$$GBF(\boldsymbol{x}, y; \boldsymbol{e}) < GBF(\boldsymbol{x}; \boldsymbol{e}). \tag{2.6}$$

Theorem 2 captures the intuition that conditionally independent variables add no additional information to an explanation in explaining the evidence. Note that these properties are all relative to an existing explanation. It is possible that a variable is independent from the evidence given one explanation, but becomes dependent on the evidence given another explanation.

With GBF defined and the theoretical properties presented we are able to give a concrete definition for MRE for finding explanations for a given evidence in BNs.

Definition 4 *Let **M** be a set of target variables, and **e** be the partial evidence on the remaining variables in a Bayesian network. Most relevant explanation is the problem of finding an explanation **x** for **e** that has the maximum generalized Bayes factor score GBF(**x**;**e**), i.e.,*

$$MRE(\boldsymbol{M}; \boldsymbol{e}) \equiv argmax_{\boldsymbol{x}, \emptyset \subset \boldsymbol{X} \subseteq \boldsymbol{M}} GBF(\boldsymbol{x}; \boldsymbol{e}) \tag{2.7}$$

Although MRE is general enough to be applied to any probabilistic distribution model, MRE's properties make it especially suitable for BNs. BNs model the conditional independence relations between the random variables of a domain so that we not only obtain a concise representation of the domain but also have efficient algorithms for reasoning about the relations between the variables.

2.3 Estimation of distribution algorithms

Estimation of distribution algorithms (EDAs) (Larrañaga and Lozano (2002)) are evolutionary algorithms that, at each generation, explore the solution space by sampling a probabilistic model constructed from the best solutions found. The EDA procedure is outlined in Algorithm 1. EDAs work with a population of candidate solutions, which are scored using a cost function (line 3). This function ranks the solutions and the best ones are selected to learn the probabilistic model (lines 4 and 5). Then a new population is sampled from the model (line 6) and the process is repeated until a termination criterion is met (line 2). From this basic procedure a multitude of variants have been developed, adapting it to different data types and more complex problems.

Algorithm 1 EDA procedure

```
   Input: Population size, cost function, selection rate
   Output: Best individual and cost
1: Initial population
2: for $t = 1, 2, \ldots$ until stopping criterion is met do
3:     Evaluate population using a cost function
4:     Select individuals
5:     Learn a probabilistic model from the best individuals
6:     Sample new individuals from the probabilistic model
7: end for
```

Note that EDAs follow a process similar to genetic algorithms but eliminating crossover and mutation, since the probabilistic model will take care of generating new solutions. There are many different EDAs (Hauschild and Pelikan (2011)), either for continuous or discrete data, or depending on the probabilistic model they learn. In this work we will focus on discrete data and will use two algorithms, univariate marginal distribution algorithm (UMDA) and estimation of Bayesian network algorithm (EBNA). UMDA (Mühlenbein and Paass (1996)) is an EDA that assumes that all variables are independent and thus their joint probability can be factorized as a product of univariate marginal probabilities, while EBNA (Etxeberria and Larrañaga (1999)) uses BNs to capture and exploit the dependencies between variables in the solution space.

EDAs will be used in this work to find the best solutions for counterfactuals and most relevant explanation problems, where modifications and adaptations to these problems will be proposed.

Chapter 3

Counterfactual Explanations

3.1 Proposal

Let $\mathbf{X} = (X_1, \ldots, X_n)$ denote the predictor (categorical) features from N labeled instances and $\mathcal{D} = \{(\mathbf{x}^1, c^1), \ldots, (\mathbf{x}^N, c^N)\}$ the dataset, where for each $\mathbf{x}^i = (x_1^i, \ldots, x_n^i), i = 1, \ldots, N$, we have the respective value c^i of a class variable C with labels in the domain $\Omega_C = \{c_1, \ldots, c_R\}$. The domain of each X_i is accordingly denoted Ω_{X_i}.

Definition 5 *Given $\phi : \Omega_{X_i} \times \cdots \times \Omega_{X_n} \to \Omega_C$ a Bayesian classifier (Bielza and Larrañaga (2014)), and $(\boldsymbol{x}^*, c)$ an instance of $\mathcal{D}$, a counterfactual explanation $\boldsymbol{x}'$ for $\boldsymbol{x}^*$ is a solution of the multi-objective problem*

$$\min_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x})) \tag{3.1}$$

where:

- $f_1(\mathbf{x}')$ is the prediction objective, defined as the Manhattan distance between the class-posterior distribution of the counterfactual $\mathbf{x}'$, $\mathbf{P}(C|\mathbf{x}')$, and the distribution corresponding to the desired outcome (i.e., a vector $\mathbf{P}'$ with all zeros except for a 1 in the position corresponding to the desired class),

$$f_1(\mathbf{x}) = \sum_{i=1}^{|\Omega_C|} |\mathbf{P}'_i - \mathbf{P}_i(C|\mathbf{x}')| \tag{3.2}$$

- $f_2(\mathbf{x}')$ is the distance objective, defined as the distance between the input instance $\mathbf{x}^*$ and the counterfactual $\mathbf{x}'$, calculated using the Gower distance (Gower (1971)) d_G,

$$d_G(\mathbf{x}', \mathbf{x}^*) = \sum_{i=1}^{n} d_i(x'_i, x^*_i)/n \tag{3.3}$$

  where the distance d_i per feature in the summation varies depending on whether the feature is categorical, where the distance d_i is 0 if $x'_i = x^*_i$ and 1 otherwise, or numeric, where we use the normalized Manhattan distance for d_i.

- $f_3(\mathbf{x}')$ is the number of feature changes from the input instance $\mathbf{x}^*$ to $\mathbf{x}'$,

$$f_3(\mathbf{x}') = \sum_{i=1}^{n} \mathbb{I}_{x'_i \neq x^*_i} \tag{3.4}$$

- $f_4(\mathbf{x}')$ is the plausibility of $\mathbf{x}'$ given $\mathcal{D}$, which is the distance between $\mathbf{x}'$ and its nearest instance in $\mathcal{D}$, given by the same distance as $f_2(\mathbf{x}')$.

These objectives will yield high quality counterfactuals, as they meet the objectives described in Section 2.2.1. The prediction objective (f_1) will allow the validity of the solutions, the distance objective (f_2) and the number of feature changes (f_3) will give minimality to the counterfactual and the plausibility objective (f_4) will give plausibility.

Our proposal is to approach this multi-objective problem with an EDA (MOEDA), where the individual selection is based on non-dominated sorting and crowding distance. Non-dominated sorting involves ranking solutions based on Pareto dominance, where a solution is non-dominated if no other solution is better in all objectives. Solutions are categorized into different fronts, where the first front consists of non-dominated solutions, the second front consists of solutions dominated only by those in the first front, and so on. Crowding distance is a measure used to maintain diversity within each front by estimating the density of solutions surrounding a particular solution. It is calculated based on the average distance of a solution to its neighbors in the objective space. Together, these methods ensure that the algorithm not only converges towards the Pareto front but also maintains a diverse set of solutions. Dandl et al. (2020) used the genetic algorithm NSGA2 (Deb et al. (2000)) to optimize a related function with four objectives. We will compare this algorithm and its single-objective counterpart against our proposal.

