Fast Bayesian Network Structure Learning using Quasi-determinism Screening

Thibaud Rahier, Sylvain Marié, Stéphane Girard, Florence Forbes

Fast Bayesian Network Structure Learning using Quasi-determinism Screening

Thibaud Rahier
Inria and Schneider Electric
Grenoble, France

Sylvain Marié
Schneider Electric
Grenoble, France

Stéphane Girard
Inria
Grenoble, France

Florence Forbes
Inria
Grenoble, France

Abstract

Learning the structure of Bayesian Networks from data is a NP-Hard problem that involves an optimization task on a super-exponential sized space. In this work, we show that in most real life datasets, a number of the arcs contained in the final structure can be pre-screened at low computational cost with a limited impact on the global graph score. We formalize the identification of these arcs via the notion of quasi-determinism, and propose an algorithm exploiting the screening that reduces the structure learning on a subset of the original variables. We show, on diverse benchmark datasets, that this algorithm exhibits a significant decrease in computational time for little decrease in performance score.

1 INTRODUCTION

Bayesian Networks are probabilistic graphical models that can present interest both in terms of knowledge discovery and density estimation. However, the problem of Bayesian Network learning from data has been proven to be NP-Hard by Chickering (1996).
There has been extensive work on tackling the ambitious problem of Bayesian Network structure learning from observational data. Algorithms fall under two main categories: *constraint-based* and *score&search*.

Constraint-based structure learning algorithms rely on testing for conditional independencies that hold in the data in order to reconstruct a Bayesian Network encoding this independencies. The *PC* algorithm by Spirtes et al. (2000) was the first practical application of this idea, followed by several optimized approaches as the *fast incremental association* (Fast-IAMB) algorithm from Yaramakala and Margaritis (2005).

Score&search structure learning, relies on the definition of a network score, then on the search for the best-scoring structure among all possible directed acyclic graphs (DAGs). There are $2^{\mathcal{O}(n^2)}$ possible DAG structures containing n nodes, which prevents exhaustive search for n typically bigger than 30.
Most of the score&search algorithms therefore rely on heuristics, as the original approach from Cooper and Herskovits (1992) which supposed a prior ordering of the variables to perform parent set selection, or Bouckaert (1995) who proposed to search through the space of possible DAGs using greedy hill climbing with random restarts. Since these first algorithms, different approaches have been proposed : some based on the search for an optimal ordering as Chen et al. (2008) or Teyssier and Koller (2012), others on pruning the search space in accordance to a given score (usually BIC) as de Campos and Ji (2011) and de Campos et al. (2017), and others on optimizing the search task through the DAG space as Scanagatta et al. (2015). Some exact methods have also been presented, as the ones by Yuan et al. (2011), Yuan and Malone (2012) Yuan et al. (2013) Silander and Myllymaki (2012) which are however unable to tackle datasets with more than 30 variables.

Moreover, real-world data now originate more and more from computer-based processes that by essence cause deterministic relationships to appear between variables. This is notably true when the data is stored in relational databases that compress the information using *keys* (TODO: deter read). Determinism has been shown to interfere with Bayesian Network structure learning, notably constraint-based methods, as mentioned by Luo (2006). Newer articles also tackle the problem of learning Bayesian Networks in a deterministic context, as de Morais et al. (2008) or Mabrouk et al. (2014).

In this paper, we focus on score&search algorithms. After reminding notations and state of the art con-

cerning Bayesian Network structure learning in section 2, we state and show some theoretical results in section 3, that enable to bridge the gap between determinism and Bayesian Networks scoring.
In section 4, exploiting the intuition brought by these theoretical results, we propose and study the complexity of the *quasi deterministic screening* algorithm, based on the idea that some of the arcs contained in the wanted Bayesian Network can be learned during a quick screening phase where quasi-deterministic relationships are detected, thus reducing the learning phase to a subset of the original variables.
In practice on benchmark datasets, not only does this algorithm accelerate the overall learning procedure with very low performance loss, but it also leads to sparser and therefore more interpretable graphs than state of the art methods, as shown in section 5.
Finally, section 6 is dedicated to the discussion and to numerous perspectives emerging from this work.

2 BAYESIAN NETWORK STRUCTURE LEARNING

2.1 Bayesian Networks

Let $X_1, \dots, X_n$ be n categorical random variables, with respective value sets $Val(X_1), \dots, Val(X_n)$. We note $\mathbf{X} = (X_1, \dots, X_n)$ and $Val(\mathbf{X}) = Val(X_1) \times \cdots \times Val(X_n)$. The joint distribution of $X_1, \dots, X_n$ is denoted by, for every $\mathbf{x} = (x_1, \dots, x_n) \in Val(\mathbf{X})$,

$$p(\mathbf{x}) = P(X_1 = x_1, \dots, X_n = x_n).$$

For $I \subset [\![1, n]\!]$, we define $\mathbf{X}_I = \{X_i\}_{i \in I}$, and we also use the notation $p(.)$ and $p(.|.)$ to refer to the marginals and conditionals of any subset of variables: $\forall (\mathbf{x}_I, \mathbf{x}_J) \in Val(\mathbf{X}_I) \times Val(\mathbf{X}_J)$, $p(\mathbf{x}_I|\mathbf{x}_J) = P(\mathbf{X}_I = \mathbf{x}_I|\mathbf{X}_J = \mathbf{x}_J)$.
Moreover, we suppose we have a dataset D containing M *i.i.d.* instances of $(X_1, \dots, X_n)$. We will use the notation $x_i[m]$ to denote the m^{th} observation of variable X_i in the dataset D.
Finally, all quantities written with an exponent D refer to these quantities empirically computed from D (*e.g.* p^D refers to the empirical distribution with respect to D).

A Bayesian network is an object $\mathcal{B} = (G, \theta)$ where

- $G = (V, A)$ is a directed acyclic graph (DAG) structure with V the set of nodes and $A \subset V \times V$ the set of arcs. We suppose $V = [\![1, n]\!]$ where each node $i \in V$ is associated with the random variable X_i, and $\pi^G(i) = \{j \in V \ s.t. \ (j, i) \in A\}$ is the set of i's parents. The exponent G may be dropped for clarity when the referred graph is obvious.
- $\theta = \{\theta_i\}_{i \in V}$, where each θ_i is the set of parameters defining the local conditional distribution $P(X_i|\mathbf{X}_{\pi(i)})$.
  More precisely for categorical variables, $\theta_i = \{\theta_{x_i|\mathbf{x}_{\pi(i)}}\}$ where for $i \in V$, $x_i \in Val(X_i)$ and $\mathbf{x}_{\pi(i)} \in Val(\mathbf{X}_{\pi(i)})$, we have

$$\theta_{x_i|\mathbf{x}_{\pi(i)}} = p(x_i|\mathbf{x}_{\pi(i)}).$$

A Bayesian Network $\mathcal{B} = (G, \theta)$ encodes the following factorization of the joint distribution of variables $X_1, \dots, X_n$:

$$p_{\mathbf{X}}(x_1, \dots, x_n) = \prod_{i=1}^{n} p(x_i|\mathbf{x}_{\pi^G(i)}) = \prod_{i=1}^{n} \theta_{x_i|\mathbf{x}_{\pi^G(i)}}.$$

