Bayesian Network Structure Learning

João Miguel Coelho de Oliveira Ferreira da Trindade
joao.o.f.trindade@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October, 2021

Abstract

Causal reasoning is a fundamental part of human intelligence and can be found across a broad range of fields, from ancient philosophy and theology to medicine and economics. Bayesian networks are probabilistic graphical models that can represent cause-effect relationships and have been used to build system models ranging from medical diagnosis to weather prediction. However, traditional approaches for building Bayesian networks oftentimes prove to be too costly, unethical or impossible. Therefore, there is a need for algorithms that can learn Bayesian networks from observational data alone, which is not trivial due to the combinatorial nature of the search space. Recently, a novel algorithm, *NOTEARS*, has established a new approach, reformulating this problem as continuous optimization, which allows the use of well-studied techniques of machine learning.

This work presents a new meta-algorithm that combines *NOTEARS* with a state-of-the-art traditional algorithm, *FGES*, providing a greater degree of certainty when identifying and interpreting specific relationships encoded on the learned Bayesian networks, even when we do not possess expert knowledge on the field of the observed data. We test the new meta-algorithm on well-known Bayesian networks, showing that it identifies specific relationships with greater precision than the individual algorithms. We also apply it to publicly available data sets and provide a method to evaluate the obtained results when there is no ground truth. In the conducted experiments, the meta-algorithm shows competitive results with the aforementioned algorithms, consistently outperforming *NOTEARS* and, in certain instances, *FGES*.

Keywords: Bayesian networks, Bayesian network parameter learning, Bayesian network structure learning, continuous optimization, causal reasoning

1. Introduction

Causal reasoning is an integral part of human intelligence, evident from its application throughout time and across a broad spectrum of areas of knowledge. Using it we are able to understand the past, *causes*, and predict the future, *effects*.

Bayesian networks are probabilistic graphical models particularly well-suited to describe cause-effect relationships. On the one hand, their use of probability theory makes them especially adept for modeling stochastic systems, having been applied to climate prediction [1], medicine [2], biological sciences [3, 4], assessing economic trends [5], social modeling [6], and decision making [7]. While on the other hand, their graphical representation provides a simple way to visualize the structure of a model. This can produce valuable insights into the properties of the system it is modeling.

A traditional approach for building a Bayesian network is to conduct randomized experiments, where we intervene in the system and analyze the effects of our intervention on the measurement data. Then, in collaboration with an expert on the field of the system that we are contemplating, we analyze the results of our interventions and build the network accordingly.

However, this approach generally tends to be either too expensive, due to the running of significant experiments and using an expert's valuable time, or simply impossible, when the experiments are unfeasible or if there is no one with expertise on the subject matter. Therefore, the need for an automated strategy based purely on observational data led to the development of alternative approaches. In the field of probabilistic graphical modeling, this is known as network structure learning [8].

These new methods essentially search for the network structure that best fits the observed data, which is not trivial due to the combinatorial nature of the search space. They make use of clever heuristics and strong assumptions in order to reduce the amount of possible structures to be considered and yet most become intractable for moderately large networks [9].

A recently proposed algorithm, *NOTEARS*, established a new approach, reformulating the search problem as continuous optimization [10]. This allows us to use well-established machine learning methods, while requiring fewer assumptions on the data.

However, neither *NOTEARS* nor the search methods guarantee that the network structure they obtain is the one that best fits the data. Both types of approach are subjected to the pitfalls of non-convex optimization, converging on local optima solutions. Therefore, we should always be skeptical about the quality of the learned network structures [9, 11].

1.1. Contributions

The main contributions of this work are:

- The formulation of the hypothesis that there are certain relationships so strongly encoded in the data, that they are common to the networks learned by different Bayesian network structure learning algorithms.
- Demonstrating on the data generated from the well-known Bayesian networks that these common relationships are more likely to be in the original Bayesian network than the edges obtained by the individual algorithms.
- Based on the demonstrated validity of the hypothesis, the development of a new meta-algorithm that combines the continuous optimization approach algorithm, *NOTEARS*, with the state-of-the-art search-based approach algorithm, *FGES*.
- Providing a method with which it is possible to evaluate and compare the performance of the various Bayesian network structure learning algorithms when there is no ground truth Bayesian network.

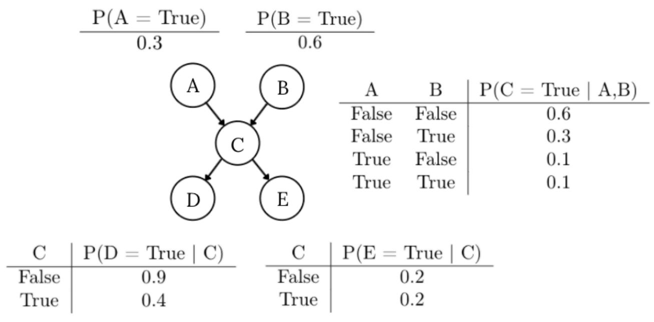
2. Background and Related Work

2.1. Bayesian networks

A Bayesian network encodes a joint probability distribution P over a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. Formally, a Bayesian network B is a pair $\{G, \boldsymbol{\Omega}\}$, where G is a directed acyclic graph (DAG) in which each node corresponds to one of the random variables, and $\boldsymbol{\Omega}$ specifies the set of conditional probability distributions $P(X_i|\text{Pa}_{X_i})$ for each X_i. The edges or lack of them encode the conditional independence relationships among the variables, with each node X_i being independent of its non-descendant variables given its parents, Pa_{X_i}. Thus, the joint probability distribution of all of the variables is given as

$$P(\boldsymbol{X}) = \prod_{i=1}^{n} P(X_i|\text{Pa}_{Xi}) \,, \qquad (1)$$

where the individual factors $P(X_i \mid \text{Pa}_{X_i})$ are called the *conditional probability distributions (CPDs)*. This equation is known as the chain rule for Bayesian networks [8].



P(A = True)
0.3

P(B = True)
0.6

A	B	P(C = True | A,B)
False	False	0.6
False	True	0.3
True	False	0.1
True	True	0.1

C	P(D = True | C)
False	0.9
True	0.4

C	P(E = True | C)
False	0.2
True	0.2

Figure 1: Example of a Bayesian network composed of boolean variables.

Bayesian networks are of particular interest since:

- They are graphical models, therefore capable of displaying relationships clearly and intuitively, see an example in figure 1.
- They are comprised of directed edges, which means that they can represent cause-effect relationships.
- They can handle uncertainty, which is pervasive in most AI application domains, through the use of probability theory.
- They can be used to represent indirect in addition to direct causality.