To compute the counterfactuals we build a group of models with five Bayesian classifiers, see Figure 3.1. The classifiers used are naive Bayes (NB), semi-naive Bayes (SNB), tree augmented naive Bayes (TAN), hill-climbing tree augmented naive Bayes (TAN-HC) and K-dependence Bayesian classifier (KDB). NB assumes that the predictive variables are conditionally independent given the class, SNB relaxes the NB assumption by allowing dependencies within some groups of variables, TAN uses a tree structure for the dependencies of the variables, TAN-HC finds this structure in a wrapper-like manner, with hill-climbing search, and KDB allows each variable to have K parents. To ensure that the generated counterfactual is as good as possible, the classifiers are first filtered based on whether their predicted class is correct. Only in this case the counterfactual will be computed. After this selection, the models are sorted by their classification accuracy and the model with the highest accuracy is used. Alternatively, we can also use the results of more than one model, so different solutions can be obtained. Note that there is a possibility that no solution is found as no model passes the first filter; however this restriction will allow to avoid generating counterfactuals that are not accurate.

3.2 Experiments

3.2.1 Implementation and algorithms

The implementation for counterfactual computation can be found on GitHub[1]. It is implemented in Python using the libraries Pymoo (Blank and Deb (2020)) for genetic algorithms and EDAspy (Soloviev et al. (2024)) for EDAs. The Bayesian classifiers

[1]`https://github.com/DanielZaragozaP/counterfactual_ensemble`.

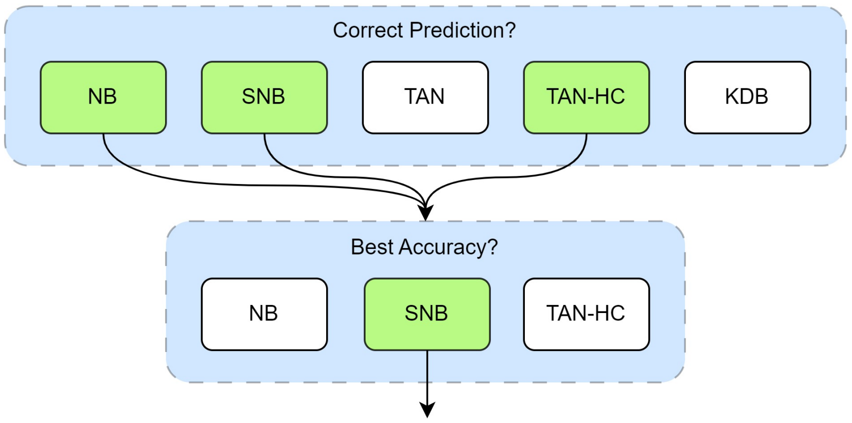


Figure 3.1: Model filtering and selection. NB=Naive Bayes, SNB=Semi-naive Bayes, TAN=Tree augmented naive Bayes, TAN-HC=Hill-climbing tree augmented naive Bayes, KDB=K-dependence Bayesian classifier

were implemented in R in the bnclassify package (Mihaljević et al. (2018)). All experiments were conducted on the same hardware (Intel i5-12500H and 16GB RAM).

Four different algorithms have been used to find the best counterfactual, splitting them into single-objective and multi-objective:

- The single-objective algorithms used are a basic genetic algorithm (GA) and a UMDA, where both aim at minimizing only the f_2 distance.
- The multi-objective algorithms are NSGA2 and MOEDA, the latter based also on a UMDA. Both algorithms will use the four objectives described in Section 3.1.

The results will consist of the average of all runs between datasets, where each run with a dataset and algorithm will consist in 100 executions with different inputs. Differences in prediction (f_1), distance (f_2), plausibility (f_4) and run time will be shown, where the objective (f_3) of minimizing the number of variable changes will not be presented, since in general a smaller distance implies fewer variable changes.

3.2.2 Datasets

The selected datasets have all discrete variables and without missing values. The datasets have been obtained from the UCI Machine Learning Repository (Kelly et al. (2023)) and from the OpenML repository (Vanschoren et al. (2014)). The datasets have been selected to contain different number of instances and features to see how the algorithms performs under different situations, see Table 3.1. In addition, although most of them are binary classification problems, some contain more than two classes to see how this affects the obtained counterfactuals. In each dataset, 90% of the data was used for training and the remaining 10% for testing. The counterfactuals were calculated from the test data, specifically using 100 instances, except for datasets where 10% of instances is lower than 100, where we used all test instances instead. The counterfactual comparisons are calculated taking those test data as an input and a random class as the desired class.

Table 3.1: Description of benchmark datasets

Dataset	#Instances	#Features	#Classes
Tic-tac-toe	958	9	2
Car evaluation	1728	6	4
Chess (kr vs kp)	3196	35	2
Mushroom	8124	22	2
Nursery	12960	8	3
Monk 1	556	6	2
Monk 2	601	6	2
Monk 3	554	6	2
Letter	20000	16	26
Phishing Websites	11100	30	2

3.2.3 Results with the best model

The first comparison contrasts the results of the single-objective EDA with the GA and then the two multi-objective counterparts, all using the filtered model with the highest accuracy. In Table 3.2 we observe the average gain (in %) of the EDAs with respect to the GAs in each of the objectives. In the single-objective case (first row), where only the distance is taken into account, the EDA obtains an average improvement of 33.88%, a considerable improvement with respect to the GA. Moreover, indirectly we obtain a 19.29% improvement in the prediction while maintaining an almost identical plausibility. In the multi-objective scenario (second row), NSGA2 obtains a slight improvement in distance with respect to MOEDA. On the other hand in the case of prediction it is observed that MOEDA obtains a much better prediction while plausibility is higher in NSGA2. This means that MOEDA is more confident that its counterfactuals are correctly classified since they obtain a probability of more than double of being the class that is being searched. It should be noted that in prediction and plausibility there is a higher variation of values than in distance, so depending on the dataset, it is possible to obtain better results with one or the other. Also MOEDA works better with datasets with more variables while NSGA2 obtains better results with few variables.

Table 3.2: Percentage of average improvement (per objective) of EDAs vs GAs, in single-objective and multi-objective problems

	Distance	Prediction	Plausibility
EDA vs GA	33.88%	19.29%	−1.05%
MOEDA vs NSGA2	−6.98%	169.36%	−26.69%

Another factor to take into account is the execution time of each algorithm. Note that if more than one model is used it will be necessary to sum the times taken by all models. Table 3.3 shows the average time per model over all the datasets used. The plausibility calculation slows down considerably the execution of the multi-objective algorithms, since it has to go through all training data in order to find the closest instance. It is observed that EDA takes on average half the time as the GA while in the multi-objective case the distance is reduced, although MOEDA takes slightly less time than NSGA2. The important detail here is that if you do not need the results with the best predictions and plausibilities you could run an EDA with all five Bayesian classifiers in the same time as a MOEDA with only one model and this is even more evident as the number of instances grows.

Table 3.3: Average execution time and standard deviation of each algorithm over all datasets

	EDA	GA	MOEDA	NSGA2
Time (s)	1.62 ± 1.72	2.89 ± 2.20	6.22 ± 5.36	6.87 ± 6.06

3.2.4 Results with the two best models

Rather than using the best model (Figure 3.1), we now observe what happens if we use more than one model to calculate the counterfactual. It is worth noting that some models may achieve the same or almost identical accuracy, this will depend on the dataset. First we analyze the possible gain when calculating the counterfactuals by adding the second best model, see Table 3.4. It can be seen how the distance improves with all the algorithms between 13% and 22%, while the prediction worsens in general and plausibility improves a little bit. Although confidence in the prediction is lost by using an additional model, this gain in distance can mean a considerable improvement in the counterfactual obtained, also maintaining plausibility. MOEDA is the algorithm that takes the most advantage with the use of two models, being the one that obtains the biggest improvement in distance and plausibility.