Such a factorization implies a set of conditional independencies: *each variable is independent of its non-descendents given its parents.*

2.2 Score&search approach to Bayesian Network structure learning

We suppose we have a *scoring* function $s : DAG_V \to \mathbb{R}$, where DAG_V is the set of all possible DAG structures with node set V. Score&search Bayesian Network structure learning comes down to solving the following combinatorial optimization problem:

$$G^* \in \underset{G \in DAG_V}{\operatorname{argmax}} \, s(G). \tag{1}$$

It can be shown that $2^{\frac{n(n-1)}{2}} \leq |DAG_V| \leq 2^{n(n-1)}$ where $|V| = n$. There are therefore $2^{\mathcal{O}(n^2)}$ possible DAG structures containing n nodes (we say the size of DAG_V is super-exponential in $|V|$) which prevents exhaustive search when n gets typically bigger than 30.
Most scoring functions used in practice are based on the Max log Likelihood sore, that we now present.

The Max log-Likelihood score Let $l(\theta : D) = \log(p_\theta(D))$ be the log-Likelihood of the dataset D given the set of parameters θ. We define the Max Log Likelihood (MLL) score of a graph $G \in DAG_V$ associated with a dataset D as:

$$s^{MLL}(G : D) = \max_{\theta \in \Theta_G} l(\theta : D).$$

where Θ_G is the set of all θ's such that $\mathcal{B} = (G, \theta)$ is a Bayesian Network.
The MLL score is intuitive, but it favorizes denser structures: if $G_1 = (V, A_1)$ and $G_2 = (V, A_2)$ are two graph structures such that $A_1 \subset A_2$, we can show that: $s^L(G_1 : D) \leq s^L(G_2 : D)$.
There are two main (non-exclusive) approaches to solve this problem:

- Constraint the structure space to avoid learning overly complex graphs, which is the idea of hybrid structure learning algorithms such as the sparse candidate algorithm presented by Friedman et al. (1999), or the Max-Min Hill Climbing (MMHC) algorithm introduced by Tsamardinos et al. (2006).
- Use a penalized version of the MLL score, as the BIC score which is used in in many recent papers on score&search structure learning as de Campos and Ji (2011), Yuan et al. (2011) or Scanagatta et al. (2016).

The BIC score We define the BIC score of $G \in DAG_V$ as follows: (it can vary from a -2 factor, we chose to follow the definition of Koller and Friedman (2009) coherent with the implementation by Scutari (2009)):

$$s^{BIC}(G:D) = s^L(G:D) - \frac{\log(M)}{2}\mathcal{P}(G)$$

where M is the number of observations in D, and $\mathcal{P}(G)$ is the number of free parameters needed in the definition of a Bayesian Network $\mathcal{B} = (G, \theta)$, *i.e.* $\mathcal{P}(G) = \sum_{i=1}^{n}(|Val(X_i)| - 1)\ |Val(\mathbf{X}_{\pi^G(i)})|$, where by convention, $|Val(\mathbf{X}_\emptyset)| = 1$.

3 THEORETICAL RESULTS ON DETERMINISM AND BAYESIAN NETWORKS

3.1 Definitions

We first propose a definition of determinism using the notion of conditional entropy. In this paper, determinism (and later on quasi-determinism) will always be meant empirically (with respect to a dataset D).

Definition 1 ***Determinism wrt*** D
The relationship $X_i \to X_j$ is deterministic with respect to D iff $H^D(X_i|X_j) = 0$
where $H^D(X_i|X_j) = -\sum_{x_i,x_j} p^D(x_i, x_j)\log(p^D(x_i|x_j))$ is the empirical conditional Shannon entropy.

It is straightforward to prove that Definition 1 relates to a common and intuitive perception of determinism, as presented by Luo (2006)) for example. Indeed,

$$H^D(X_i|X_j) = 0$$
$$\Leftrightarrow \forall x_j \in Val(X_j), \exists! x_i \in Val(X_i)\ s.t.\ p^D(x_i|x_j) = 1.$$

Definition 2 ***Deterministic DAG wrt*** D
$G \in DAG_V$ is said to be deterministic with respect to D iff:

$$\forall i \in V \mid \pi(i) \neq \emptyset,\ H^D(X_i|\mathbf{X}_{\pi^G(i)}) = 0$$

3.2 Deterministic trees and MLL score

We first recall a lemma that relates the MLL score presented in section 2 to the notion of empirical conditional entropy. This result is well known and notably stated by Koller and Friedman (2009).

Lemma 1 *For $G \in DAG_V$ associated with variables $X_1, \dots, X_n$ observed in a dataset D,*

$$s^L(G:D) = -M\sum_{i=1}^{n} H^D(X_i|\mathbf{X}_{\pi(i)})$$

where by convention $H^D(X_i|\mathbf{X}_\emptyset) = H^D(X_i)$.

Proof: First let us rewrite the MLL score in terms of data counts. For a given $G \in DAG_V$ and $\theta \in \Theta_G$,

$$\begin{aligned} l(\theta:D) &= \sum_{m=1}^{M} \log(\underbrace{p_\theta(x_1[m]\dots,x_n[m])}_{\prod_{i=1}^n \theta_{x_i[m]|\mathbf{x}_{\pi(i)}[m]}}) \\ &= \sum_{m=1}^{M}\sum_{i=1}^{n} \log(\theta_{x_i[m]|\mathbf{x}_{\pi(i)}[m]}) \\ &= \sum_{i=1}^{n}\sum_{x_i,\mathbf{x}_{\pi(i)}} C^D(x_i,\mathbf{x}_{\pi(i)})\log(\theta_{x_i|\mathbf{x}_{\pi(i)}}) \end{aligned}$$

where $C^D(.)$ is the count function associated with D: $\forall I \subset V,\ C^D(\mathbf{x}_I) = \sum_{m=1}^{M} \mathbb{I}_{\mathbf{x}_I[m]=\mathbf{x}_I} = Mp^D(\mathbf{x}_I)$.
Moreover, it is well known that for categorical variables, the maximum likelihood estimator θ^{MLE} is given by the local empirical frequencies *i.e.*

$$\theta^{MLE}_{x_i|\mathbf{x}_{\pi(i)}} = p^D(x_i|\mathbf{x}_{\pi(i)}) = \frac{C^D(x_i,\mathbf{x}_{\pi(i)})}{C^D(\mathbf{x}_{\pi(i)})}.$$