There are two particularly relevant concepts regarding Bayesian networks:

- *D-separation*, which allows us to visually identify the conditional independencies implied by the Bayesian network's graph.
- *Forward sampling*, which allows us to generate from the Bayesian network a set of samples $\boldsymbol{D}$, *i.e.,* instantiations of all of the network's random variables $\boldsymbol{D} = \{D_1, \ldots, D_M\}$, where $D_m = \{X_1 = x_1^{(m)}, \ldots, X_n = x_n^{(m)}\}$.

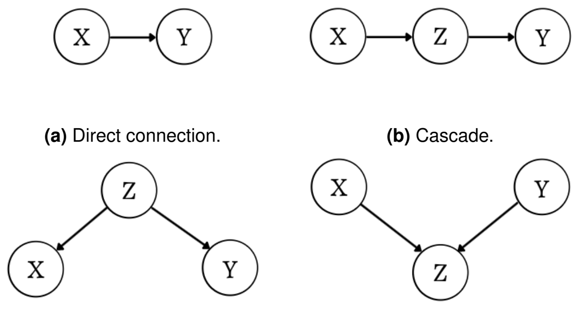
2.1.1 D-separation

In order to comprehend *d-separation*, it is necessary to understand its base concepts, specifically:

- *Independence*: Distribution P satisfies $(X \perp\!\!\!\perp Y)$ if and only if $P(X,Y) = P(X)P(Y)$, which means that knowing the outcome of X does not influence our belief in the outcome of Y.
- *Conditional independence*: Distribution P satisfies $(X \perp\!\!\!\perp Y \mid Z)$ if and only if $P(X,Y \mid Z) = P(X \mid Z)P(Y \mid Z)$, which means that given the value of Z, knowing the value of X does not influence our belief in the outcome of Y.

A direct acyclic graph encodes a specific set of conditional independence relationships between its variables [9]. In order to discover this set of independencies, the DAG can be seen as a combination of sub-graphs of the following types:

- *Direct connection*: G is of the form $X \to Y$, in which case $X \not\perp\!\!\!\perp Y \mid \boldsymbol{Z}$ regardless of $\boldsymbol{Z}$.
- *Cascade*: G is of the form $X \to Z \to Y$, in which case $X \perp\!\!\!\perp Y \mid Z$.
- *Common cause*: G is of the form $X \leftarrow Z \to Y$, in which case $X \perp\!\!\!\perp Y \mid Z$.
- *V-structure*: G is of the form $X \to Z \leftarrow Y$, in which case $X \perp\!\!\!\perp Y$ only holds if Z and Ch_Z are unknown.



(a) Direct connection. **(b)** Cascade. **(c)** Common cause. **(d)** V-structure.

Figure 2: Important sub-graphs.

Formally, sets $\boldsymbol{X}$ and $\boldsymbol{Y}$ are said to be *d-separated* given $\boldsymbol{Z}$ if there is no active trail between any node $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$ given $\boldsymbol{Z}$, which is denoted as d-sep$_G(\boldsymbol{X};\boldsymbol{Y} \mid \boldsymbol{Z})$.

A *trail* is an ordering of nodes $X_1, \ldots, X_n$, where consecutive nodes are connected by an edge regardless of its direction, and is considered *active* given a subset $\boldsymbol{Z}$ of observed variables if:

- For all *v-structures* $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, then X_i or any of its descendants are in $\boldsymbol{Z}$.
- No other node in $\{X_1, \ldots, X_n\}$ is in $\boldsymbol{Z}$.

2.1.2 Forward sampling

Given the joint probability function $P(X_1, \ldots, X_n)$ specified by a Bayesian network, we can sample variables in topological order [8]. This method is called *forward* (or *ancestral*) *sampling* and works as follows:

1. Start by sampling the variables with no parents.
2. Sample the variables on the next topological level by conditioning these variables' CPDs to the values sampled in the previous step.
3. Proceed to the next topological level until all n variables have been sampled.

2.1.3 Bayesian network parameter learning

In order to learn a Bayesian network's parameters, $\boldsymbol{\theta}$, there are two particularly important approaches [8]:

- *Maximum Likelihood Estimator (MLE)*;
- *Bayesian parameter estimation*.

For each variable, X, with parents, $\boldsymbol{U}$, if we represent the CPD $P(X \mid \boldsymbol{U})$ as a table, then we will have a parameter $\theta_{x|\boldsymbol{u}}$ for each combination of $x \in$ Val(X) and $\boldsymbol{u} \in$ Val$(\boldsymbol{U})$. By using the counts in the data for the different outcomes x, $\{M_{x,\boldsymbol{u}} \colon x \in \text{Val}(X)\}$, we can then compute the MLE parameters,

$$\hat{\theta}_{x|\boldsymbol{u}} = \frac{M_{x,\boldsymbol{u}}}{M_{\boldsymbol{u}}}\,, \tag{2}$$

where $M_{\boldsymbol{u}} = \sum_x M_{x,\boldsymbol{u}}$, and $M_{x,\boldsymbol{u}}$ is the number of times $D_m = x$ and $\boldsymbol{u}^{(m)} = \boldsymbol{u}$ in $\boldsymbol{D}$ [8].

Note that we need $M_{\boldsymbol{u}}$ data points to estimate the parameter $\hat{\theta}_{x|\boldsymbol{u}}$. As the number of parents $\boldsymbol{U}$ increases, the number of different $\boldsymbol{u}$ increases exponentially, therefore the number of data points that we expect to have for a single $\boldsymbol{u}$ decreases exponentially. This is known as *data fragmentation* and leads to *overfitting* and the presence of a large amount of zeros in the distribution due to unseen/rarely seen $\boldsymbol{u}$ in the data set. Therefore, if the data set is not representative of the real distribution, MLE can lead to parameters that prove themselves inadequate when dealing with unseen data.

A more sensible parameter estimator is to use *Bayesian estimation*, which starts with already existing prior CPDs that express our beliefs about the variables before the data was observed [12]. Those *priors* are then updated, using the state counts from the observed data.

Essentially, the *priors* are *pseudo state counts* that are added to the actual state counts before normalization. In some cases we can have expert knowledge on the distributions of the random variables, so we may want to encode our beliefs with specific priors. However, we usually use uniform priors, *i.e.,* priors that consider all states to be equally likely [8], such as the:

- *K2 prior*, which simply adds 1 to the count of every single state.
- *likelihood-equivalent uninformative Bayesian Dirichlet (BDeu) prior*, for which the *pseudo state counts* are the equivalent of having observed a specified number of uniform samples of each variable and each of its parents' configuration.

2.2. Bayesian network structure learning

Given a data set $\boldsymbol{D} = \{D_1, \ldots, D_M\}$, where D_m is an instantiation of all the variables in $\boldsymbol{V}$, Bayesian network structure learning is the problem of learning a graph structure from $\boldsymbol{D}$. Assuming $\boldsymbol{D}$ is complete, *i.e.,* all variables of $\boldsymbol{X}$ have an observed instantiation, its set of parameters is maximized using frequency counts from the data, as seen in section 2.1.3. Consequently, finding the optimal Bayesian network is reduced to finding the optimal structure that fits the data.