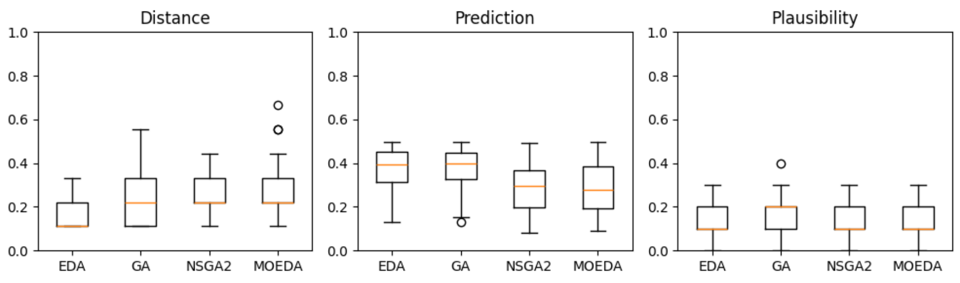
Table 3.4: Gain (in%) when using the results of two best models versus only the best model

	Distance	Prediction	Plausibility
EDA	14.44%	−25.77%	5.85%
GA	14.42%	−23.20%	1.09%
MOEDA	22.52%	−34.70%	20.13%
NSGA2	13.09%	−23.11%	−0.09%

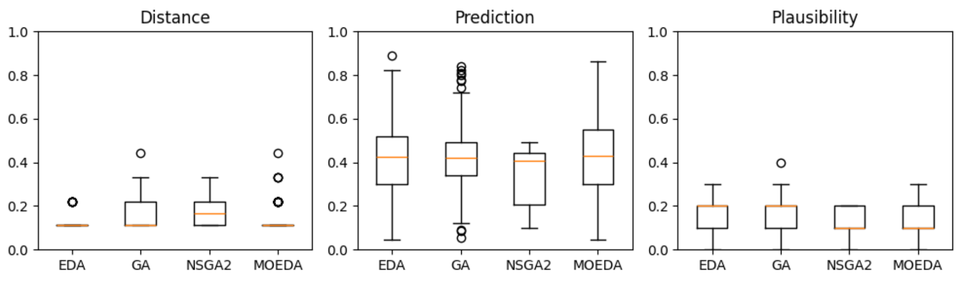
To observe in a more visual way the effect of using two models in each algorithm, Figure 3.2 shows the result for the `Tic-tac-toe` dataset. The boxplots show the execution of 96 test cases where the algorithm tried to calculate the counterfactual, where the line in the plot is the median. The results are similar to those seen in Table 3.4, although it is worth paying attention to the MOEDA improvement in distance where its results are close to those seen with the single-objective EDA, taking into account that the worsening of the precision is also observed. It can be seen that switching to another model does not improve plausibility.

3.2.5 Results with all models

By having two models it is possible to obtain an improvement in distance without significantly worsening the rest of the objectives, so it is worth checking what happens if all five Bayesian classifiers available are used. To check this we will analyse how the different algorithms compare with the best model, two best models and all models.



(a) Best model



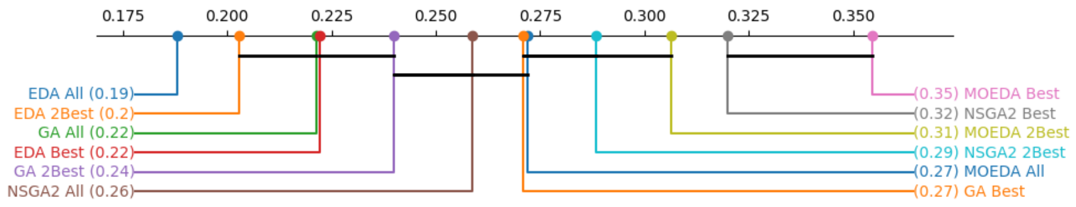
(b) Two best models

Figure 3.2: Results from all executions in the `Tic-tac-toe` dataset using the best accuracy model (*a*) and the two best models (*b*)

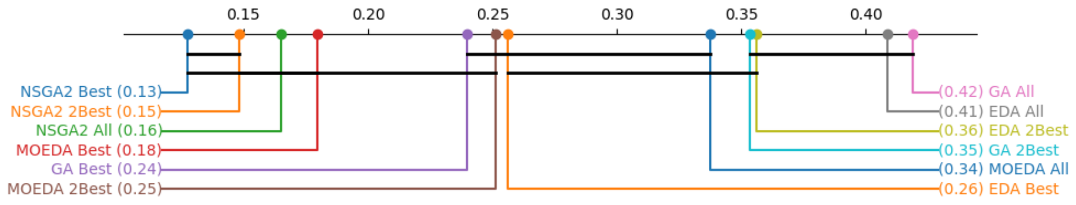
Figure 3.3 shows with critical difference diagrams (Demšar (2006)) the comparison in distance, prediction and plausibility average over all datasets. The values on the axis indicate the average objective value obtained by the corresponding algorithm and models used over all the datasets, where smaller the better. The lines linking different results mean that they do not show a statistically significant difference, calculated using the Friedman test followed by the post-hoc Nemenyi test. Starting with the distance, Figure 3.3a, it is possible to see how the best results are obtained by the single-objective algorithms, where the best is the EDA with all the models. The multi-objective versions come after, alternating between NSGA2 and MOEDA. In the case of prediction and plausibility, as expected, the multi-objective versions are ahead of the single-objective ones, and in these objectives not always having all the models improves the results. In prediction, Figure 3.3b, NSGA2 obtains the best results followed by MOEDA with the best model, while in plausibility, Figure 3.3c, the gap between NSGA2 and MOEDA is more remarkable, but all results are really close for this objective. Note that in all objectives the version with the two best models is better or it is close to the results obtained by all models, so adding these models does not provide significant improvement given that they add execution time. Looking at the significant differences in all critical difference diagrams, in general two cases occur, the first one is that methods using the same number of models do not show a statistically significant difference between them, and on the other hand the basic genetic algorithm does not differ in some cases from MOEDA.

3.2.6 Counterfactual example

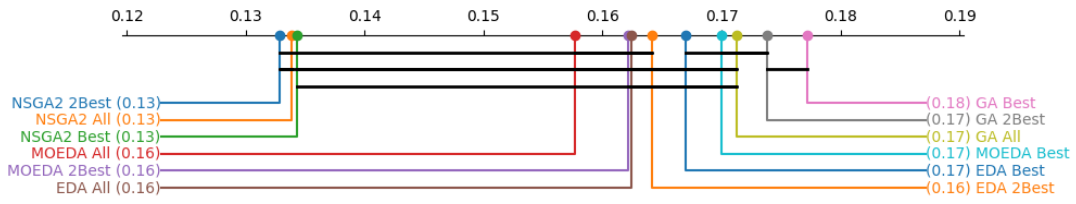
This section includes an example of a counterfactual explanation computation to better understand how the different algorithms work. The dataset used is `Car evaluation` (Table 3.1), which consists of a dataset of car specifications and the output is how ac-



(a) Distance comparison



(b) Prediction comparison



(c) Plausibility comparison

Figure 3.3: Critical difference diagrams of distance, prediction and plausibility

ceptable the car is, based on the specifications (unacceptable, acceptable, good, very good). The features consist of the price of the car, maintenance cost, number of doors, passenger capacity, boot capacity and safety. We start from the initial instance that appears at the beginning of Table 3.5, which correct prediction is unacceptable, and we search for the changes necessary to obtain acceptable as a prediction. In Table 3.5 the result with the best model is observed, in this particular case the K-dependence Bayesian classifier (with k=2). The features that have not been modified in each algorithm are marked with a hyphen. The results show that a change in maintenance and safety are essential to change the prediction output. Moreover, some models obtain results with a smaller distance than others, the best ones being EDA and NSGA2 since their distance is the smallest. Also, models with higher distance have more feature changes, so they are worse solutions although having similar prediction and plausibility values. On the other hand, in Table 3.6 the same counterfactuals can be seen but using the results from all models, showing in the first column which model(s) obtain the best solution. Note that there is no consistency between models, since all models appear except the semi-naive Bayes, due to the fact that there are few features. In this case it can be seen how practically all the algorithms obtain the same result, i.e., maintenance should be reduced to medium and safety to high.