Therefore we get:

$$\begin{aligned} s^L(G:D) &= \max_{\theta\in\Theta_G} l(\theta:D) \\ &= l(\theta^{MLE}:D) \\ &= \sum_{i=1}^{n}\sum_{x_i,\mathbf{x}_{\pi(i)}} C^D(x_i,\mathbf{x}_{\pi(i)})\log(\theta^{MLE}_{x_i|\mathbf{x}_{\pi(i)}}) \\ &= \sum_{i=1}^{n}\sum_{x_i,\mathbf{x}_{\pi(i)}} Mp^D(x_i,\mathbf{x}_{\pi(i)})\log(p^D(x_i|\mathbf{x}_{\pi(i)})) \\ &= -M\sum_{i=1}^{n} H^D(X_i|\mathbf{X}_{\pi(i)}). \end{aligned}$$

■

The next proposition follows then straightforwardly. We remind that a tree is a DAG in which each node but one (the root node) has exactly one parent.

Proposition 1 *If T is a deterministic tree with respect to D then*

$$s^L(T:D) = \max_{G \in DAG_V} s^L(G:D).$$

Before proving this proposition, it is worth noticing that dense DAGs also maximize the MLL score. The main interest of proposition 1 resides in the fact that, under a strong hypothesis, we are able to explicit a sparse solution of (1), with $n-1$ arcs compared to $\frac{n(n-1)}{2}$ for a dense DAG.

Proof: Let $G \in DAG_V$ with $V = [\![1,n]\!]$ and D containing observations of $X_1,\dots,X_n$ respectively associated with nodes in V.
First, we notice that $s^L(G:D)$is upper-bounded by the score of a dense DAG. We have shown in Lemma 1 that:

$$s^L(G:D) = -M\sum_{i=1}^{n} H^D(X_i|\mathbf{X}_{\pi(i)}).$$

It is commonly known that all DAGs are compatible with at least one ordering of the nodes, *i.e.* that $\exists \sigma \in \mathcal{S}_n$ such that

$$\forall i,j \in V \text{ s.t. } j \in \pi^G(i), \sigma(j) < \sigma(i).$$

In other words, σ represents an ordering in which each node comes after its parents.
Let $\sigma \in \mathcal{S}_n$ be an ordering compatible with G. Using the fact that for any variables X,Y,Z, $H^D(X|Y) \geq H^D(X|Y,Z)$ we then get that $\forall i \in V \setminus \{\sigma^{-1}(1)\}$,

$$H^D(X_i|\mathbf{X}_{\pi(i)}) \geq H^D(X_i|\mathbf{X}_{\sigma^{-1}(\{1,\dots,\sigma(i)-1\})}).$$

Plugging this inequality in the first equation, reordering the sum according to σ, and using the chain rule for entropies, we get:

$$\begin{aligned}
-\frac{s^L(G:D)}{M} &\geq \sum_{i=1}^{n} H^D(X_i|\mathbf{X}_{\sigma^{-1}(\{1,\dots,\sigma(i)-1\})}) \\
&= H^D(X_{\sigma^{-1}(1)}) \\
&\quad + \sum_{\sigma(i)=2}^{n} H^D(X_{\sigma^{-1}(\sigma(i))}|\mathbf{X}_{\sigma^{-1}(\{1,\dots,\sigma(i)-1\})}) \\
&= H^D(X_{\sigma^{-1}(1)}) \\
&\quad + H^D(X_{\sigma^{-1}(2)}|X_{\sigma^{-1}(1)}) + \dots \\
&\quad + H^D(X_{\sigma^{-1}(n)}|X_{\sigma^{-1}(1)},\dots,X_{\sigma^{-1}(n-1)}) \\
&= H^D(X_{\sigma^{-1}(1)},\dots,X_{\sigma^{-1}(n)}) \\
&= H^D(X_1,\dots,X_n),
\end{aligned}$$

which gives

$$s^L(G:D) \leq -M\ H^D(X_1,\dots,X_n).$$

Let T be as in the hypothesis of Proposition 1, we are now going to prove that this bound is reached for T which will give us the wanted result.
Without any loss of generality, let us suppose that T's root is 1. Then,

$$\begin{aligned}
s^L(T:D) &= -M\sum_{i=1}^{n} H^D(X_i|\mathbf{X}_{\pi(i)}) \\
&= -M\left(H^D(X_1) + \sum_{i=2}^{n} \underbrace{H^D(X_i|\mathbf{X}_{\pi(i)})}_{=0}\right) \\
&\geq -M\ H^D(X_1,\dots,X_n) \\
&= \max_{G \in DAG_V} s^L(G:D).
\end{aligned}$$

■

3.3 Deterministic forests and the MLL score

The deterministic tree hypothesis of Proposition 1 is very restrictive. In this section, it is extended to deterministic forests, defined as follows:

Definition 3 ***Deterministic forest wrt*** *D $F \in DAG_V$ is said to be a deterministic forest with respect to D iff $F = \bigcup_{k=1}^{p} T_k$, where $\bigcup$ is the canonical union for graphs: $G \cup G' = (V_G \cup V_{G'}, A_G \cup A_{G'})$, and $T_1,\dots,T_p$ are p disjoint deterministic trees wrt D such that $\bigcup_{k=1}^{p} V_{T_k} = V$.*

Moreover, for a given deterministic forest F wrt D, we define $R(F) \subset V$ to be the set of F's roots (the union of the roots of each of its trees), and note $D_{R(F)}$ the restriction of D to the observations of $\mathbf{X}_{R(F)}$.

Proposition 2 *We suppose F is a determinisitc forest wrt D. Let $G^*_{R(F)}$ be a solution of the structure learning optimization problem* (1) *for variables $X_1,\dots,X_p$ and the MLL score i.e.*

$$s^L(G^*_{R(F)} : D_{R(F)}) = \max_{G \in DAG_{R(F)}} s^L(G : D_{R(F)}).$$

Then, defining $G^* = F \cup G^*_{R(F)}$, we have

$$s^L(G^* : D) = \max_{G \in DAG_V} s^L(G:D).$$

Proof: Let $F = \bigcup_{k=1}^{p} T_k$ and $G^*_{r(F)}$ be as in the Proposition's hypotheses. Without loss of generality, we consider i to be the root of the tree T_i. Therefore, $R(F) = [\![1,p]\!]$.
Let us also define the following *root* function that associates to each node the root of the tree it belongs

to:

$$r : \left| \begin{array}{ccc} V & \longrightarrow & R(F) \\ i & \longmapsto & k \ s.t. \ X_i \in V_{T_k}. \end{array} \right.$$

Let $G^*_{R(F)} \in DAG_{R(F)}$ such that:

$$G^*_{R(F)} \in \underset{G \in DAG_{R(F)}}{\operatorname{argmax}} \ s^L(G : D)$$

and $G^* = F \cup G^*_{R(F)}$ *i.e.*

- $V_{G^*} = V$
- $A_{G^*} = (\bigcup_{k=1}^{p} A_{T_k}) \cup A_{G^*_{R(F)}}$

We will show as in the proof of Proposition 1 that

$$s^L_{DAG_V}(G^* : D) \geq \max_{G \in DAG_V} s^L_{DAG_V}(G : D)$$

which implies that $G^* \in \underset{G \in DAG_V}{\operatorname{argmax}} \ s^L_{DAG_V}(G : D)$.