Since the focus is on Bayesian networks, the problem of structure learning amounts to learning the DAG from data. Traditional approaches are split into two major classes:

- Score-based approaches.
- Constraint-based approaches.

2.2.1 Score-based approaches

Score-based approaches for Bayesian network structure learning resort to a:

1. *Scoring function*, which is used to measure how well a given structure fits the data.
2. *Search algorithm*, which is used to find the best scoring structure out of all possible DAGs.

Useful scoring functions are *decomposible*, which means that the score for a given network B can be computed as the sum of scores for its individual variables, and *score-equivalent*, *i.e.,* networks that encode the same d-separation facts are scored the same.

Commonly used scoring functions fall into one of two camps:

- *Information-theoretic scoring functions*.
- *Bayesian scoring functions*.

The former are based in the log-likelihood function, which is the log probability of D given B. Assuming the data samples are independent and identically distributed, the *log-likelihood (LL)* can be computed as

$$\begin{aligned}\text{LL}(\boldsymbol{D} \mid B) &= \sum_{m=1}^{M} \log P(D_m \mid B) \\ &= \sum_{i=1}^{n} \sum_{m=1}^{M} \log P(X_i^{(m)} \mid \text{pa}_{X_i}^{(m)})\,,\end{aligned} \tag{3}$$

where $X_i^{(m)}$ is the instantiation of X_i in the data sample D_m, and $\mathsf{pa}_{X_i}^{(m)}$ is the instantiation of X_i's parents in the data sample D_m.

One of the most commonly used information-theoretic scoring functions is the *Bayesian Information Criterion (BIC) score* [8], which introduces to LL the penalty term:

$$\text{LL-PN}_{\text{MDL}}(X_i, B, D) = -\frac{\log M \times p_i}{2}, \tag{4}$$

where p_i is the number of parameters for X_i.

The BIC score is based on the *Minimum Description Length (MDL)* [13] and has been shown to be very competitive with scores that require more assumptions on the nature of the training data [14]. It requires a sufficiently large set of training data, since it is based on the asymptotic behavior of models.

On the other hand, Bayesian Dirichlet scoring functions are based on the *Bayesian Dirichlet (BD) score*. For a Bayesian network B with network structure G and Dirichlet priors, where $P(\boldsymbol{\theta}_{X_i|\mathsf{pa}_{X_i}} \mid G)$ has hyperparameters $\{(\alpha^G_{x_i|\boldsymbol{u}_i} : j = 1, \ldots, |X_i|\}$, the BD score is

$$\begin{aligned}\text{BD}(G, \boldsymbol{D}) = P(B) \prod_{i=1}^{n} \prod_{\boldsymbol{u}_i \in \text{Val}(\text{Pa}^G_{X_i})} \frac{\Gamma(\alpha^G_{X_i|\boldsymbol{u}_i})}{\Gamma(\alpha^G_{X_i|\boldsymbol{u}_i} + M_{\boldsymbol{u}_i})} \\ \cdot \prod_{x_i^j \in \text{Val}(X_i)} \frac{\Gamma(\alpha^G_{x_i|\boldsymbol{u}_i} + M_{x_i^j, \boldsymbol{u}_i})}{\Gamma(\alpha^G_{x_i^j|\boldsymbol{u}_i})},\end{aligned} \tag{5}$$

where $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the *Gamma function*, $M_{\boldsymbol{u}_i}$ is the number of instances in the data set $\boldsymbol{D}$ that $\text{Pa}_{X_i} = \boldsymbol{u}_i$, and $\alpha^G_{X_i|\boldsymbol{u}_i} = \sum_{m=1}^M \alpha^G_{x_i^j|\boldsymbol{u}_i}$.

Furthermore, if we assume that all network structures are equally likely and force those that encode the same d-separation facts to be assigned the same score, then we can compute the hyperparameters, $\alpha^G_{x_i^j|\boldsymbol{u}_i}$, using a single hyperparameter, α, which is known as *equivalent sample size*,

$$\alpha^G_{x_i^j|\boldsymbol{u}_i} = \frac{\alpha}{r_i \cdot q_i}, \tag{6}$$

where $r_i = |X_i|$ is the number of possible values of X_i, and $q_i = \sum_{X_j \in \text{Pa}_{X_i}} r_j$ is the number of possible configurations of the parent set, Pa_{X_i}, of X_i. Thus, we obtain the *BDeu score*, similar to section 2.1.3, yet with the prior over structures instead of parameters.

The density of the network structure is directly correlated to the value of α and it has been shown that it is very sensitive to it [15]. Therefore, when the density of the desired network structure is completely unknown, α's selection is not trivial.

Having now a score function that allows us to evaluate the fitness of possible network structures to the data, we want to evaluate possible DAGs.

The "simplest" method is known as exhaustive search and evaluates all possible directed acyclic graphs, choosing the one with the best score. Like all brute force methods, this quickly becomes intractable since the number of possible DAGs is super-exponential to the number of nodes [16]. If R_n is the number of DAGs with n vertices, then

$$R_n = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} R_{n-k}, \tag{7}$$

for $n \geq 1$, and with $R_0 = 1$.

Therefore a search strategy for traversing the possible DAGs search space is required. Here is where the decomposability of the scoring function comes in handy. Instead of scoring all possible DAGs, this property allows us to score mere edge operations, such as adding, deleting or reversing an edge. This allows the creation of greedy algorithms that iteratively perform the edge operation that maximizes the scoring function, starting either with an empty or a complete graph.

However, one should note that though this clever heuristic greatly simplifies the search over possible DAGs, it comes at a cost. Since the graph space is highly non-convex, there is the risk of getting stuck in local maxima of the scoring function.

The most renowned score-based search Bayesian network structure learning algorithm is the Greedy Equivalence Search (*GES*) algorithm. It conducts its search in the space of Markov Equivalence Classes, which are represented as completed partially directed acyclic graphs (CPDAGs), also known as *patterns* [17] and follows these steps:

1. Start with an empty graph, *i.e.,* all possible marginal and conditional independence constraints.
2. Repeatedly add or reverse a specific edge, with the chosen operation being the one with the highest score according to the chosen score function, until a maximum is reached.
3. Repeatedly remove edges, as long as it increases the scoring function.
4. When a maximum is reached, the result is the CPDAG of the desired structure.

Two DAGs are said to be in the same Markov Equivalence Class if they share the same d-separation facts. In this case, they can both be represented by a CPDAG which contains directed and undirected edges. In this graph, an undirected edge means that neither possible direction would alter the d-separation facts encoded by the graph [9].