3.2.7 Counterfactuals in a real dataset

To show the potential of these methods and how to apply them, we are going to use a real dataset. We will approach the problem of classifying GABAergic interneurons according to their morphology. In DeFelipe et al. (2013) 48 leading neuroscientists were asked to classify 320 interneurons by inspecting images of their morphology, being

Table 3.5: Counterfactual explanation example obtained using the best model

Algorithm	price	maint	doors	persons	lugboot	safety	dist	prec	plau
initial	high	vhigh	5more	more	small	low	-	-	-
EDA (KDB)	-	med	-	-	-	high	0.14	0.24	0.00
GA (KDB)	-	med	-	-	big	med	0.22	0.23	0.00
MOEDA (KDB)	-	med	-	-	med	med	0.30	0.30	0.00
NSGA2 (KDB)	-	med	-	-	-	high	0.14	0.24	0.00

Table 3.6: Counterfactual explanation example obtained using all models

Algorithm	price	maint	doors	persons	lugboot	safety	dist	prec	plau
initial	high	vhigh	5more	more	small	low	-	-	-
EDA (KDB,TAN)	-	med	-	-	-	high	0.14	0.24	0.00
GA (TAN-HC)	med	-	-	-	-	high	0.14	0.22	0.05
MOEDA (NB)	-	med	-	-	-	high	0.14	0.24	0.00
NSGA2 (KDB)	-	med	-	-	-	high	0.14	0.24	0.00

the first work to look for a common classification of these, obtaining an agreement in some types and disagreement in others among neuroscientists.

The data used in the work of DeFelipe et al. (2013) were globally available and labeled by Mihaljević et al. (2019). In addition to morphological features, some high-level morphological features, named F1, F2, F3 and F4, are defined. They have the following categories: (F1) intralaminar and translaminar; (F2) intracolumnar and transcolumnar; (F3) centered and displaced; (F4) ascending, descending and both. The class will be the type of interneuron, where eight different types are distinguished, shown in Figure 3.4. The data also contains an additional feature that can be characterized and uncharacterized, where the uncharacterized category means that the reconstruction of a cell is not good enough to reliably classify it, but we will not use it because we are going to use a single neuroscientist who has all neurons as characterized and fully labeled, where the neuroscientist used is the third one. We will use the labels provided by this neuroscientist to train the classifiers, using 90% of the data for training and the rest for testing.

We will show several examples on test data, using MOEDA with the two best models (in this case SNB and TAN models). Applying counterfactuals on the interneurons can allow us to detect what differentiates one type from another, by looking at the key features of each type. In Table 3.7 we can see the change from common basket to large basket, showing that the difference is in F1 and F2. On the other hand, in Table 3.8 we can see the change from horse-tail to Martinotti, where their similarity can be seen visually in Figure 3.4, and as the only necessary change is the F4 changing descending by ascending.

Due to the fact that this is a small dataset and some types of interneurons have almost no instances, see Figure 3.5, the counterfactuals involving these classes will be less accurate and less plausible, even not being possible to compute them because the models do not correctly classify the input instances. This can be seen in Table 3.9, where going from Martinotti to arcade the algorithm does not find a very accurate or

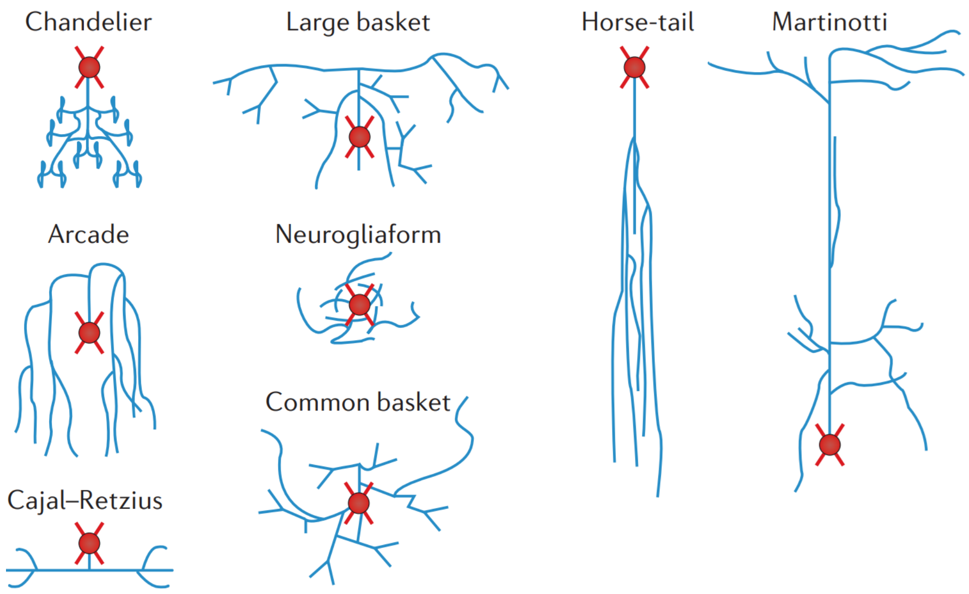


Figure 3.4: Neuron types

Table 3.7: Counterfactual from a common basket to a large basket type

Algorithm	F1	F2	F3	F4	class	dist	prec	plau
initial	intrasl	intrac	center	None	common basket	-	-	-
MOEDA	transr	transc	-	-	large basket	0.5	0.22	0.00

Table 3.8: Counterfactual from a horse-tail to a Martinotti type

Algorithm	F1	F2	F3	F4	class	dist	prec	plau
initial	transr	intrac	displ	descend	horse-tail	-	-	-
MOEDA	-	-	-	ascend	Martinotti	0.25	0.19	0.00

plausible explanation, since the distances tried to be minimized, although this does not mean that it is not correct, as it would need more data to see its plausibility.