We write:

$$\begin{aligned} s^L_{DAG_V}(G^*) &= -M \sum_{i=1}^{n} H^D(X_i | \mathbf{X}_{\pi^{G^*}(i)}) \\ &= \underbrace{-M \sum_{i=1}^{p} H^D(X_i | \mathbf{X}_{\pi^{G^*}(i)})}_{(a)} \\ &\quad \underbrace{-M \sum_{i=p+1}^{n} H^D(X_i | \mathbf{X}_{\pi^{G^*}(i)})}_{(b)} \end{aligned}$$

We then compute separately the terms (a) and (b):

- *Computation of (a)*
  The first term corresponds to the score of the graph $G^*_{R(F)}$ as an element of $DAG_{R(F)}$.
  Indeed, by construction of G^*,

  $$\forall i \in R(F), \ \pi^{G^*}(i) = \pi^{G^*_{R(F)}}(i).$$

  Moreover, $G^*_{R(F)}$ maximizes the MLL score on $DAG_{R(F)}$. We can now write:

  $$\begin{aligned} (a) &= -M \sum_{i=1}^{p} H^D(X_i | \mathbf{X}_{\pi^{G^*}(i)}) \\ &= -M \sum_{i=1}^{p} H^D(X_i | \mathbf{X}_{\pi^{G^*_{R(F)}}(i)}) \\ &= s^L(G^*_{R(F)} : D) \\ &= \max_{G \in DAG_{R(F)}} s^L(G : D_{R(F)}) \\ &= -MH^D(X_1, \dots, X_p). \end{aligned}$$

- *Computation of (b)*
  By construction of G^*,

  $$\forall i \in V \setminus R(F), \ \pi^{G^*}(i) = \pi^{T_{r(i)}}(i).$$

  Moreover since the T_k's are deterministic trees, it follows that

  $$\forall i \in V \setminus R(F), \ H^D(X_i | \mathbf{X}_{\pi^{T_{r(i)}}(i)}) = 0.$$

  Therefore we can write

  $$\begin{aligned} (b) &= -M \sum_{i=p+1}^{n} H^D(X_i | \mathbf{X}_{\pi^{G^*}(i)}) \\ &= -M \sum_{i=p+1}^{n} H^D(X_i | \mathbf{X}_{\pi^{T_{r(i)}}(i)}) \\ &= 0. \end{aligned}$$

Collecting the above results yields

$$\begin{aligned} s^L_{DAG_V}(G^*) &= (a) \\ &= -MH^D(X_1, \dots, X_p) \\ &\geq -MH^D(X_1, \dots, X_n) \\ &= \max_{G \in DAG_V} s^L(G : D). \end{aligned}$$

■

The idea of the proof relies on the fact that $H^D(X_1, \dots, X_p) = H^D(X_1, \dots, X_n)$, *i.e.* that the information relative to the variables $X_1, \dots, X_n$ is entirely contained in the roots $X_1, \dots, X_p$.
Even if this means reordering the variables, the assumptions of Proposition 2 are always verified: if there is no determinism in the dataset D, then $p = n$, $R(F) = V$, and every tree T_k is reduced to its root k. In that case solving problem (1) for $G^*_{R(F)}$ is the same as solving it for G^*.

The main issue with the MLL score for structure learning is that it favors dense graphs (as seen in section 2). However, a deterministic forest containing p trees is very sparse ($n - p$ edges), and even if the graph $G^*_{R(F)}$ is quite dense, the graph G^* may still satisfy sparsity conditions.
Let us state a specific example: suppose that we regularize the structure learning problem (1) with the MLL score by restricting the learning space DAG_V to $\{G \in DAG_V \mid \max_{i \in V} |\pi^G(i)| \leq P\}$ for $P \in \mathbb{N}$ significantly smaller than n. Assume that we have F a deterministic forest on $X_1, \dots, X_n$ containing $p \leq P$ trees. In that case, $G^* = F \cup G^{dense}_{R(F)}$ (with $G^{dense}_{R(F)}$ a fully connected DAG $\in DAG_{R(F)}$) will be a solution of problem (1) (from Proposition 2), while still satisfying the regularizing condition $\max_{i \in V} |\pi^G(i)| \leq P$. This idea is the inspiration for the algorithm presented in the next section.

4 STRUCTURE LEARNING WITH QUASI DETERMINISTIC SCREENING

4.1 Quasi-determinism

When it comes to Bayesian Network structure learning algorithms, even heuristics are computationally intensive. We would like to use the theoretical results presented in section 3 to simplify the structure learning problem.
Our idea is to reduce the problem to a subset of the variables: the roots of the deterministic forest, and therefore significantly decrease overall computation time: this is the idea of deterministic relationship screening.
However, in many datasets, we do not observe real empirical determinism ($H^D(X|Y) = 0$), although there are very strong relationships between some of the variables. Our second idea is to *relax* the aforementioned deterministic screening to quasi-deterministic screening, where *quasi* is meant with respect to a parameter ϵ: we will talk about $\epsilon-$*quasi-deterministic* relationships.
There are several ways to measure how close a relationship is from deterministic. Huhtala et al. (1999) considers the minimum number of observations that must be discarded from the data for the relationship to be considered deterministic.
We will rather use ϵ as a threshold on the empirical conditional entropy.

Definition 4 $\epsilon-$***quasi-determinism*** *(epsilon−**qd**) Given a dataset D containing observations of variables X and Y, the relationship $X \to Y$ is $\epsilon-$qd wrt D iff $H^D(Y|X) \leq \epsilon$.*

We have seen in Proposition 2 that deterministic forests are part of an optimal DAG with respect to the MLL score. Moreover, forests have a very low complexity in terms of number of arcs, which also implies a low complexity in terms of number of parameters if $\max_{1\leq i\leq n} |Val(X_i)|$ is reasonably bounded.
We are therefore confident that a DAG built from the union of a deterministic forest and an optimized root DAG will have good performance in terms of penalized log-likelihood scores as BDe (Heckerman et al. (1995)) or its asymptotical equivalent BIC (). Combining this intuition with the $\epsilon-$qd criteria presented in Definition 4, we propose the quasi-determinism screening approach to Bayesian Network structure learning, defined in the next subsections.