2.2.2 Constraint-based approaches

Constraint-based approaches make use of independence tests between the variables, in order to identify a set of edge constraints that the best DAG must satisfy [18]. We know that if two variables are independent, then there is no edge connecting them. This type of approach requires extensive testing, so for large networks (above 200 nodes) it becomes intractable.

The most famous constraint-based Bayesian network structure learning algorithm is the PC algorithm [9]. Similarly to *Greedy Equivalence Search*, this algorithm reduces the DAG search space to the CPDAG search space. However, it reaches the skeleton of the desired graph differently, doing so as follows:

1. Start with a fully connected undirected graph.
2. For each pair of adjacent nodes X and Y, find the set of nodes $\boldsymbol{Z}$ that are adjacent to X yet are not Y.
3. Check if $X \perp\!\!\!\perp Y \mid Z$ holds, *i.e.,* if X and Y are conditionally independent given $Z \in \boldsymbol{Z}$.
4. If so, remove the edge connecting X and Y and add Z to the separation sets of X and Y, denoted S_{XY} and S_{YX} accordingly.

At the end of this process, we will have the skeleton of the desired graph. In order to transform it into the CPDAG, we will have to identify all possible v-structures in the following manner:

1. For each pair of non-adjacent nodes X and Y with common neighbor Z, check if $Z \notin S_{XY}$.
2. If so, then replace $X - Z - Y$ with $X \rightarrow Z \leftarrow Y$.
3. This results in a partially directed acyclic graph (PDAG), of which we can still assign a direction to certain undirected edges according to *Meek's rules* [19]

Finally, we will have obtained the desired CPDAG. From here, a specific DAG can then be extracted, as we will see later.

It is required to identify all possible v-structures in step 2) in order to obtain the CPDAG, because otherwise the graph would encode different d-separation facts.

The PC algorithm's evaluation of the independence between pairs of nodes that represent discrete random variables resorts to two well-known tests imported from the field of statistics, such as the *Chi-square test* [20] and the *G-square test* [9].

2.2.3 Hybrid approaches

This type of algorithms combines both approaches discussed previously. They follow these steps:

1. In the same vein as *constraint-based algorithms*, they use conditional information tests to infer the skeleton of the desired graph.
2. Then, they employ the methods of *score-based algorithms*, greedily performing local search, choosing the edge operation that maximizes a specified score.

These hybrid algorithms share both the positive and negative aspects of the previously mentioned approaches. An extensive conditional independence testing phase is intractable for large networks, though it has been shown to be theoretically sound. While the skeleton orientation phase incurs the risks of non-convex optimization, it does not provide any theoretical guarantees on the network structure itself [14].

A well-known algorithm that follows this kind of approach is the Min-Max Hill-Climb algorithm (*MMHC*), which has been shown to achieve competitive results when compared with the more traditional algorithms [21].

2.2.4 Continuous optimization approach

Recently, a new approach has been developed that avoids the need for extensive knowledge of graph theory and transforms the search over possible DAG space problem into a continuous optimization problem subject to a novel condition of acyclicity [10]. The author's approach amounts to an equality constraint optimization problem, which ensures the acyclicity of the resulting DAG

$$h(W) = \mathrm{tr}\left(e^{W \circ W}\right) - d = 0\,, \tag{8}$$

where W is the weighted adjacency matrix, $\circ$ is the *Hadamard product*, tr(.) is the trace operator, and d is the number of variables.

This in turn makes the previous *score-based approaches*, which were engaged with maximizing a specific score function while searching in the DAG (or CPDAG) space, into a continuous optimization problem of the form

$$\begin{aligned} \min_{W \in \mathbb{R}^{d \times d}} \quad & F(W) = \frac{1}{2n}\|X - XW\|_F^2 + \lambda\|W\|_1 \\ \text{s.t.} \quad & h(W) = 0\,, \end{aligned} \tag{9}$$

where n is the total number of samples, $\|.\|_F$ is the Frobenius norm, and λ is a regularization parameter that controls the sparsity of the identified weights.

This strategy used to solve this equality constraint problem (ECP) follows these steps:

1. Convert the constrained problem into a sequence of unconstrained sub-problems. This is achieved via the use of the augmented Lagrangian strategy [22].
2. Optimize the unconstrained sub-problems, for which they employ L-BFGS and Proximal Quasi-Newton optimization techniques [23].
3. Threshold the resulting weighted adjacency matrix, W.

This approach enables the use of several well-studied optimization techniques, such as gradient descent. Though it is a non-convex optimization problem, the authors found that the obtained results were close to the ones found by the state-of-the-art exact algorithm, *GOBNILP* [24].

3. Proposed solution

3.1. Regularized Search

While different structure learning algorithms return different graphs for the same data set, there is some overlap on the identified edges. This leads to the hypothesis that these common edges have a higher degree of certainty than the rest and represent some of the data's underlying dependencies.

Following this hypothesis, the following meta-algorithm was devised, combining the traditional state-of-the-art score-based search algorithm, *FGES*, with the recent continuous optimization algorithm, *NOTEARS*:

1. Apply the *FGES* algorithm, which starts from an empty graph and iteratively performs the edge operation (adding, deleting or reversing an edge) that

maximizes the graph's score according to the chosen score function. This will result in a CPDAG.
2. Apply the *NOTEARS* algorithm, which iteratively updates the whole weighted adjacency matrix via optimization of the objective function $F(W)$, see equation 9.
3. Compare the CPDAG obtained from the *FGES* algorithm with the DAG obtained from the *NOTEARS* algorithm in order to find the set of directed edges that were detected by both algorithms.
4. Using the set of directed edges that were detected by both algorithms as prior knowledge, apply the *FGES* algorithm again. However, instead of starting from an empty graph, it starts from a graph that only contains this set of common edges. This algorithm, *Reg-FGES*, will result in a CPDAG.

It has been shown [25] that the *FGES* algorithm, which is an optimized and parallelized version of the original *GES* algorithm [26], outperforms both *PC algorithm* [9] and *MMHC* algorithm [21]. Therefore, the meta-algorithm was applied with *FGES* and *NOTEARS* [10], in order to regularize *FGES*, thus obtaining *Reg-FGES*.

Since the devised method is a meta-algorithm, its overall computational complexity will be the sum of complexities of the individual algorithms. For instance, in the case of regularizing *FGES* with *NOTEARS*, the complexity is as follows:

1. *FGES*: $\mathcal{O}(n^2)$, where n is the number of nodes/variables [27]. Keep in mind that this "low" complexity is only achieved by sufficiently bounding the maximum node degree, otherwise it would be exponential in the number of variables $\mathcal{O}(e^n)$, since it is a combination problem.
2. *NOTEARS*: $\mathcal{O}(n^3)$, due to the fact that the innovative acyclicity constraint requires evaluating the weighted adjacency matrix exponential, $e^{W \circ W}$, see equation 8 [10].
3. *Reg-FGES*: $\mathcal{O}(n^2)$, same as *FGES*, yet in practice the newly-attained prior knowledge will restrict the search space of Markov Equivalent Classes, therefore speeding up the search. While *FGES* starts its search from the empty graph, its regularized version, *Reg-FGES*, will start from a graph containing the directed edges that were found by both *NOTEARS* and *FGES*.