Figure 3.5: Number of instances of each neuron type

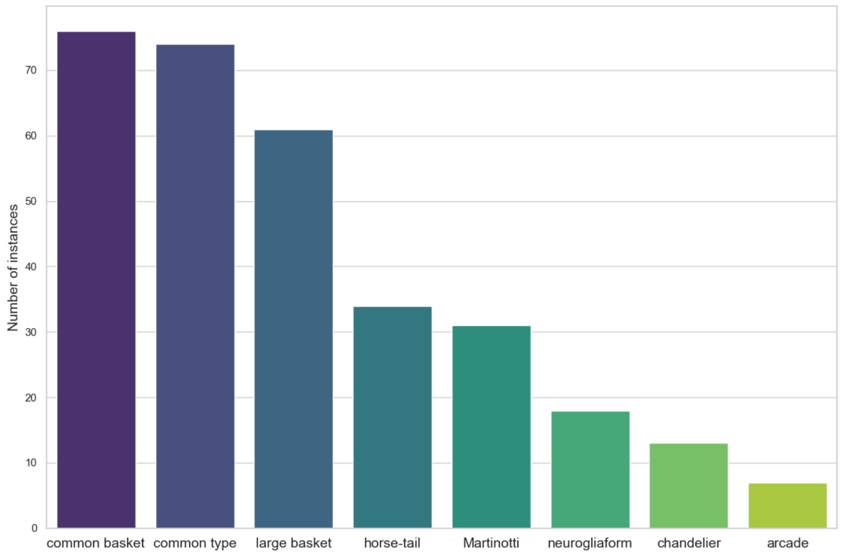


Table 3.9: Counterfactual from a Martinotti to an arcade type

Algorithm	F1	F2	F3	F4	class	dist	prec	plau
initial	transr	intrac	displ	ascend	Martinotti	-	-	-
MOEDA	-	transc	None	None	Arcade	0.75	0.59	0.20

With these examples we can see how MODEA would be applied to a real dataset, and how good and consistent results are obtained. The results will vary depending on the classes used, since the method needs examples to be as accurate as possible, but it can be applied to learn the differences of the classes and to reinforce previous knowledge.

3.3 Conclusion

We have explored counterfactual explanations using estimation of distribution algorithms with Bayesian classifiers. Our work involved comparing the performance of single-objective and multi-objective solutions against their genetic algorithm counterparts. In terms of the number of models, the best configuration in the experiments is to use the two best models, regardless of the algorithm employed. The NSGA2 algorithm consistently yielded the best results overall when considering metrics such as distance, prediction, and plausibility. However, if minimizing execution time is crucial, a single-objective EDA is advisable despite producing slightly worse results. The multi-objective EDA performs exceptionally well with datasets that have numerous features, often matching or surpassing NSGA2 in prediction accuracy. Also, we have observed how MOEDA behaves in a small real dataset, seeing how its results can be useful to observe differences between classes and what problems can arise with small datasets.

Chapter 4

Most Relevant Explanation

4.1 Proposal

The solution space of most relevant explanation (MRE) is very large and therefore to find it is a very computationally expensive task as the number of nodes and states of the BN grows. Yuan et al. (2011a) showed that MRE is at least *NP*-hard and thus not feasible by a brute-force algorithm, as can be seen in Figure 4.1. Some optimization algorithms and heuristics have been applied in the literature to solve this problem. Yuan et al. (2011a) proposed to use reversible jump Markov Chain Monte Carlo (MCMC) and compare it with other solutions such as local search, tabu search or simulated annealing. Later, Zhu and Yuan (2017) proposed an algorithm based on hierarchical beam search, improving the results obtained with other previously tested algorithms.

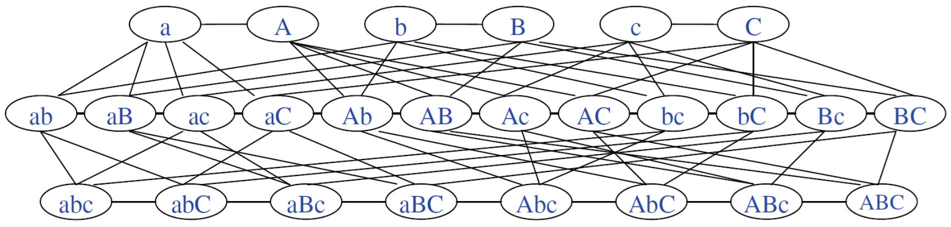


Figure 4.1: MRE solution space for three nodes, where each one is binary (the nodes are {a,A},{b,B} and {c,C}) (Yuan et al. (2011a))

Our proposal is to use EDAs to solve MRE. For this we will use two different EDAs, UMDA and EBNA, both in their discrete version. The algorithm will follow the basic structure of an EDA presented in Section 2.3, creating a cost function that calculates the GBF of the current solutions. Each individual will consist of a vector, where each position corresponds to a node and its values are the possible states adding a value indicating that this node is not included in the explanation. The process that the MRE will follow with EDAs can be seen in Algorithm 2.

In addition to modifying the cost function, other modifications are proposed to improve the solutions found and the speed of the algorithms. The first one is not to initialize completely randomly the initial population, but rather to initialize the algorithm with short solutions (line 2). This change aims to speed up the search, since the GBF eliminates redundant variables in the explanations and therefore MRE will find the best solutions with a small number of variables. The second improvement has to do with the solutions with a single variable, since Zhu and Yuan (2017) found that quite often the best solution can be found among those with a single variable. For this reason, a variant of EDA is made that takes into account these solutions together with the one found by the algorithm (lines 1 and 9).

Algorithm 2 EDA for MRE with all modifications

```
   Input: Evidence nodes, target nodes
   Output: Most relevant explanation
1: Calculate one node solutions
2: Pseudo-random initial population
3: for $t = 1, 2, \ldots$ until stopping criterion is met do
4:     Evaluate each individual of the population using the GBF
5:     Select individuals
6:     Learn a probabilistic model from the selected individuals
7:     Sample new individuals from the probabilistic model
8: end for
9: Compare best solution with one variable solutions
```

MRE works with three sets of nodes: evidence, target and neutral (see Equation 2.7). Evidence nodes are the nodes **E** instantiated to a value in the BN and are used in the GBF calculation. Target nodes are the nodes **M** that are part of the search space and must be selected by the user, either to relevant nodes (nodes for which an explanation is sought) or all nodes. The neutral nodes correspond to all other nodes and are only taken into account by the BN. When the number of target nodes is very large, the inference in the network will be very slow and therefore it will not be feasible to compute it many times during the MRE computation. For this reason, a threshold has been set for the number of nodes to be used for explanations even if there are more target nodes. Where to put this limit will depend on the machine being used and we will try to keep it as high as possible. This limitation with large targets does not significantly affect the results obtained, since the explanations in general will not have as many nodes. When you want to calculate the GBF with a greater number of nodes than possible, two ways of selecting nodes are proposed. The first one is to randomly select nodes until the maximum limit of nodes is reached, while the second alternative is to select the nodes that have higher GBF separately. These proposals will allow the target to be larger and at the same time feasible to be solved by MRE.

4.2 Experiments

4.2.1 Implementation and algorithms

The implementation of the different algorithms is done in Python using different specific libraries for each algorithm. For EDAs we have used EDAspy (Soloviev et al. (2024)), genetic algorithms with Pymoo (Blank and Deb (2020)) and inspyred (Tonda (2020)) for different algorithms like differential evolution algorithm (DEA) or particle swarm optimization (PSO). All experiments were conducted on the same hardware (Intel Xeon Gold 6230 and 128GB RAM).

Algorithms based on EDAs have been implemented using UMDA and EBNA, where the former has a variant that adds the modifications described in Section 4.1 (in the results it is named as UMDA2). There are two implementations of genetic algorithms, a basic implementation and one based on NSGA2 (Deb et al. (2000)) with a single objective. In addition to these algorithms, we implemented tabu search, evolution strategy (ES), differential evolution algorithm (DEA) (Storn (1995)), particle swarm optimization (PSO) (Kennedy and Eberhart (1995)) and hierarchical beam search (HBS) (Zhu and Yuan (2017)). These algorithms have been selected because they have been used previously to solve the MRE problem or because they are relevant heuristic algorithms that will allow to obtain more complete results. In the results all algorithms use the improved initialization explained in Section 4.1, although some algorithms have one version that does not use it and another that does (it will be denoted as "-i", i.e. EBNA-i).