4.2 Quasi-determinism screening algorithm

The following algorithm details how to find the simplest $\epsilon-$qd forest F_ϵ from a dataset D and a threshold ϵ. Here *simplest* refers to the complexity in terms of number of parameters $\mathcal{P}(F_\epsilon)$.

Algorithm 1 Quasi-determinism screening (qds)

Input: D dataset containing M instances of variables $X_1, \ldots, X_n$, ϵ quasi determinism threshold
1: Compute empirical conditional entropy matrix $\mathbb{H}^D = \left(H^D(X_i|X_j)\right)_{1\leq i,j\leq n}$
2: **for** $i = 1$ to n **do** *# identify the set of potential $\epsilon-$qd parents for each i*
3: compute $\pi_\epsilon(i) = \{j \in [\![1, n]\!] \setminus \{i\} \mid \mathbb{H}^D_{ij} \leq \epsilon\}$
4: **for** $i = 1$ to n **do** *# check for cycles in $\epsilon-$qd relations*
5: **if** $\exists j \in \pi_\epsilon(i)$ *s.t.* $i \in \pi_\epsilon(j)$ **then**
6: **if** $\mathbb{H}_{ij} \leq \mathbb{H}_{ji}$ **then**
7: $\pi_\epsilon(i) \leftarrow \pi_\epsilon(i) \setminus \{j\}$
8: **else**
9: $\pi_\epsilon(j) \leftarrow \pi_\epsilon(j) \setminus \{i\}$
10: **for** $i = 1$ to n **do** *# choose the simplest among all potential parents*
11: $\pi^*_\epsilon(i) \leftarrow \underset{j\in\pi_\epsilon(i)}{argmin}\ |Val(X_j)|$
12: Compute forest $F_\epsilon = (V_{F_\epsilon}, A_{F_\epsilon})$ where $V_{F_\epsilon} = [\![1, n]\!]$ and $A_{F_\epsilon} = \{(\pi^*_\epsilon(i), i) \mid i \in [\![1, n]\!]$ *s.t.* $\pi^*_\epsilon(i) \neq \emptyset\}$
Output: F_ϵ

The next result is nontrival for $\epsilon > 0$, its proof is given in the supplementary material.

Proposition 3 *For any input D and ϵ, the output of Algorithm 1 is a forest (i.e. a directed acyclic graph with at most node parent per node).*

Proof: Let D and ϵ be objects that satisfy the input constraints of Algorithm 1, and let F_ϵ be the object that is returned by Algorithm 1 with inputs D and ϵ. F_ϵ is a **directed graph**, by definition. Moreover, it is built so that all of its nodes had at most one parent (line 12).
To conclude, we therefore only have to prove that F_ϵ does not contain cycles.
Let us suppose that there is a cycle $i_1, \ldots i_p$ in F_ϵ. There are two cases:

1. Either all associated variables have the same entropy: $H^D(X_{i_1}) = H^D(X_{i_2}) = \cdots = H^D(X_{i_p})$.
   In which case, there is necessarily two succesive nodes in the cycle i_l, i_{l+1} such that $i_l > i_{l+1}$.
   However, $H^D(X_{i_l}|X_{i_{l+1}}) = H^D(X_{i_{l+1}}|X_{i_l}) \leq \epsilon$, which means that when the algorithm reaches

line 4, $i_{l+1} \in \pi_\epsilon(i_l)$ and $i_l \in \pi_\epsilon(i_{l+1})$. Since i_{l+1} is treated before i_l in the for loop, this would result in i_l being removed from $\pi_\epsilon(i_{l+1})$, thus preventing for i_l to ever be i_{l+1}'s parent: we have a **contradiction**.

2. Either there exist at least two variables in the cycle that do not have the same entropy: $H^D(X_{i_k}) \neq H^D(X_{i_{k'}})$, for $k, k' \in [\![1, p]\!]$.
In this case, there also exist two successive nodes i_l, i_{l+1} such that $H^D(X_{i_l}) \neq H^D(X_{i_{l+1}})$.
Let us suppose that $H^D(X_{i_{l+1}}) > H^D(X_{i_l})$. In that case:

$$\begin{aligned} H^D(X_{i_l}|X_{i_{l+1}}) &= H^D(X_{i_{l+1}}|X_{i_l}) \\ &\quad + \underbrace{H^D(X_{i_l}) - H^D(X_{i_{l+1}})}_{<0} \\ &< H^D(X_{i_{l+1}}|X_{i_l}) \\ &\leq \epsilon. \end{aligned}$$

Therefore when the algorithm reaches line 4, $i_{l+1} \in \pi_\epsilon(i_l)$. But when treating either i_l or i_{l+1} during the for loop of lines 4-9, the test on line 6 necessarily implies that i_l is removed from $\pi_\epsilon(i_{l+1})$ (since $H^D(X_{i_l}|X_{i_{l+1}}) < H^D(X_{i_{l+1}}|X_{i_l})$). This is in **contradiction** with the fact that i_l is i_{l+1}'s parent.

Therefore, $H^D(X_{i_{l+1}}) < H^D(X_{i_l})$, and arcs in A_{F_ϵ} follow *nonincreasing entropies*, with at least one decrease since we suppose that all entropies are not equal. This is not possible in a cycle: there is a **contradiction**.

We conclude that there is **no cycle** in F_ϵ. ■

4.3 Learning Bayesian Networks using the quasi deterministic screening

We now present an algorithm that uses quasi-determinism screening to accelerate Bayesian Network structure learning. This algorithm takes the following input:

- D: a dataset
- ϵ: a threshold for quasi-determinism
- *sota-BNSL*: a state of the art structure learning algorithm, taking for input a dataset, and giving for an output a Bayesian Network structure. This algorithm is typically hill-climbing with tabu list and random restarts, associated with a penalized score like BDe or BIC.