Since *NOTEARS* clearly dominates the other algorithms, the overall complexity of the meta-algorithm will be $\mathcal{O}(n^3)$.

3.2. Extracting a DAG from a CPDAG

Since the CPDAG represents a Markov Equivalence Class, *i.e.,* a class of DAGs that represent the same set of conditional independences, it is well-nigh impossible to distinguish between DAGs within the same Markov Equivalence Class. This is due to the fact that DAGs in the same Markov Equivalence Class are score-equivalent, *i.e.,* the scoring function assigns the same value to them since they encode the same probability distribution [27].

So in order to extract a DAG from a CPDAG, the method described in [9] was applied, which works as follows:

1. Pick a random undirected edge from the CPDAG and give it a random direction.
2. Check if there are any remaining undirected edges that share the target node of the originally undirected edge that was given a direction in the previous step.
3. If so, then assign a direction to the undirected edges that were identified in the previous step, in such a way that no new *v-structures* are formed.
4. Repeat the previous two steps, effectively propagating the effect of the first step, as long as there are undirected edges that form a trail starting with the first oriented edge.
5. When all undirected edges that were along this trail are directed, repeat the process starting at the first step, for as long as there are undirected edges.
6. Finally, when all undirected edges have been oriented, we will have a DAG.

However, since certain edges were given an arbitrary direction, this may result in an output DAG where these edges may have the opposite orientation of the original one. In addition, when there is no ground truth graph it is even impossible to know whether this occurred at all.

This is precisely the motivating factor for the new meta-algorithm. For in this case it is impossible to know the *precision (PPV)* of a resulting DAG, *i.e.,* know how many of the identified edges are in fact *True Positives*. If in simulated-data experiments where we can use the previously discussed metrics, we find that the directed edges that were detected by both algorithms have a higher *PPV* than the ones detected by the individual algorithms, then the starting hypothesis will have been validated. While in real-data experiments, *i.e.,* when we only possess the data set, it follows that the common directed edges detected by both algorithms should also more likely be *True Positives*.

3.3. Comparing DAGs without ground truth

In real-data experiments, all we possess is the data set of instantiations of the random variables. Therefore, in order to evaluate the DAGs obtained by the different algorithms, we will make use of the concept of *relative entropy* [28].

The *relative entropy*, $D(p\|q)$, is a measure of the inefficiency of assuming that the distribution is q when the true distribution is p, *i.e.,* it is a measure of the distance between the two distributions. More formally, let $p(x)$ and $q(x)$ be two probability mass functions, where p is the true distribution, then the *relative entropy* is defined as

$$D(p\|q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \,. \tag{10}$$

The method this work proposes for evaluating and comparing the Bayesian networks learned from purely observational data is as follows:

1. Find an approximation of the joint probability distribution underlying the data, P^*, by computing the rela-

tive frequency of the distinct combinations of the random variable values (*i.e.,* rows) of the data set.

2. Using the DAG obtained by the various Bayesian network structure learning algorithms, obtain the full Bayesian network by learning its parameters, *i.e.,* the CPD tables, using MLE.
3. Use equation 1 for each of the unique combination of the random variable values identified in step 1), thus obtaining the joint probability distribution of the obtained Bayesian network, Q.
4. Compute the *relative entropy*, $D(P^* \| Q)$, using equation 10.

It is important to note that despite the drawbacks of MLE, it is the most appropriate Bayesian network parameter estimator for our purposes, since we also approximated the distribution underlying the data using relative frequency.

4. Results and discussion

4.1. Generated data

The *FGES* algorithm was implemented with the [29] package, while *NOTEARS* was implemented with the code publicly provided by the authors [10].

Since *FGES* requires specifying the maximum node, all trials were run with this parameter set to 10. Bounding this parameter is specially useful for larger networks, in order to lower the computation time.

The scoring functions that were used for *FGES* were those available in the well-renowned *Tetrad software suite*. Since all used Bayesian networks were discrete, the used scoring functions were the *bdeu-score* (*bdeu*), *discrete-bic-score* (*db*) and *degen-bic-score* (*dgb*).

Since *NOTEARS* produces a weighted adjacency matrix at the end of its optimization steps, this requires thresholding the individual weights, otherwise it might contain cycles. Following the recommendation of the authors [10] and also of the authors of one of its follow-up papers [30], this threshold value ω was set to 0.3.

To be able to test the hypothesis that edges identified by both algorithms were more likely to be true edges, there was the need to use well-known publicly available Bayesian networks, which can be seen on table 1.

Table 1: Properties of the used well-known Bayesian networks.

Network	Nodes	Edges	Parameters	Max In-Degree
Asia [31]	8	8	18	2
Child [32]	20	25	230	2
Insurance [33]	27	52	984	3
Alarm [2]	37	46	509	4
Hailfinder [1]	56	66	2656	4
Win95pts	76	112	574	7
Andes [34]	223	338	1157	6

Since the used repository (`https://www.bnlearn.com/bnrepository/`) supplies both the graphs and conditional probability tables for well-known Bayesian networks, in order to generate a data set for a given network, we simply need to repeatedly apply forward sampling.

In order to validate the starting hypothesis that the directed edges detected by both algorithms are highly likely to be true edges, since the true graph is known, then it is a simple matter of analyzing the precision values of these edges, see table 2.

Table 2: Precision of the directed edges that were detected by both *NOTEARS* and *FGES* for the various score functions.

PPV [%]	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
Common/bdeu	**100.0**	**100.0**	62.5	**95.0**	71.43	**92.45**	96.36
Common/db	-	**100.0**	**85.71**	94.44	**100.0**	86.36	**100.0**
Common/dgb	80.0	**100.0**	**85.71**	88.89	71.43	87.5	91.53

Apart from a few relatively low precision values, there are multiple instances where the precision values are perfect, whilst the rest are also relatively high. Motivated by these experimental results, the hypothesis now appears to be validated. That is, there is likely a set of edges crucial to the underlying probability distribution of the data, therefore being captured by both algorithms.

Comparing the adjacency precision values of the DAGs obtained by the various algorithms and score functions with the adjacencies expressed by the set of directed edges common to both *NOTEARS* and *FGES* only further corroborates the starting hypothesis. See table 3, which compares the precision values of the adjacencies obtained by the different algorithms and score functions. Note that it compares quartets (*NOTEARS*, *FGES*, *Reg-FGES* and the directed edges common to the former two) for each score function, with the best result out of the four in bold, while the best result overall is enclosed in parenthesis.