4.2.2 Bayesian networks used

Different BNs have been used for the experiments, with different numbers of nodes and parameters. In Table 4.1 the networks used can be seen, which have been obtained from the bnlearn repository (Scutari (2010)). For each BN, all leaf nodes are used as evidence and from the remaining nodes the targets are chosen randomly. Two target sizes will be evaluated, a small one of 12 nodes and a large one of 20 nodes. Small networks will not be used with 20 target nodes. For each algorithm, 200 test cases will be generated, where the evidence will be instantiated by simulating instances from the BN and the target nodes will be chosen randomly.

Table 4.1: BNs used for the experiments

Networks	Nodes	Arcs	Parameters
`Child`	20	25	230
`Insurance`	27	52	1008
`Alarm`	37	46	509
`Hepar2`	70	123	1453
`Win95pts`	76	112	574

4.2.3 Twelve target nodes

In this section we use a target size of 12 nodes, being this a standard configuration of target nodes without having a limitation of resources and time. First we will compare the average position of the algorithms with respect to the rest of algorithms for all the BNs used. In Figure 4.2 we can see how the best results are obtained with the DEA algorithm, closely followed by PSO. HBS and UMDA2 are quite close, and DEA and UMDA2 are connected and therefore their results cannot be distinguished statistically. It can also be seen that the algorithms with improved initialization perform better than the randomly initialized algorithms and that the UMDA2 algorithm improves the results of basic UMDA. There is a difference between the algorithms depending on the datasets, observing that in datasets with many nodes and many parameters the algorithms such as DEA or PSO obtain better results. On the other hand, UMDA-based algorithms improve these for smaller networks or with a smaller number of parameters.

Considering how many times the algorithms find the best solution, UMDA2 obtains the best results 157 times out of 200 on average. This result is an improvement of 8% over DEA and 10% over PSO, these being the next algorithms that find the best solutions more often.

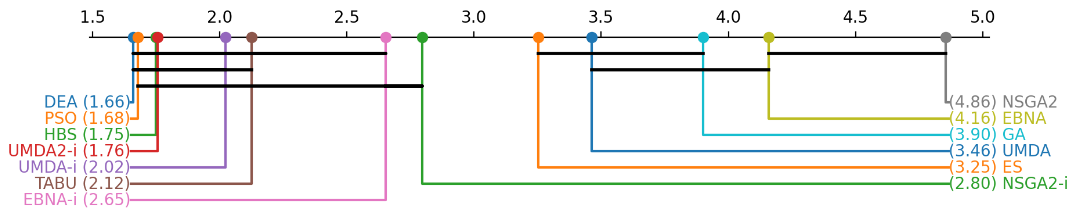


Figure 4.2: Critical difference diagram for all algorithms in the 12 target setting

Another important detail to take into account is execution time. Figure 4.3 shows the execution time for all BNs. It can be seen how the UMDA-based algorithms obtain the lowest and most stable times, with times concentrated in the same interval. Behind these are the NSGA2 and Tabu algorithms with times not very far from the UMDA ones. It should be noted that the algorithms that obtain the best ranking results are the algorithms that take the longest to execute. DEA and PSO take on average four times as long as the improved version of UMDA, which is even longer than the basic version. It is also worth noting that the basic version of UMDA takes on average 40% less time than UMDA2, being a good choice if you are looking for speed and good results.

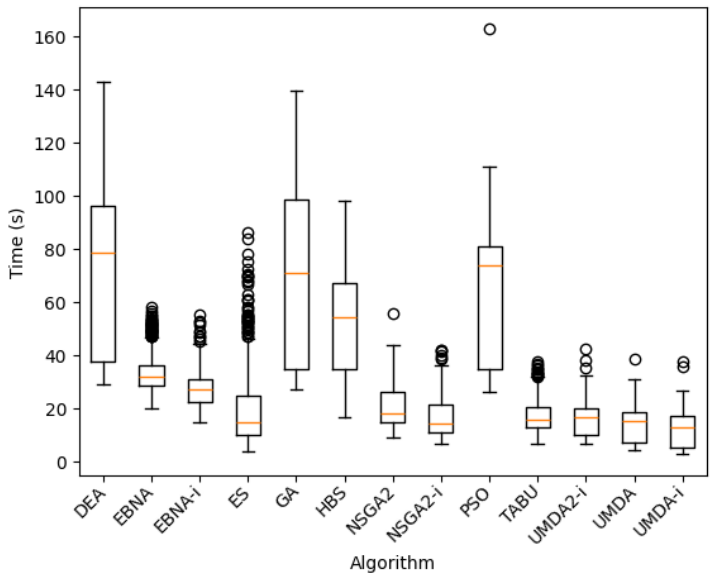


Figure 4.3: Time comparison of all algorithms in the 12 target setting

4.2.4 Twenty target nodes

When the number of targets grows, the inference time and the resources needed for its computation grow as well and thus it is not feasible to compute the MRE with large target sets. Therefore, an adaptation has been made to support any number of nodes, as explained in Section 4.1. Two ways of selecting nodes when the algorithm encounters too many for an explanation are described, corresponding to selecting the best ones individually (algorithms marked with "-b") or selecting randomly (algorithms marked with "-r").

Figure 4.4 shows the critical difference diagram taking into account all BNs and algorithms. When having 20 target nodes the PSO and DEA algorithms obtain a greater difference with respect to the others than in the cases with fewer nodes in the target, although these algorithms do not show significant statistical differences with the algorithms based on EDAs. It is also remarkable how Tabu and HBS obtain a better ranking than the UMDA-based algorithms. In terms of best solutions found, UMDA2 is close to or equal to PSO, DEA or HBS, although it falls further behind them when it does not find the best solution. Regarding the two variants proposed for selecting nodes, in general, it can be seen that the random version obtains better results than the one based on selecting the best ones.

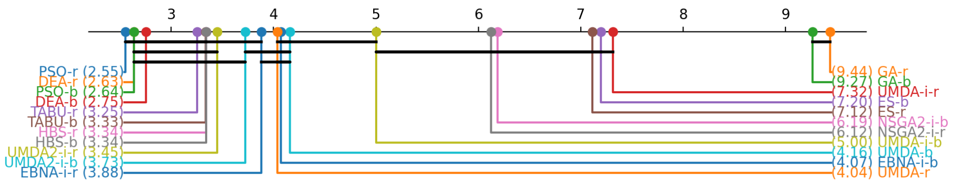


Figure 4.4: Critical difference diagram for all algorithms in the 20 target setting

As the number of nodes increases, the search space and the complexity increase, and therefore the times increase, as can be seen in Figure 4.5. It can be seen how the times of many algorithms grow, especially how some executions can take much longer, where DEA, NSGA2 and PSO stand out. It is observed that UMDA algorithms with improved initialization take much less time than the others and achieve a lower variability, the latter is also obtained in genetic algorithms. Taking the average times in the different datasets, it can be seen that UMDA2 takes on average between 7 and 13 times less time than PSO or DEA.