Algorithm 2 Bayesian Network structure learning with quasi deterministic screening (qds-BNSL)

Input: D: dataset containing M instances of variables $\mathbf{X}_n = X_1, \ldots, X_n$
ϵ quasi-determinism threshold
$sota-BNSL$ state of the art structure learning algorithm

1: Compute F_ϵ by running **Algorithm 1** with inputs D and ϵ
2: Identify $R(F_\epsilon) = \{i \in [\![1, n]\!] \mid \pi^{F_\epsilon}(i) = \emptyset\}$, the set of F_ϵ's roots.
3: Compute $G^*_{R(F_\epsilon)}$ by running *sota-BNSL* on $\mathbf{X}_{R(F_\epsilon)}$
4: $G^* \leftarrow F_\epsilon \cup G^*_{R(F_\epsilon)}$

Output: G^*

One of the interests of Definition 3 for considering quasi-determinism is that it straightforwardly gives the following guarantee concerning the graph G^* returned by Algorithm 2:

Lemma 2 *Let ϵ, D and sota-BNSL be rightful inputs to Algorithm 2, and G^* the associated output.*
Then, if sota-BNSL is exact (i.e. always returns an optimal solution) with respect to the MLL score, we have the following lower bound for $s^L(G^* : D)$:

$$s^L(G^* : D) \geq \left(\max_{G \in DAG_V} s^L(G : D) \right) - Mn\epsilon.$$

Proof: The structure of the proof is the same as the one from Proposition 2. The only difference lies in the computation of term (b):

$$\begin{aligned} (b) &= -M \sum_{i=p+1}^{n} H^D(X_i|\mathbf{X}_{\pi^{G^*}(i)}) \\ &= -M \sum_{i=p+1}^{n} \underbrace{H^D(X_i|\mathbf{X}_{\pi^{T_{r(i)}}(i)})}_{\leq \epsilon} \\ &\geq -M(n-p)\epsilon \\ &\geq -Mn\epsilon. \end{aligned}$$

plugging this in the separated expression of the MLL score of G^* in terms (a) and (b) yields the wanted result.
■
In practice, this bound is not very tight and this result therefore has few applicative potential. However, it shows that for $\epsilon \to 0$, we are guaranteed to find the best overall graph (assuming optimality of *sota-BNSL*), going back to the setting of Proposition 2.
This algorithm is therefore promising, notably if for small ϵ we can significantly reduce the number of variables to be considered by *sota-BNSL*. We would in

this case have a much faster algorithm for only a small performance loss.

4.4 Complexity analysis

Complexity of the state of the art algorithm The number of possible DAG structures being super exponential in the number of nodes, state of the art is heuristics that do not entirely explore the structure space but which use smart caching and pruning methods to have a good performance & computation time trade-off.
Let *sota-BNSL* be a state of the art Bayesian Network structure learning algorithm and $C_{sota}(M, n)$ be its complexity.
$C_{sota}(M, n)$ should typically be thought of as linear in M and exponential, or at least high degree polynomial, in n for the best algorithms.

Complexity of Algorithm 1 We have the following decomposition of the complexity of Algorithm 1:

1. Lines 1-3: $\mathcal{O}(Mn^2)$. Computation of $\mathbb{H}$: we need counts for every couple (X_i, X_j) for $i < j$ (each time going through all rows of D), which implies $M\frac{n(n-1)}{2}$ operations.
2. lines 4-9: $\mathcal{O}(n^2)$. Going through $\mathbb{H}$ once.
3. lines 10-12: $\mathcal{O}(n^2)$. Going through $\mathbb{H}$ once.

Overall we have that $C_{Alg1}(M, n) = \mathcal{O}(Mn^2)$.

Complexity of Algorithm 2 For a given dataset D and threshold parameter ϵ, we define

$$n_\epsilon = |\{i \in [\![1, n]\!] \mid \pi^{F_\epsilon}(i) = \emptyset\}|$$

the number of roots of the forest F_ϵ returned by Algorithm 1. The complexity of Algorithm 2 then decomposes as:

1. Line 1: $\mathcal{O}(Mn^2)$. Run of Algorithm 1.
2. Lines 2-4: $C_{sota}(M, n_\epsilon)$. Run of *sota-BNSL* on reduced dataset $D_{R(F)}$ with n_ϵ columns.

This gives $C_{Alg2}(M, n) = \mathcal{O}(Mn^2) + C_{sota}(M, n_\epsilon)$. We are interested in how much it differs from $C_{sota}(M, n)$, which depends on two main elements:

- How n_ϵ compares to n,
- How $C_{sota}(M, n)$ varies with respect to n.

For example, if $C_{sota}(M, n) = \mathcal{O}(n^p)$ with $p \geq 3$ (which is realistic) and if $n_\epsilon = \frac{n}{2}$, we can expect an important decrease of computation time from *sota-BNSL* to *qds-BNSL* (Algorithm 2). If moreover this is true for small ϵ, then we also have the guarantee that the performance of the Bayesian Networks learnt will not differ too much.

It is hard to obtain a theoretical difference in computation time, since it is not clear how to estimate the complexity of state-of-the-art learning algorithms often based on local search heuristics. In the next section, we use benchmark datasets to run both a state of the art algorithm and Algorithm 2, in order to confirm our intuitions.

5 RESULTS

5.1 Setting

Data : Table 1 summarizes the data we used in our experiments.
We chose open-source[1] datasets presented by Davis and Domingos (2010) which contained more than 30 variables: 20 Newsgroup, Abalone, Adult, Book (Ziegler et al. (2005)), Covertype, KDDCup 2000, MSNBC, MSWeb, Plants and Reuters-52, some of which are also accessible on the UCI machine learning repository Lichman (2013). Moreover, we chose the largest Bayesian Networks available in the literature[2], for each of which we simulated 10000 observations: Alarm, Hailfinder, Hepar 2, Link, Munin 1-4.

Table 1: Dataset presentation

DATASET	n	M
20ng	930	11293
abalone	31	3134
adult	125	36631
book	500	8700
covertype	84	30000
kddcup2000	64	180092
msweb	294	29441
plants	69	17412
r52	941	6532
webkb	843	4874
wine	48	1728
alarm	37	10000
andes	223	10000
hailfinder	56	10000
hepar2	70	10000
link	724	10000
munin1	186	10000
munin2	1003	10000
munin3	1041	10000
munin4	1038	10000
pathfinder	109	10000

[1] `http://alchemy.cs.washington.edu/papers/davis10a/`
[2] urlhttp://www.bnlearn.com/bnrepository/

Algorithm Evaluation We will evaluate the algorithms using 3 axes of performance:

- Bayesian Network BIC score: this corresponds to the $s^{BIC}(. : D)$ presented in section 2. It is the most straightforward penalized-likelihood score, used in several recent papers as de Campos and Ji (2011), Yuan et al. (2011), Scanagatta et al. (2015), Scanagatta et al. (2016). It is quickly computable, with efficient caching, and does not require the tuning of a hyperparameter.
- Bayesian Network number of arcs. The complexity of Bayesian Networks is included in the aforementioned BIC score through the number of parameters. But it is interesting to look at the number of arcs, since this is closer in practice to how complex a Bayesian Network structure appears to a human being, and therefore to its interpretability.
- computing time t_{run} of the structure learning algorithm, ran on a cluster TODO specs Alexis.