Table 3: Precision values of the adjacencies obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the various score functions.

PPV [%]	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	66.67	50.0	40.0	45.76	23.53	54.47	78.26
FGES/bdeu	50.0	**(100.0)**	79.25	91.67	66.67	62.34	86.14
R-FGES/bdeu	50.0	**(100.0)**	78.85	89.8	64.38	58.28	80.78
Common/*bdeu*	**(100.0)**	**(100.0)**	**(100.0)**	**(100.0)**	**85.71**	**(92.45)**	**96.36**
NOTEARS	66.67	50.0	40.0	45.76	23.53	54.47	78.26
FGES/db	83.33	**(100.0)**	92.86	93.33	75.38	78.43	89.15
R-FGES/db	83.33	**(100.0)**	92.86	91.3	75.38	73.91	88.79
Common/*db*	-	**(100.0)**	**(100.0)**	**(100.0)**	**(100.0)**	**90.91**	**(100.0)**
NOTEARS	66.67	50.0	40.0	45.76	23.53	54.47	78.26
FGES/dgb	83.33	**(100.0)**	72.13	77.19	79.71	75.68	82.16
R-FGES/dgb	83.33	**(100.0)**	72.13	75.86	71.01	67.97	80.06
Common/*dgb*	**(100.0)**	**(100.0)**	**85.71**	**94.44**	**85.71**	**89.58**	**96.61**

In order to analyze how close the learned DAGs are to the original network, we make use of the Structural Hamming Distance (SHD) metric, that counts the number of edge operation (addition, reversing, removing) that separate both of the considered DAGs. See table 4, which compares the *SHD* values of the DAGs obtained by the different algorithms and score functions. Note that it compares triplets (*NOTEARS*, *FGES* and *Reg-FGES*) for each score function, with the best result out of the three in bold, while the best result overall is enclosed in parenthesis. Also, keep in mind that for SHD, smaller is better.

Table 4: SHD values for the DAGs obtained by *NOTEARS*, *FGES* and *Reg-FGES*, with the latter two using the various score functions.

SHD	*Asia*	*Child*	*Insurance*	*Alarm*	*Hailfinder*	*Win95pts*	*Andes*
NOTEARS	9	33	75	63	154	129	383
FGES/bdeu	14	**(2)**	53	**(12)**	**52**	**84**	**115**
R-FGES/bdeu	**8**	4	**48**	21	53	101	165
NOTEARS	**9**	33	75	63	154	129	383
FGES/db	12	**4**	**(30)**	**15**	**39**	**70**	**(87)**
R-FGES/db	12	6	**(30)**	20	**39**	81	193
NOTEARS	9	33	75	63	154	129	383
FGES/dgb	**(6)**	**(2)**	39	**25**	**(33)**	**(67)**	**160**
R-FGES/dgb	**(6)**	6	**33**	30	47	84	193

In spite of the relatively poor results of *NOTEARS* in all metrics, *Reg-FGES* displays similar results to *FGES*. Though generally poorer than in the non-regularized version, *Reg-FGES* still manages to achieve the best or on-par results for many of the Bayesian networks, for the various score functions. In addition, even when it obtains a worse score, it is often close to the best achieved score, for instance compare the scores for the *Hailfinder* network for the *bdeu* score function or the ones for the *Andes* network for the *bd* score function.

While the regularization of *FGES* did not translate into significant improvements on the various metrics, the validity of the starting hypothesis is already a valuable development.

By discovering that the directed edges common to both *NOTEARS* and *FGES* have a higher likelihood of being present in the goal DAG, our knowledge of the previously unknown system, of which all that was known were observed instances of its variables, is now enriched with the discovery of some highly likely relationships.

4.2. Real data

In order to illustrate the applicability of the Bayesian network model to real-world data, the previously described Bayesian network structure learning algorithms were tested on some publicly available data sets. However, one should note that these data sets contain a set of mixed random variables, so in order to stay consistent with the focus on discrete random variables, the continuous variables were discretized.

Unlike generated data of section 4.1, now the DAG underlying the data is unknown. It is then impossible to apply metrics such as SHD and PPV to evaluate and compare the DAGs obtained by the various algorithms. This makes measuring the quality of the obtained DAGs based solely on their structure not trivial, specially when dealing with data from fields of study of which we do not possess expert knowledge.

For this reason, we will follow the method explained in section 3.3 so as to be able to evaluate and compare the learned Bayesian networks.

4.2.1 Data sets

The used data sets are publicly available and consist of:

- *Cardiovascular Disease Data Set* – This data set was obtained from *Kaggle* [35]. It is comprised of $70,000$ records of patient data collected at the moment of a medical examination to be used for diagnosing cardiovascular disease.
- *Acute Inflammation Data Set* – This data set was obtained from the *UCI Machine Learning Repository* [36]. It was created by a medical expert as a data set to test the expert system, which will perform the presumptive diagnosis of two diseases of the urinary system [37].
- *Marital Depression Data Set* – This data set was obtained from *Kaggle* [38]. It is comprised of the answers of married individuals from Istanbul to an online form, which aims to examine the influence of certain demographic factors on depression. In each form, an individual inputs his personal information and then answers the Beck Depression Inventory, which is a 21-question multiple-choice questionnaire widely used for measuring the severity of depression, focusing on the individual's thought regarding certain statements [39].
- *Titanic Data Set* – This data set was obtained from *Kaggle* [40]. It contains the data of passengers who boarded the *RMS Titanic* ship that sank after strinking an iceberg in the North Atlantic Ocean in 1912.

4.2.2 Overall Results

The relative entropy is used to measure how close the learned joint probability distribution is to the original distribution, being equal to zero when they are the same, therefore the lower its value the better.

The values of the relative entropy between the joint probability distribution underlying the data set and the joint probability distribution encoded by the Bayesian network learned by the various algorithms and score functions can be found on table 5.

Table 5: Relative entropy (KL-divergence) values of the distributions encoded the learned Bayesian networks for all data sets.

$D(P^*\|Q)$	*Cardio*	*Diagnosis*	*Marriage*	*Titanic*
NOTEARS	0.694	0.191	$\mathbf{0.287}$	0.765
FGES/bdeu	0.532	0.226	0.49	0.774
FGES/db	0.515	0.356	0.491	0.845
FGES/dgb	$\mathbf{0.512}$	0.279	0.493	$\mathbf{0.67}$
R-FGES/bdeu	0.532	$\mathbf{0.185}$	0.49	0.783
R-FGES/db	0.528	0.356	0.491	0.845
R-FGES/dgb	0.69	0.273	0.493	0.678

The obtained results show that:

- When using the same score function, the *FGES* and *Reg-FGES* algorithms generally achieve similar relative entropy values, except for the *degen-gauss-bic* score function on the *Cardio* data set.
- Unlike with the generated data sets of section 4.1 and apart from the *Cardio* data set, the *NOTEARS* algorithm generally proved to be competitive with the other algorithms, even achieving the best result on the *Marriage* data set.
- By comparing the best results for each data set, we find that the Bayesian networks obtained for the *Diagnosis* and *Marriage* data sets fit their respective observed data better than the ones obtained for the *Cardio* and *Titanic* data sets.