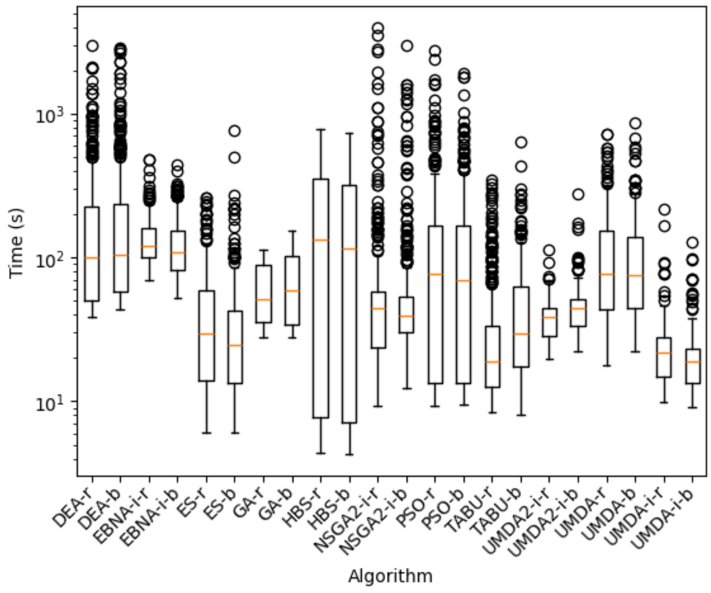


Figure 4.5: Time comparison of all algorithms in the 20 target setting

4.2.5 Brute force comparison

To check how good are the solutions found by the different algorithms we are going to compare them with the brute force algorithm. As this algorithm is computationally very expensive, these tests are performed on a six node target setting and on the BN `Alarm`. Figure 4.6 shows the ranking of the algorithms with respect to the brute force (BF) algorithm. The algorithm that comes closest is UMDA2, since it only fails to find the best solution in one case. Close behind are the HBS, UMDA, DEA and Tabu algorithms which find almost all the best solutions. The worst algorithm of all is NSGA2, although not statistically different from the genetic algorithm, which in turn is not statistically different from the solutions found by the brute-force algorithm.

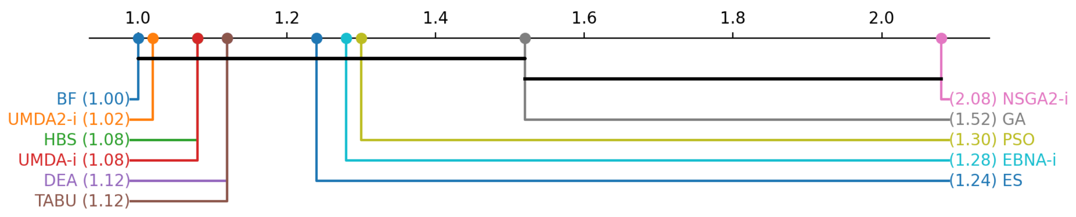


Figure 4.6: Critical difference diagram for the BN `Alarm` with a 6 target setting

As for the times, it can be seen in Figure 4.7 how the GA, DEA and PSO algorithms take the same time or more than the brute-force algorithm. On the other hand, the fastest algorithms are UMDA, HBS and Tabu, finding the solutions about 10 times faster than the brute-force algorithm. In the case of UMDA2, the added time it takes to compute the individual scenarios is noticeable, although it is still significantly faster than others that find worse solutions.

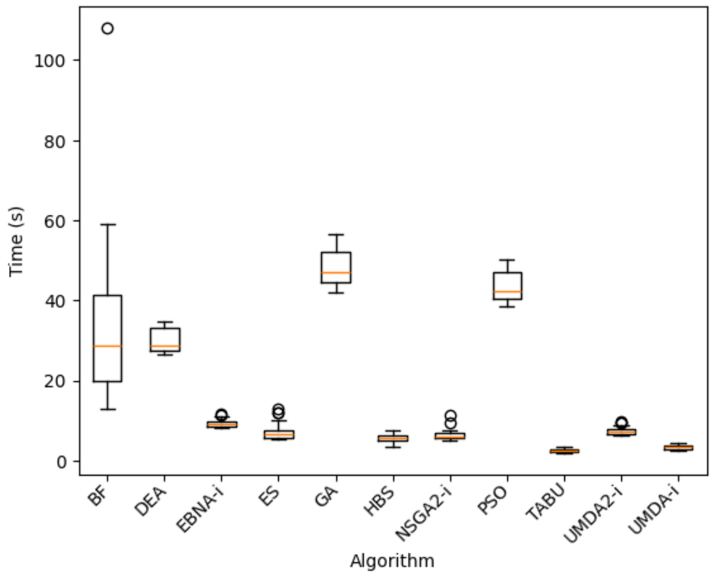


Figure 4.7: Time comparison of all algorithms in the 6 target setting for the `Alarm` network

4.2.6 Execution example

Having seen how the different algorithms compare, we are going to see how they behave with a concrete example and how the GBF varies from one method to another. For this we have used the BN `Alarm`, see Table 4.1 for more information about the network. All leaf nodes have been used as evidence, in this particular case eleven nodes and as target nodes, twelve nodes have been randomly selected from the remaining ones. The specific nodes used in both evidence and target are listed in Appendix A. In Table 4.2 the results with the different algorithms can be seen, observing the length of the solutions (number of variables, out of 12), GBF obtained and execution time. It can be seen that there is a lot of variability in the lengths of the solutions found, although the GBF has more variability. HBS finds the best solution, followed by Tabu, PSO and UMDA2, where all but HBS are seven or eight in length, while the HBS solution is longer. In terms of times UMDA2, ES and NSGA2 are the fastest, while on the other hand DEA, GA, HBS and PSO are the slowest. Although the algorithms find different solutions, some variables remain the same among the solutions and therefore will be important variables to explain the evidence. The complete solutions can be seen in Appendix A.

Table 4.2: Solutions found by each algorithm for the MRE example

Algorithm	Solution Length	GBF	Time
UMDA2-i	7	162.99	8.38
DEA	1	77.95	21.81
EBNA	7	132.55	16.56
ES	6	50.11	6.95
GA	8	137.06	28.03
HBS	11	181.79	24.86
NSGA2	7	151.93	8.92
PSO	8	169.61	27.53
TABU	8	175.70	10.14

4.3 Conclusion

We have proposed a new algorithm for most relevant explanation based on estimation of distribution algorithms. Different variants of the algorithm have been proposed and other heuristic algorithms have been implemented. We have compared the existing algorithms to solve the MRE with all the implemented ones, observing the results with different target sizes and different Bayesian networks. It has been observed that algorithms such as PSO, HBS, DEA or the proposed UMDA obtain the best results, although the algorithm based on EDAs runs much faster, taking 4 to 13 times less time.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this work we have explored explainability in BNs and Bayesian classifiers, using algorithms based on EDAs.

We have approached counterfactual explanations with EDAs using Bayesian classifiers. We compared our single-objective and multi-objective solutions with the genetic algorithms counterparts. Regarding the number of models the best configuration in the experiments is to use the two best models regardless of which algorithm is being used and the NSGA2 algorithms obtain the best results on average taking into account distance, prediction and plausibility. However, if the priority is execution time, it is recommendable to use a single-objective EDA, even if the results are slightly worse. MOEDA obtains very good results when dealing with datasets with many features, with results equal or superior to NSGA2, especially in prediction. Due to this, and the fact that it is slightly faster than NSGA2, in datasets with many features you can take advantage of its use.

For the most relevant explanation problem we have proposed a new algorithm based on EDAs. Different variants of the algorithm have been proposed and other heuristic algorithms have been implemented. We have compared the existing algorithms to solve the MRE with all the implemented ones, observing the results with different target sizes and different BNs. It has been observed that algorithms such as PSO, HBS, DEA or the proposed UMDA obtain the best results, although the algorithm based on EDAs runs much faster, taking 4 to 13 times less time.

5.2 Future work

As for future work, different ideas have been proposed and discussed as potential future upgrades in the algorithms or alternative methods. In this section we will present some ideas for improving counterfactuals and most relevant explanations.

As for counterfactual explanations, it would be interesting to see how to improve MOEDA to better deal with cases with few variables. We could also use other types of classification models or use continuous predictor variables. In addition, an alternative could be searched for when none of the classifiers is able to find a solution, either because no model is able to predict the input correctly or because the algorithms do not find a solution of the desired class.

Regarding MRE, it would be interesting to explore and develop improvements to be able to use larger target sets without requiring large hardware resources and time. On the other hand, improvements can be sought to allow EDA to find better solutions that outperform slower algorithms, all this while minimizing the execution time.

5.3 Scientific dissemination

The counterfactual explanation work was accepted and peer reviewed in the conference Probabilistic Graphical Models 2024 (`https://www.ru.nl/en/about-us/events/probabilistic-graphical-models-pgm-2024`) (Zaragoza et al. (2024)).

Bibliography

Umberto Bertele and Francesco Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

Concha Bielza and Pedro Larrañaga. Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1):1–43, 2014.

J. Blank and K. Deb. PYMOO: Multi-objective optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

Gregory F Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

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

Rónán Daly, Qiang Shen, and Stuart Aitken. Learning Bayesian networks: Approaches and issues. *The Knowledge Engineering Review*, 26(2):99–157, 2011.

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pages 448–469. Springer, 2020.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceeding of the 6th International Conference of Parallel Problem Solving from Nature VI*, pages 849–858. Springer, 2000.

Javier DeFelipe, Pedro L López-Cruz, Ruth Benavides-Piccione, Concha Bielza, Pedro Larrañaga, Stewart Anderson, Andreas Burkhalter, Bruno Cauli, Alfonso Fairén, Dirk Feldmeyer, et al. New insights into the classification and nomenclature of cortical gabaergic interneurons. *Nature Reviews Neuroscience*, 14(3):202–216, 2013.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7(1):1–30, 2006.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, volume 31, page 590–601, 2018.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable AI (xAI): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.

Ramon Etxeberria and Pedro Larrañaga. Global optimization using Bayesian networks. In *Proceedings of the 2nd Symposium on Artificial Intelligence*, 1999.

Ronald A Fisher. The logic of inductive inference. *Journal of the Royal Statistical Society*, 98(1):39–82, 1935.

John C Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971.

Riccardo Guidotti. Counterfactual explanations and how to find them: Literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55, 2022.

Mark Hauschild and Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, 2011.

David Heckerman, John S Breese, and Koos Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, 1995.

Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. Explainable AI methods - a brief overview. In *International Workshop on Extending Explainable AI beyond Deep Models and Classifiers*, pages 13–38. Springer, 2022.

H Jeffreys. Theory of probability. *Oxford University Press*, 1961.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. UCI machine learning repository, 2023. URL `http://archive.ics.uci.edu/ml`.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

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

Carmen Lacave and Francisco J Díez. A review of explanation methods for Bayesian networks. *The Knowledge Engineering Review*, 17(2):107–127, 2002.

Pedro Larrañaga and Jose A Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* Kluwer Academic Publisher, 2002.

Michael T Lash, Qihang Lin, Nick Street, Jennifer G Robinson, and Jeffrey Ohlmann. Generalized inverse classification. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 162–170. SIAM, 2017.

Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B*, 50(2):157–194, 1988.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable AI: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

Ana Lucic, Hinda Haned, and Maarten de Rijke. Why does my model fail? Contrastive local explanations for retail forecasting. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 90–98, 2020.

Bojan Mihaljević, Ruth Benavides-Piccione, Concha Bielza, Pedro Larrañaga, and Javier DeFelipe. Classification of gabaergic interneurons by leading neuroscientists. *Scientific Data*, 6(1):221, 2019.

Bojan Mihaljević, Concha Bielza, and Pedro Larrañaga. bnclassify: Learning Bayesian network classifiers. *The R Journal*, 10(2):455–468, 2018.

Tim Miller, Piers Howe, and Liz Sonenberg. Explainable AI: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547*, 2017.

Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2nd edition, 2022.

Jonathan Moore, Nils Hammerla, and Chris Watkins. Explaining deep learning models with constrained adversarial examples. In *Procedings of the 16th Pacific Rim International Conference on Artificial Intelligence Part I 16*, pages 43–56. Springer, 2019.

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617, 2020.

Heinz Mühlenbein and Gerhard Paass. From recombination of genes to the estimation of distributions I. Binary parameters. In *International Conference on Parallel Problem Solving from Nature*, pages 178–187. Springer, 1996.

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

Marco Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

Andrew D Selbst and Solon Barocas. The intuitive appeal of explainable machines. *Fordham Law Review*, 87:1085, 2018.

Ross D Shachter and Mark A Peot. Simulation approaches to general probabilistic inference on belief networks. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 221–231. Elsevier, 1990.

Vicente P Soloviev, Pedro Larrañaga, and Concha Bielza. Edaspy: An extensible python package for estimation of distribution algorithms. *Neurocomputing*, page 128043, 2024.

Rainer Storn. Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical report, International Computer Science Institute*, 11, 1995.

Alberto Tonda. Inspyred: Bio-inspired algorithms in python. *Genetic Programming and Evolvable Machines*, 21(1):269–272, 2020.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2): 49–60, 2014.

Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2, 2020.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31:841, 2017.

Jeremy York. Use of the Gibbs sampler in expert systems. *Artificial Intelligence*, 56 (1):115–130, 1992.

Changhe Yuan, Heejin Lim, and Michael L Littman. Most relevant explanation: Computational complexity and approximation methods. *Annals of Mathematics and Artificial Intelligence*, 61(3):159–183, 2011a.

Changhe Yuan, Heejin Lim, and T Lu. Most relevant explanation in Bayesian networks. *Journal of Artificial Intelligence Research*, 42:309–352, 2011b.

Daniel Zaragoza, Concha Bielza, and Pedro Larranaga. Multi-objective counterfactuals in Bayesian classifiers with estimation of distribution algorithms. In *International Conference on Probabilistic Graphical Models*, page TBD. PMLR, 2024.

Nevin L Zhang and David Poole. A simple approach to Bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, 1994.

Xiaoyuan Zhu and Changhe Yuan. Hierarchical beam search for solving most relevant explanation in Bayesian networks. *Journal of Applied Logic*, 22:3–13, 2017.

Appendix

A MRE example complete results

In this Appendix the concrete details about the MRE example are shown, presenting the complete evidence, the target nodes and the complete results of the algorithms. Table 1 shows the evidence nodes used, Table 2 shows the target nodes and Table 3 shows the complete results obtained by each algorithm.

Nodes	history	cvp	pcwp	hrbp	hrekg	hrsat	expco2	minvol	pap	press	bp
State	false	normal	normal	low	low	low	low	ZERO	normal	zero	low

Table 1: Evidence nodes and their states

Nodes	sao2	ventm	strokevol	tpr	hypovol	co	discon	anaphy	lvedvol	shunt	insuffa	intuba

Table 2: Target nodes used

Alg	sao2	ventm	strokevol	tpr	hypovol	co	discon	anaphy	lvedvol	shunt	insuffa	intuba
UMDA2	low	low	normal	low		normal			normal	normal		
DEA									normal			
EBNA	low	zero	normal		false	normal		true				normal
ES		low	normal		false	normal		true	normal			
GA	low				false	normal	true	true	normal	normal		normal
HBS	low	normal	normal	low	false	normal	true	false	normal	normal		normal
NSGA2	low	low		low	false	normal			normal	normal		
PSO	low	normal	normal	low		normal	true		normal	normal		
TABU	low	low	normal	low		normal			normal	normal		normal

Table 3: Results for each algorithm