Code and selected state-of-the-art algorithm Most of the code associated with this project was done in `R`, enabling an optimal exploitation of the `bnlearn` package from Scutari (2009), which is a very good reference among open-source package on Bayesian Networks structure learning.
We need a state-of-the-art Bayesian Network structure learning algorithm, both to use inside Algorithm 2 after the quasi-determinism screening phase, and to run separately on the full dataset to use as a reference for evaluating *qds-BNSL*. After carefully evaluating several algorithms implemented in the `bnlearn` package, we chose to use *Greedy Hill Climbing with 20 random restarts* as it consistently outperformed other algorithms both in time and score.

Choice of ϵ for *qds-BNSL* An approach to choosing ϵ in the case of the *qds-BNSL* algorithm is to pick values for n_ϵ, and reverse engineer the values of ϵ that enable those n_ϵ's.
It must be noted that since $\mathbb{H}$ is already computed and stored, evaluating the function $nbroots : \epsilon \mapsto n_\epsilon$ is done in constant time, and finding one of its quantiles is doable in at most $\mathcal{O}(\log(n))$ operations (dichotomy), which is negligible compared to the overall complexity of the screening phase.
For this exploratory analysis, we picked the following values for n_ϵ: $0.9n$, $0.75n$ and $0.5n$. For a given dataset, we write $\epsilon_x = nbroots^{-1}(\lfloor xn \rfloor)$ where $x \in [0, 1]$.

In the next section, we present the obtained results for our selected state of the art algorithm *sota*, and 3 versions of *qds-BNSL* which we refer to by their associated values of ϵ: $\epsilon_{0.9}$, $\epsilon_{0.75}$ and $\epsilon_{0.5}$.

5.2 Empirical results on benchmark datasets

For the BIC score table, we display in bold every result that is in a 5% range of the state of the art algorithm score.

Table 2: normalized negative BIC score

dataset	*sota*	$qd(\epsilon_{0.9})$	$qd(\epsilon_{0.75})$	$qd(\epsilon_{0.5})$
20ng	**144.22**	**145.02**	**146.86**	**150.23**
abalone	**5.65**	6.37	7.17	8.73
book	**35.93**	**36.07**	**36.28**	**37.23**
covertype	**13.89**	**13.91**	**14.03**	15.33
kddcup2000	**2.40**	**2.41**	**2.42**	**2.48**
msnbc	**6.22**	**6.24**	**6.36**	**6.48**
msweb	**9.90**	**9.90**	**9.90**	9.94
plants	**13.22**	**13.52**	14.14	15.84
r52	**97.44**	**98.03**	**98.99**	102.40
webkb	**14.48**	15.35	18.97	21.52
alarm	**10.66**	**10.80**	**11.17**	12.07
andes	**93.28**	**93.73**	99.07	108.69
hailfinder	**49.85**	**49.87**	**51.18**	54.86
hepar2	**32.63**	**32.72**	**33.06**	**33.65**
link	**218.87**	**220.53**	**222.47**	256.99
munin1	**43.49**	**43.51**	**43.56**	47.86
munin2	**171.49**	**171.85**	**171.85**	**175.25**
munin3	**172.44**	**172.44**	**172.44**	**173.97**
munin4	**194.77**	**194.90**	**194.90**	**200.87**
pathfinder	**28.60**	**28.64**	**28.62**	**29.82**

We see that *qds-BNSL* consistently requires less computational time that *sota-BNSL*, like suggested by the complexity study of the previous section. Moreover, the models learned with *qds-BNSL* are sparser and therefore more interpretable. This phenomenon increases as n_ϵ decreases.

For most of the datasets, the decrease in *BIC* score for *qds-BNSL* with $\epsilon_{0.9}$ and $\epsilon_{0.75}$ is smaller than 5%. This is even true for $\epsilon_{0.5}$ when the data contains a lot of very strong relationships as in *msweb*. Moreover, we observe a decrease in computational time and model complexity for *qds-BNSL*, which consistently intensifies when ϵ gets bigger.
In the best cases, we have both a small decrease in BIC score (less than 5%), and an important decrease in computational time ((*msweb*, TODO, even 30% for *TODO..*).

The following plot shows the relative change in BIC score, computation time and number of arcs, for *sota-BNSL* and *qds-BNSL* for $\epsilon_{0.9}$, $\epsilon_{0.75}$ and $\epsilon_{0.5}$,

Table 3: Computation time (seconds)

dataset	*sota*	$qd(\epsilon_{0.9})$	$qd(\epsilon_{0.75})$	$qd(\epsilon_{0.5})$
20ng	17193	13991	10005	4341
abalone	11	10	8	5
book	3851	3396	2939	1618
covertype	540	483	342	147
kddcup2000	1530	1424	958	447
msnbc	216	170	90	28
msweb	2710	2906	2863	1540
plants	227	188	137	57
r52	17193	13991	10005	4341
webkb	36	34	23	15
alarm	113	187	19	7
andes	4590	857	588	257
hailfinder	241	45	37	19
hepar2	391	72	45	23
link	32869	6045	5731	2252
munin1	2499	437	355	282
munin2	33403	6328	5541	2801
munin3	43040	6481	6963	3719
munin4	37982	6720	6656	4152
pathfinder	1141	193	148	65

Table 4: Model complexity (number of arcs)

dataset	*sota*	$qd(\epsilon_{0.9})$	$qd(\epsilon_{0.75})$	$qd(\epsilon_{0.5})$
20ng	3136	2995	2669	2136
abalone	86	85	76	56
book	1296	1237	1173	973
covertype	337	334	299	210
kddcup2000	217	214	188	148
msnbc	96	89	67	36
msweb	464	460	461	475
plants	291	265	237	170
r52	2713	2614	2465	2031
webkb	157	147	130	114
alarm	54	50	55	42
andes	336	333	312	259
hailfinder	64	63	63	55
hepar2	92	89	72	64
link	1137	1116	1150	885
munin1	208	208	210	188
munin2	874	879	879	761
munin3	898	898	898	828
munin4	905	903	903	826
pathfinder	161	154	147	122

averaged over all datasets.

6 DISCUSSION

The quasi-deterministic screening phase enables a decrease in computational time for a small decrease in graph performance. This tradeoff gets more and more advantageous when there actually are very srong 1 to 1 relationships in the data, that can be detected during the screening phase, and enable a decrease in the number of variables to be considered by the state of the art structure learning algorithm of the second phase.
Optimal cases for this algorithm happen when we get a huge decrease in the number of variables to consider after the screening phase for ϵ reasonably small (*e.g.* datasets *msweb*, TODO). We have tested our algorithm on industrial datasets corresponding to descriptive metadata, for which most of the variables possess empirically-deterministic parents: in this case we have $n_{\epsilon=0}$ very small with respect to n (typically $n_{\epsilon=0} \approx 2-3$ and $n \approx 80$), and *qds-BNSL* is up to two orders of magnitude faster for no performance loss.