It is important to note that all used data sets are conducive to *classification* tasks, *i.e.,* where we want to predict the values of specific random variables based on knowing the values of the rest. These random variables are known as *decision variables*. For instance, consider a system for medical diagnosis, where the variables that represent specific symptoms would then influence the decision variable, which represents the diagnosis.

For causal inference, *i.e.,* determining causal links, the direction of the edges would then be from the other nodes to the decision node. However, barring the *Marriage* data set, on the DAGs obtained for the various data sets, the decision nodes were more akin to the root node of the naive Bayes network. In other words, the edges are outgoing from the decision nodes, therefore the various algorithms proved to be inadequate for causal inference.

In order to interpret the meaning of the edges in the obtained DAGs, we should think of $X \rightarrow Y$ not as "X *causes* Y", but instead as "*knowing* X *influences* Y".

Furthermore, the Bayesian networks that were fully learned (*i.e.,* both the structure and the parameters) allow us to reach a deeper understanding on the nature of the data sets. For instance, we can analyse the CPDs encoded by the learned Bayesian networks to measure the influence certain variables have on others or to ascertain how well the learned models are in accordance with expert knowledge.

5. Conclusions and future prospects

5.1. Final remarks

This work addressed the problem of learning Bayesian networks from observational data alone, focusing especially in the Bayesian network structure learning subproblem. First, we covered the traditional approaches and their most famous algorithms, which generally focus on restricting the DAG search space. Then, we covered the recently established continuous optimization approach of the *NOTEARS* algorithm, which focuses on the acyclicity constraint.

With a grasp of the different approaches, we then formulated the hypothesis that certain relationships are so strongly encoded in the data that they are identified by different Bayesian network structure learning algorithms. Thus, we developed a meta-algorithm, which combined the state-of-the-art traditional score-based search algorithm, *FGES*, with the *NOTEARS* algorithm. The resulting algorithm, *Reg-FGES*, takes that set of strong edges as prior knowledge and then applies the standard *FGES* algorithm starting from this set.

The results for the data generated from well-known Bayesian networks via forward sampling appear to validate the hypothesis, see table 2. However, the regularized algorithm, *Reg-FGES*, dit not attain a significant improvement over the original algorithm, *FGES*. This is likely because *FGES* does not guarantee the optimal solution (*i.e.,* it only returns local optima) or due to the random nature of the method used to extract a DAG from a CPDAG, which may assign an undirected edge the opposite direction of the proper edge on the original DAG.

Nonetheless, the directed edges detected by both *NOTEARS* and *FGES* were shown to be very likely, if not certain, to correspond to true relationships between those variables when not considering their direction (*i.e,* considering them as proper adjacencies), which further strengthened the hypothesis, see table 3.

On the other hand, when dealing with real data we did not know the DAGs underlying the data sets, which made impossible the use of traditional metrics (*e.g.,* accuracy, precision, recall, F_1-score, SHD). Therefore, we had to resort to the method described in section 3.3, so as to be able to evaluate and compare the results obtained by the different Bayesian network structure learning algorithms.

While on the generated data the *FGES* algorithm generally achieved the best performance, on the real data sets both *NOTEARS* and *Reg-FGES* achieved the best result on certain instances, see table 5.

Furthermore, the Bayesian networks were similar to the naive Bayes network, which means that the edges outgoing from the decision nodes have the opposite direction of the expected causal links. Therefore, an edge, $(X \rightarrow Y)$, in the obtained DAGs should be thought of as "*knowing* X *influences* Y", instead of "X *causes* Y".

It is important to note that we obtained the Bayesian network's parameters using MLE, therefore the learned models are inadequate for inference tasks on unseen data. Nonetheless, by analysing the learned CPDs we can gain a deeper knowledge of the influence variables have on each other. When allied with the learned Bayesian network structure, this allows us to understand the reasoning of the learned model, which is an advantage over neural networks where this is not trivial.

5.2. Future work

While developing the proposed solution presented in section 3, several choices were made that left unexplored avenues. This leads us to propose as lines of future work:

- With the aim of achieving markedly better results using the regularized form of the traditional algorithm as opposed to its original form, **extend the meta-algorithm by using other traditional Bayesian network structure learning algorithms**, such as the ones mentioned in section 2.2 (*e.g., PC* [9], *MMHC* [21]) or others (*e.g., LiNGAM* [41], *FCI* [9]), instead of just using the *FGES* algorithm.
- In order to reduce the computational complexity of the meta-algorithm, **use an improved version of *NOTEARS* or any other continuous optimization Bayesian network structure learning algorithm**, see [42] for a recent review of algorithms that follow this approach.
- To overcome the random nature of the method used for extracting a DAG from the CPDAG obtained by the *FGES* and *Reg-FGES* algorithms, **use causal directionality discovery methods to assign a direction to the undirected edges**, see [43] for an extensive review of these methods.
- For real data sets, where the underlying distribution is unknown, **use as ground truth the DAG obtained by the exact solver, *GOBNILP*** [24]. This would allow the use of the standard metrics to compare the DAGs obtained by the various Bayesian network structure learning algorithms.
- Another alternative to evaluate and compare the obtained DAGs for real data sets would be to **use *cross-validation*, learning the full Bayesian network from the *training data* and using the *testing data* to find the accurary of the learned models**.

References

[1] B. Abramson, J. Brown, W. Edwards, A. Murphy, and R.L. Winkler. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12(1):57–71, 1996.

[2] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, and G.F. Cooper. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In *In Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine*, pages 247–256. Springer-Verlag, 1989.

[3] Chris J. Needham, James R. Bradford, Andrew J. Bulpitt, and David R. Westhead. A Primer on Learning Bayesian Networks for Computational Biology. *PLOS Computational Biology*, 3(8):1–8, 8 2007.

[4] M. Beaumont and B. Rannala. The Bayesian Revolution in Genetics. *Nature Reviews Genetics*, 5:251–261, 2004.

[5] Mariia Voronenko, Dmytro Nikytenko, Jan Krejci, Oleksandr Naumov, Nataliia Savina, Elzara Topalova, Viktoriia Filippova, and Volodymyr Lytvynenko. Dynamic Bayesian Networks Application for Economy Competitiveness Situational Modeling. In *Advances in Intelligent Systems and Computing V*, volume 1293. Springer International Publishing, 2021.

[6] P. Whitney, A. White, S. Walsh, A. Dalton, and A. Brothers. Bayesian Networks for Social Modeling. *Social Computing, Behavioral-Cultural Modeling and Prediction*, 6589, 2011.

[7] Rosa Maria Arnaldo Valdés, V. Fernando Gómez Comendador, Alvaro Rodriguez Sanz, Eduardo Sanchez Ayra, Javier Alberto Pérez Castán, and Luis Perez Sanz. Bayesian Networks for Decision-Making and Causal Analysis under Uncertainty in Aviation. *Bayesian Networks - Advances and Novel Applications*, 2018.

[8] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive Computation and Machine Learning Series. The MIT Press, 1 edition, 2009.

[9] Peter Sprites, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Adaptive Computation and Machine Learning. The MIT Press, 2 edition, 2001.

[10] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[11] Marcus Kaiser and Maksim Sipos. Unsuitability of NOTEARS for Causal Graph Discovery, 2021.

[12] Evan Archer, Il Memming Park, and Jonathan W. Pillow. Bayesian and Quasi-Bayesian Estimators for Mutual Information from Discrete Data. *Entropy*, 15(5):1738–1755, 2013.

[13] Wai Lam and Fahiem Bacchus. Learning Bayesian Belief Networks: An Approach Based on the MDL Principle. *Computational Intelligence*, 10(3):269–293, 1994.

[14] Zhifa Liu, Brandon Malone, and Changhe Yuan. Empirical Evaluation of Scoring Functions for Bayesian Network Model Selection. *BMC Bioinformatics*, 13:S14, 09 2012.

[15] Tomi Silander, Petri Kontkanen, and Petri Myllymaki. On Sensitivity of the MAP Bayesian Network Structure to the Equivalent Sample Size Parameter. In *Proceedings of the 23rd Conference on Uncertainty in Machine Learning*, pages 360–367, 2012.

[16] Brendan D. McKay, Frederique E. Oggier, Gordon F. Royle, N. J. A. Sloane, Ian M. Wanless, and Herbert S. Wilf. Acyclic Digraphs and Eigenvalues of 0,1 Matrices. *Journal of Integer Sequences*, 7:1–5, 2003.

[17] Max Chickering. Optimal Structure Identification with Greedy Search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[18] Dimitris Margaritis. *Learning Bayesian Network Model Structure from Data*. PhD thesis, Carnegie Mellon University, 5 2003.

[19] Christopher Meek. Causal Inference and Causal Explanation with Background Knowledge. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 403–410. Morgan Kaufmann Publishers Inc., 1995.

[20] Stephen E. Fienberg. *The Analysis of Cross-Classified Categorical Data*. Springer, 2 edition, 2007.

[21] Ioannis Tsamardinos, Laura Brown, and Constantin Aliferis. The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm. *Machine Learning*, 65:31–78, 10 2006.

[22] Arkadi Nemirovski. *Lecture Notes in Optimization II: Standard Numerical Methods for Nonlinear Continuous Optimization*. Technion - Israel Institute of Technology, 1999.

[23] Kai Zhong, Ian E. H. Yen, Inderjit S. Dhillon, and Pradeep Ravikumar. Proximal Quasi-Newton for Computationally Intensive l_1-Regularized M-Estimators. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2375–2383. MIT Press, 2014.

[24] James Cussens. Bayesian Network Learning with Cutting Planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 153–160. Morgan Kauffman, 2012.

[25] Bryon Aragam and Qing Zhou. Concave Penalized Estimation of Sparse Gaussian Bayesian Networks. *Journal of Machine Learning Research*, 16:2273–2328, 2015.

[26] Joseph Ramsey, Madelyn Glymour, Ruben Sanchez-Romero, and Clark Glymour. A Million Variables and More: The Fast Greedy Equivalence Search Algorithm for Learning High-Dimensional Graphical Causal Models, With an Application to Functional Magnetic Resonance Images, 2017.

[27] Marco Scutari, Claudia Vitolo, and Allan Tucker. Learning Bayesian Networks from Big Data with Greedy Search: Computational Complexity and Efficient Implementation. *Statistics and Computing*, 29(5):1095–1108, 2018.

[28] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommnunications and Signal Processing. Wiley-Interscience, 2 edition, 2006.

[29] Chirayu (Kong) Wongchokprasitti, Harry Hochheiser, Jeremy Espino, Eamonn Maguire, Bryan Andrews, Michael Davis, and Chris Inskip. bd2kccd/py-causal v1.2.1, 2019.

[30] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG Structure Learning with Graph Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*. PMLR, 2019.

[31] Steffen Lauritzen and David John Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the royal Statistical Society. Series B (Methodological)*, [Royal Statistical Society, Wiley](2):157–224, 1988.

[32] David John Spiegelhalter and Robert G. Cowell. *Learning in Probabilistic Expert Systems*. Claredon Press, Oxford, 1992.

[33] John Binder, Daphne Koller, Stuart Russel, and Keiji Kanazawa. Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, 29(2-3):213–244, 11 1997.

[34] S. Conati, A.S. Gertner, K. VanLehn, and M.J. Druzdzel. On-Line Student Modeling for Coached Problem Solving Using Bayesian Networks. In *Proceedings of the 6th International Conference on User Modeling*, pages 231–242. Springer-Verlag, 1997.

[35] Svetlana Ulianova. Cardiovascular Disease dataset, 1 2017.

[36] Dheery Dua and Casey Graff. UCI Machine Learning Repository, 2019.

[37] J. Czerniak and H. Zarzycki. Application of Rough Sets in the Presumptive Diagnosis of Urinary System Diseases. In *Artificial Intelligence and Security in Computing Systems, ACS'2002 9th International Conference Proceedings*, pages 41–51. Kluwer Academic Publishers, 2003.

[38] babyoda. Depression in Married Couples in Istanbul, 11 2020.

[39] A.T. Beck, C.H. Ward, M. Mendelson, J. Mock, and J. Erbaugh. An Inventory for Measuring Depression. *Archives of General Psychiatry*, 4(6):561–571, 1961.

[40] Kaggle. Titanic - Machine Learning from Disaster.

[41] Shohei Shimizu, Patrik O. Hoyer, Asspo Hyvärinen, and Antti Kerminen. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research*, 7:2003–2030, 12 2006.

[42] Matthew J. Vowels, Necati Cihan Camgoz, and Richard Bowden. D'ya like DAGs? A Survey on Structure Learning and Causal Discovery, 2021.

[43] Joris M. Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing Cause from Effect Using Observational Data: Methods and Benchmarks. *Journal of Machine Learning Research*, 17(1):1103–1204, 1 2016.