The main axis of research is to anticipate how good the tradeoff may be before running any algorithm all the way through, thus preventing us from using *qds-BNSL* on datasets for which there are absolutely no strong 1 on 1 relationships, and enabling us to choose the perfect value of ϵ in cases where there is a lot of potential for computational time win with controlled performance loss. A first idea would be to used the bound presented in Lemma 2 TODO, even though this bound concerns the MLL score (and not the BIC score), and that it is far from tight in practice. Ideally, we would like to find a tight bound on BIC score to estimate the BIC loss of our graph, which would enable a good estimation of ϵ realizing the best tradeoff.

On the implementation's side, we still have potential to better the *qds-BNSL* algorithm, by paralellizing the computation of H, and implement it in `C` instead of `R`.

Finally, we have some ideas of how to generalize even more our quasi-detrministism screening idea: The proof of Proposition 2 suggests that the result still holds when F is *any kind of deterministic DAG* (and not only a forest). We could use rule induction, techniques that detect determinism in a broader sense that 1 to 1 to make the screening more efficient. For this purpose we could take inspiration from papers of the knowledge discovery in databases (KDD) community, as Huhtala et al. (1999), Dechter and Mateescu (2004), Liao et al. (2005) or more recently Papenbrock et al. (2015) which compares different functional dependencies discovery methods. We also could broaden our definition of quasi-determinism: we choose to consider

the information-theoretic quantity $H^D(Y|X)$ to describe the strength of the relationship $X \to Y$, since it is consistent with the rewriting of the MLL score given by Lemma 1. However, one could choose other quantities for this definition. For example, we could prefer considering $X \to Y$ to be $\epsilon-$qd when $\frac{H^D(X|Y)}{H^D(X)} \leq \epsilon$ (*i.e.* $\frac{MI^D(X,Y)}{H(Y)} \geq 1-\epsilon$) which gives another appreciation of the strength of the relationship: for a given variable Y, this comes down to finding a variable X such that $MI^D(X,Y)$ is high. This is connected to the ideas of Chow and Liu (1968) or Cheng et al. (1997) where pairwise empirical mutual information is central. Choosing this approach instead of the one given by Definition 3 does not change the upcoming algorithms and complexity considerations. Finally, we could use other definitions of entropy, as Renyi entropies presented by Rényi et al. (1961).

References

Bouckaert, R. (1995). *Bayesian belief networks: from inference to construction*. PhD thesis, PhD thesis, Faculteit Wiskunde en Informatica, Utrecht University.

Chen, X.-W., Anantha, G., and Lin, X. (2008). Improving bayesian network structure learning with mutual information-based node ordering in the k2 algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):628–640.

Cheng, J., Bell, D. A., and Liu, W. (1997). Learning belief networks from data: An information theory based approach. In *Proceedings of the sixth international conference on Information and knowledge management*, pages 325–331. ACM.

Chickering, D. M. (1996). Learning bayesian networks is np-complete. *Learning from data: Artificial intelligence and statistics V*, 112:121–130.

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

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

Davis, J. and Domingos, P. (2010). Bottom-up learning of markov network structure. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 271–278.

de Campos, C. P., Scanagatta, M., Corani, G., and Zaffalon, M. (2017). Entropy-based pruning for learning bayesian networks using bic. *arXiv preprint arXiv:1707.06194*.

de Campos, C. P. d. and Ji, Q. (2011). Efficient structure learning of bayesian networks using constraints. *Journal of Machine Learning Research*, 12(Mar):663–689.

de Morais, S. R., Aussem, A., and Corbex, M. (2008). Handling almost-deterministic relationships in constraint-based bayesian network discovery: Application to cancer risk factor identification. In *European Symposium on Artificial Neural Networks, ESANN'08*.

Dechter, R. and Mateescu, R. (2004). Mixtures of deterministic-probabilistic networks and their and/or search space. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 120–129. AUAI Press.

Friedman, N., Nachman, I., and Peér, D. (1999). Learning bayesian network structure from massive datasets: the 'sparse candidate 'algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc.

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

Huhtala, Y., Kärkkäinen, J., Porkka, P., and Toivonen, H. (1999). Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Liao, S. S., Wang, H. Q., Li, Q. D., and Liu, W. Y. (2005). A functional-dependencies-based bayesian networks learning method and its application in a mobile commerce system. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(3):660–671.

Lichman, M. (2013). UCI machine learning repository.

Luo, W. (2006). Learning bayesian networks in semi-deterministic systems. In *Canadian Conference on AI*, pages 230–241. Springer.

Mabrouk, A., Gonzales, C., Jabet-Chevalier, K., and Chojnacki, E. (2014). An efficient bayesian network structure learning algorithm in the presence of deterministic relations. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, pages 567–572. IOS Press.

Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.-P., Schönberg, M., Zwiener, J., and Naumann, F. (2015). Functional dependency discovery: An experimental evaluation of seven al-

gorithms. *Proceedings of the VLDB Endowment*, 8(10):1082–1093.

Rényi, A. et al. (1961). On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California.

Scanagatta, M., Corani, G., de Campos, C. P., and Zaffalon, M. (2016). Learning treewidth-bounded bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pages 1462–1470.

Scanagatta, M., de Campos, C. P., Corani, G., and Zaffalon, M. (2015). Learning bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pages 1864–1872.

Scutari, M. (2009). Learning bayesian networks with the bnlearn r package. *arXiv preprint arXiv:0908.3817*.

Silander, T. and Myllymaki, P. (2012). A simple approach for finding the globally optimal bayesian network structure. *arXiv preprint arXiv:1206.6875*.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press.

Teyssier, M. and Koller, D. (2012). Ordering-based search: A simple and effective algorithm for learning bayesian networks. *arXiv preprint arXiv:1207.1429*.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78.

Yaramakala, S. and Margaritis, D. (2005). Speculative markov blanket discovery for optimal feature selection. In *Data mining, fifth IEEE international conference on*, pages 4–pp. IEEE.

Yuan, C. and Malone, B. (2012). An improved admissible heuristic for learning optimal bayesian networks. *arXiv preprint arXiv:1210.4913*.

Yuan, C., Malone, B., et al. (2013). Learning optimal bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*.

Yuan, C., Malone, B., and Wu, X. (2011). Learning optimal bayesian networks using a* search. In *IJCAI proceedings-international joint conference on artificial intelligence*, volume 22, page 2186.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM.