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, 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 prescreened 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 associated algorithm that reduces the structure learning to a subset of the original variables. We show, on diverse benchmark datasets, that this algorithm exhibits a significant decrease in computational time and complexity for only a little decrease in performance score.

1 INTRODUCTION

Bayesian networks are probabilistic graphical models that present interest both in terms of knowledge discovery and density estimation. Learning Bayesian network from data has been however 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-based*.

Constraint-based structure learning algorithms rely on testing for conditional independencies that hold in the data in order to reconstruct a Bayesian network encoding these 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-based 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). The number of possible DAG structures is super-exponential in the number of nodes, which prevents exhaustive search when n is typically larger than 30, even for the most recent algorithms (Yuan and Malone (2012), Silander and Myllymaki (2012) or Yuan et al. (2013)).
Most of the score-based algorithms used in practice 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 structure space 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 optimizing the search task in accordance to a given score (usually BIC) as de Campos and Ji (2011), Scanagatta et al. (2015) and de Campos et al. (2017).

Meanwhile, data itself may contain determinism in domains such as cancer risk identification (de Morais et al. (2008)) or nuclear safety (Mabrouk et al. (2014)). Moreover, data is increasingly collected and generated by software systems whether in social networks, smart buildings, smart grid, smart cities or the internet of things (IoT) in general (Koo et al. (2016)). These systems in their vast majority rely on relational data models or lately on semantic data models (El Kaed et al. (2016)) which cause deterministic relationships between variables to be more and more common in datasets. Determinism has been shown to interfere with Bayesian network structure learning, notably constraint-based methods as mentioned by Luo (2006).

In this paper, we focus on score-based algorithms. Af-

ter reminding the background of Bayesian network structure learning in section 2, we state some theoretical results in section 3, that enable to bridge the gap between determinism and Bayesian network scoring.
In section 4, exploiting the intuition brought by these theoretical results, we propose and study the complexity of the *quasi deterministic screening* algorithm. The idea is that some of the arcs contained in the desired 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 presented in section 5.
Finally, section 6 is dedicated to a discussion and to numerous perspectives emerging from this work. Proofs of all lemmas and propositions are available in the supplementary material.

2 BAYESIAN NETWORK STRUCTURE LEARNING

2.1 Bayesian networks

Let $\mathbf{X} = (X_1, \dots, X_n)$ be a n-tuple of categorical random variables with respective value sets $Val(X_1), \dots, Val(X_n)$. The distribution of $\mathbf{X}$ is denoted by, $\forall\ \mathbf{x} = (x_1, \dots, x_n) \in Val(\mathbf{X})$,

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

For $I \subset [\![1, n]\!]$, we define $\mathbf{X}_I = \{X_i\}_{i \in I}$, and the notation $p(\cdot)$ and $p(\cdot|\cdot)$ is extended to the marginals and conditionals of any subset of variables: $\forall (\mathbf{x}_I, \mathbf{x}_J) \in Val(\mathbf{X}_{I \cup J})$, $p(\mathbf{x}_I|\mathbf{x}_J) = P(\mathbf{X}_I = \mathbf{x}_I | \mathbf{X}_J = \mathbf{x}_J)$.
Moreover, let D be a dataset containing M *i.i.d.* instances of $(X_1, \dots, X_n)$. All quantities empirically computed from D will be written with a $.^D$ exponent (*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 in G. The exponent G may be dropped for clarity when the referred graph is obvious.
- $\theta = \{\theta_i\}_{i \in V}$ is a set of parameters. Each θ_i defines the local conditional distribution $P(X_i|\mathbf{X}_{\pi(i)})$.
  More precisely, $\theta_i = \{\theta_{x_i|\mathbf{x}_{\pi(i)}}\}$ where for $i \in V$, $x_i \in Val(X_i)$ and $\mathbf{x}_{\pi(i)} \in Val(\mathbf{X}_{\pi(i)})$,
  $$\theta_{x_i|\mathbf{x}_{\pi(i)}} = p(x_i|\mathbf{x}_{\pi(i)}).$$

A Bayesian network $\mathcal{B} = (G, \theta)$ encodes the following factorization of the distribution of $\mathbf{X}$: for $\mathbf{x} = (x_1, \dots, x_n) \in Val(\mathbf{X})$,

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|\mathbf{x}_{\pi^G(i)}) = \prod_{i=1}^{n} \theta_{x_i|\mathbf{x}_{\pi^G(i)}}.$$

Such a factorization notably implies that *each variable is independent of its non-descendents given its parents.*

2.2 Score-based approach to Bayesian network structure learning

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-based Bayesian network structure learning comes down to solving the following combinatorial optimization problem:

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

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: the size of DAG_V is said to be super-exponential in $|V|$.
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 set of parameters θ given the dataset D.
For a given DAG structure $G \in DAG_V$, we define the Max log-likelihood (MLL) score of G 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 very straightforward and 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: $s^{MLL}(G_1 : D) \leq s^{MLL}(G_2 : D)$.
There are two main (non-exclusive) approaches to solve this problem:

- Constrain 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 BIC (Schwarz et al. (1978)) or BDe (Heckerman et al. (1995)).

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)):

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

where M is the number of observations in D, and $\mathcal{P}(G)$ is the dimension (*i.e.* number of free parameters) of a Bayesian network $\mathcal{B} = (G, \theta)$, given by:

$$\mathcal{P}(G) = \sum_{i=1}^{n}(|Val(X_i)| - 1)\ |Val(\mathbf{X}_{\pi^G(i)})|,$$

where by convention, $|Val(\mathbf{X}_\emptyset)| = 1$.

3 DETERMINISM AND BAYESIAN NETWORKS

3.1 Definitions

We propose the following definitions of determinism and deterministic DAGs using the notion of conditional entropy. In this paper, determinism will always be meant empirically, with respect to a dataset D.

Definition 1 ***Determinism wrt*** D
*Given a dataset D containing observations of variables X_i and X_j, the relationship $X_i \rightarrow X_j$ 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 the one presented by Luo (2006). Indeed,

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

This definition is naturally extended to $\mathbf{X}_I$ and $\mathbf{X}_J$ for $I, J \subset V$, *i.e.* $\mathbf{X}_I \rightarrow \mathbf{X}_J$ is deterministic with respect to D iff $H^D(\mathbf{X}_J|\mathbf{X}_I) = 0$.

Definition 2 ***Deterministic DAG wrt*** D
$G \in DAG_V$ is said to be deterministic with respect to D iff $\forall i \in V$ s.t. $\pi^G(i) \neq \emptyset$, $\mathbf{X}_{\pi^G(i)} \rightarrow X_i$ is deterministic wrt D.

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 $X_1, \ldots, X_n$ observed in a dataset D,*

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

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. We denote $x_i\,[m]$ the m^{th} observation of variable X_i in the dataset D. For a given $G \in DAG_V$ and $\theta \in \Theta_G$,

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

where $C^D(\cdot)$ 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 has exactly one parent, except one node (the root node) which has none.

Proposition 1 *If T is a deterministic tree with respect to D then T is a solution of* (1)*:*

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

Proof: Let $G \in DAG_V$ with $V = [\![1, n]\!]$ and D containing observations of $X_1, \ldots, X_n$ 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

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

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 $\forall i \in V \setminus \{\sigma^{-1}(1)\}$,

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

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

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

which gives

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

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,

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

■

It is worth noticing that complete 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 instead of $\frac{n(n-1)}{2}$ for a complete DAG.

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 $T_1, \ldots, T_p$ are p disjoint deterministic trees wrt D s.t. $\bigcup_{k=1}^{p} V_{T_k} = V$.

In the expression $\bigcup_{k=1}^{p} T_k$, $\cup$ is the canonical union for graphs: $G \cup G' = (V_G \cup V_{G'}, A_G \cup A_{G'})$.
For a given deterministic forest F wrt D, we define

$$R(F) = \{i \in V \mid \pi^F(i) = \emptyset\}$$

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 *Suppose F is a deterministic forest wrt D. Let $G^*_{R(F)}$ be a solution of the structure learning optimization problem* (1) *for $\mathbf{X}_{R(F)}$ and the MLL score i.e.*

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

Then, $G^* = F \cup G^*_{R(F)}$ is a solution of (1) *for* $\mathbf{X}$*:*

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

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:

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

- *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

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

■

The idea of the proof relies on the fact that $H^D(\mathbf{X}) = H^D(\mathbf{X}_{R(F)})$: the information associated with all the variables $X_1, \ldots, X_n$ is entirely contained in the root variables $\mathbf{X}_{R(F)}$.
It should be noted however that the assumptions of Proposition 2 are always verified: if there is no determinism in the dataset D, then $R(F) = V$, 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^*.

As seen in section 2, the main issue with the MLL score for structure learning is that it favors complete graphs. However, a deterministic forest containing p trees is very sparse ($n - p$ arcs), and even if the graph $G^*_{R(F)}$ is dense, the graph G^* may still satisfy sparsity conditions.
Let us state a specific example: suppose the structure learning problem (1) with the MLL score is regularized by a restriction of 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. If F is a deterministic forest wrt D containing $p \leq P$ trees, then $G^* = F \cup G^{comp}_{R(F)}$ (with $G^{comp}_{R(F)}$ a fully connected DAG $\in DAG_{R(F)}$) would 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-DETERMINISM 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 original variables, the roots of a deterministic forest, in order to significantly decrease the overall computation time. This is what we call determinism screening.

However, in many datasets, one does not observe real empirical determinism, although there are very strong relationships between some of the variables. We therefore propose to *relax* the aforementioned determinism screening to quasi-determinism screening, where *quasi* is meant with respect to a parameter ϵ: we talk about $\epsilon-$*quasi-determinism.*
There are several ways to measure how close a relationship is from deterministic. Huhtala et al. (1999) consider the minimum number of observations that must be dropped from the data for the relationship to be deterministic. Since we are in a score-maximization context, we will rather use ϵ as a threshold on the empirical conditional entropy. The following definition is the natural generalization of Definition 1.

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

It has been seen in Proposition 2 that deterministic forests are included in 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 $\max\limits_{1\leq i\leq n} |Val(X_i)|$ 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 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

Algorithm 1 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 $\mathcal{P}(F_\epsilon)$.

This algorithm takes for input:

- D: a dataset containing M observations of $\mathbf{X}$,
- ϵ: a threshold for quasi-determinism.

Algorithm 1 Quasi-determinism screening (qds)

```
    Input: $D$ , $\epsilon$
 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 $\mathbb{H}^D_{ij} \leq \mathbb{H}^D_{ji}$ then
 7:             $\pi_\epsilon(j) \leftarrow \pi_\epsilon(j) \setminus \{i\}$
 8:         else
 9:             $\pi_\epsilon(i) \leftarrow \pi_\epsilon(i) \setminus \{j\}$
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: $F_\epsilon$
```

The next proposition states that Algorithm 1 is well defined.

Proposition 3 *For any rightful input D and ϵ, the output of Algorithm 1 is a forest (i.e. a directed acyclic graph with at most one 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 $i_1, \dots i_p$ in F_ϵ. There are two cases:

1. Either all associated variables have the same entropy: $H^D(X_{i_1}) = H^D(X_{i_2}) = \dots = H^D(X_{i_p})$.
   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 is treated before i_{l+1} 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. ■

Proposition 4 *Let ϵ, D and sota-BNSL be rightful input to Algorithm 2, and G^*_ϵ 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 $s^{MLL}(G^*_\epsilon : D)$:*

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

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.
■

4.3 Learning Bayesian networks using quasi-determinism screening

We now present Algorithm 2 that uses quasi-determinism screening to accelerate Bayesian network structure learning. This algorithm takes the following input:

- D: a dataset containing M observations of $\mathbf{X}$,
- ϵ: a threshold for quasi-determinism,
- *sota-BNSL*: a state of the art structure learning algorithm, taking for input a dataset, and returning a Bayesian network structure.

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

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

One of the advantages of the introduction of quasi-determinism given by Definition 3 is that it straightforwardly yields the following guarantee concerning the graph G^*_ϵ returned by Algorithm 2:

Proposition 5 *Let ϵ, D and sota-BNSL be rightful input to Algorithm 2, and G^*_ϵ 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 $s^{MLL}(G^*_\epsilon : D)$:*

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

In practice, this bound is not very tight and this result therefore has few applicative potential. However, it shows that:

$$s^{MLL}(G^*_\epsilon : D) \underset{\epsilon \to 0}{\longrightarrow} \max_{G \in DAG_V} s^{MLL}(G : D).$$

In other words, $\epsilon \mapsto s^{MLL}(G^*_\epsilon : D)$ is continuous in 0, and Proposition 4 generalizes Proposition 2.

Algorithm 2 is therefore promising, notably if for small ϵ we can significantly decrease the number of variables to be considered by *sota-BNSL*. We would in this case have a much faster algorithm for a controlled 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 algorithms do not entirely explore the structure space but 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 $\mathbb{H}^D$: 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 all elements of $\mathbb{H}^D$ once.
3. lines 10-12: $\mathcal{O}(n^2)$. Going through all elements of $\mathbb{H}^D$ once.

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

Complexity of Algorithm 2 For a given dataset D, we define:

$$\forall \epsilon \geq 0,\ n_r(\epsilon) = |R(F_\epsilon)|.$$

The function $n_r(\cdot)$, associates to $\epsilon \geq 0$ 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: $C_{sota}(M, n_r(\epsilon))$. Run of *sota-BNSL* on reduced dataset $D_{R(F_\epsilon)}$ with $n_r(\epsilon)$ columns.

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

- how $n_r(\epsilon)$ compares to n,
- how $C_{sota}(M, n)$ varies with respect to n.

It is hard to obtain a theoretical difference $C_{sota} - C_{Alg2}$, since it is not clear how to estimate the complexity of state of the art learning algorithms, often based on local search heuristics. However, we know that all Bayesian network structure learning algorithms are very time-intensive, which lets us expect an important decrease in computational time for Algorithm 2 compared to a state of the art algorithm.
In the next section, we run a state of the art structure learning algorithm and Algorithm 2 on benchmark datasets in order to confirm this intuition.

5 EXPERIMENTS

Data Table 1 summarizes the data used in our experiments. We chose open-source[1] datasets presented by Davis and Domingos (2010) which contained more than 50 variables: 20 Newsgroup, Adult, Book (Ziegler et al. (2005)), Covertype, KDDCup 2000, MSWeb, Plants and Reuters-52, some of which are also accessible on the UCI machine learning repository (Lichman (2013)). Moreover, as it was done by Scanagatta et al. (2016), we chose the largest Bayesian networks available in the literature[2], for each of which we simulated 10000 observations: Andes, Hailfinder, Hepar 2, Link, Munin 1-4, PathFinder and Win95pts.

Table 1: Datasets presentation

name	short name	n	M
20 newsgroups	20ng	930	11293
adult	adult	125	36631
book	book	500	8700
covertype	covertype	84	30000
kddcup 2000	kddcup	64	180092
msweb	msweb	294	29441
plants	plants	69	17412
reuters 52	r52	941	6532
andes	andes	223	10000
hailfinder	hailfinder	56	10000
hepar 2	hepar2	70	10000
link	link	724	10000
munin 1	munin1	186	10000
munin 2	munin2	1003	10000
munin 3	munin3	1041	10000
munin 4	munin4	1038	10000
pathfinder	pathfinder	109	10000
windows 95 pts	win95pts	76	10000

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 packages treating 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 built-in

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

algorithms both in time and score. In what follows, we refer to this algorithm as *sota-BNSL*.

Choice of ϵ for *qds-BNSL* An approach to choosing ϵ in the case of the *qds-BNSL* algorithm is to pick values for $n_r(\epsilon)$, and manually find the corresponding values for ϵ. For a given dataset and $x \in [0, 1]$, we define $\epsilon_x = n_r^{-1}(\lfloor xn \rfloor)$. In other words, ϵ_x is the value of ϵ for which the number of roots of the deterministic forest F_ϵ represents the proportion x of the total number of variables.
It must be noted that once $\mathbb{H}^D$ is computed and stored, evaluating $n_r(\epsilon)$ can be done in constant time, and finding one of $n_r(\cdot)$'s quantiles is doable in at most $\mathcal{O}(\log(n))$ operations (dichotomy), which is negligible compared to the overall complexity of the screening phase.

Algorithm evaluation The algorithms are evaluated using 3 axes of performance:

- Bayesian network BIC score: this corresponds to the s^{BIC} presented in section 2. It is the most straightforward penalized-likelihood score, used in a number of 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 (all algorithms were run on the same machine).

We now present the obtained results for our selected state of the art algorithm *sota-BNSL*, and 3 versions of *qds-BNSL*. We picked the following values for $n_r(\epsilon)$: $0.9n$, $0.75n$ and $0.5n$ (corresponding to the values $\epsilon_{0.9}$, $\epsilon_{0.75}$ and $\epsilon_{0.5}$ of ϵ), in order to reduce the learning task to 90%, 75% and 50% of the original variables respectively.
The results are shown in Tables 2-4, one per evaluation criterion. In each table, the actual value of the criterion is displayed for *sota-BNSL*, and the relative difference is displayed for the three versions of *qds-BNSL* we consider ($\epsilon_{0.9}$, $\epsilon_{0.75}$ and $\epsilon_{0.5}$).

It appears in Table 2 that the decrease in BIC score is smaller than 5% for all the considered datasets when 90% variables remain after the pre-screening ($\epsilon_{0.9}$), and for most of them when 75% variables remain

Table 2: BIC score averaged by observation. Every result that is less than 5% smaller than *sota-BNSL*'s score is boldfaced.

dataset	**sota**	**qds$_{\epsilon_{0.9}}$** (%)	**qds$_{\epsilon_{0.75}}$** (%)	**qds$_{\epsilon_{0.5}}$** (%)
20ng	-144.22	**-0.56**	**-1.83**	**-4.16**
adult	-13.09	**-0.18**	**-0.10**	**-3.56**
book	-35.93	**-0.41**	**-0.98**	**-3.64**
covertype	-13.89	**-0.10**	**-0.96**	-10.37
kddcup	-2.40	**-0.26**	**-0.84**	**-3.40**
msweb	-9.90	**-0.02**	**-0.02**	**-0.42**
plants	-13.22	**-2.25**	-6.93	-19.80
r52	-97.44	**-0.60**	**-1.58**	-5.09
andes	-93.28	**-0.48**	-6.21	-16.52
hailfinder	-49.84	**-0.05**	**-2.67**	-10.06
hepar2	-32.63	**-0.28**	**-1.33**	**-3.15**
link	-221.50	**-0.01**	**-0.25**	-16.14
munin1	-43.49	**-0.06**	**-0.17**	-10.05
munin2	-171.82	**-0.02**	**-0.02**	**-1.83**
munin3	-172.44	**0.00**	**0.00**	**-0.89**
munin4	-194.86	**-0.03**	**-0.03**	**-3.09**
pathfinder	-28.59	**-0.14**	**-0.11**	**-4.30**
win95pts	-9.31	**0.00**	**-1.00**	-8.72

($\epsilon_{0.75}$). This is also observed with $\epsilon_{0.5}$ for datasets that contain a lot of very strong one to one relationships as 20 newsgroup, kddcup 2000, msweb, munin2-4, *etc.*
Table 3 and Table 4 show a significant decrease in computational time and model complexity for *qds-BNSL*, which consistently intensifies when ϵ gets larger.

In the best cases, we have both a very small decrease in BIC score, and an important decrease in computational time, *e.g.* the algorithm *qds-BNSL* with $\epsilon = \epsilon_{0.5}$ is 43% faster for msweb, and 61% for munin 3, while producing graphs that score less that 1% below *sota-BNSL*. If we allow a 5% score decrease, *qds-BNSL* can be up to around 70% faster (20 newgroups, kddcup 2000, hepar2, pathfinder).
Even when only 25% of the variables are eliminated by the screening ($\epsilon = \epsilon_{0.75}$), we observe important decreases in computational time: more than 30% less for 20 newsgroup, covertype, kddcup 2000, reuters 52, hepar 2, munin 3 for scores that stay in the 5% tolerance range from *sota-BNSL*'s score. These results confirm the complexity analysis of the previous section, in which we supposed that the screening phase had a very small computational cost compared to the standard structure learning phase.

Graphs learnt by *qds-BNSL* are consistently less complex than those learnt by *sota-BNSL*. Several graphs learnt with $\epsilon_{0.5}$-qd screening have more than 25% less

Table 3: Computation time (seconds). Every result that corresponds to a BIC score less than 5% smaller than *sota-BNSL*'s score is boldfaced.

dataset	**sota** (*seconds*)	**qds$_{\epsilon_{0.9}}$** (%)	**qds$_{\epsilon_{0.75}}$** (%)	**qds$_{\epsilon_{0.5}}$** (%)
20ng	$19,521$	**-8.98**	**-38.35**	**-70.04**
adult	$1,02$	**-6.61**	**-22.03**	**-61.20**
book	$3,851$	**-11.80**	**-23.68**	**-57.97**
covertype	540	**-10.53**	**-36.52**	-72.67
kddcup	$1,531$	**-6.91**	**-37.40**	**-70.79**
msweb	$2,711$	**7.23**	**5.65**	**-43.18**
plants	228	**-17.10**	-39.60	-74.94
r52	$17,194$	**-18.62**	**-41.81**	-74.75
andes	792	**-4.43**	-30.40	-70.93
hailfinder	47	**-0.09**	**-22.81**	-57.69
hepar2	76	**-3.60**	**-40.43**	**-68.20**
link	$6,668$	**-10.77**	**-12.40**	-66.32
munin1	497	**6.18**	**-21.44**	-62.68
munin2	$7,093$	**-20.46**	**-21.66**	**-43.68**
munin3	$10,438$	**-34.88**	**-34.96**	**-61.39**
munin4	$7,774$	**-4.29**	**-0.98**	**-49.15**
pathfinder	221	**-12.97**	**-34.42**	**-69.36**
win95pts	114	**-2.98**	**-29.41**	-66.41

Table 4: Model complexity (number of arcs). Every result that corresponds to a BIC score less than 5% smaller than *sota-BNSL*'s score is boldfaced.

dataset	**sota**	**qds$_{\epsilon_{0.9}}$** (%)	**qds$_{\epsilon_{0.75}}$** (%)	**qds$_{\epsilon_{0.5}}$** (%)
20ng	3136	**-4.50**	**-14.89**	**-31.89**
adult	371	**3.23**	**7.01**	**-13.75**
book	1296	**-4.55**	**-9.49**	**-24.92**
covertype	337	**-0.89**	**-11.28**	-37.69
kddcup	217	**-1.38**	**-13.36**	**-31.80**
msweb	464	**-0.86**	**-0.65**	**2.37**
plants	291	**-8.93**	-18.56	-41.58
r52	2713	**-3.65**	**-9.14**	-25.14
andes	336	**-0.89**	**-7.14**	-22.92
hailfinder	64	**-1.56**	**6.25**	-15.62
hepar2	92	**-3.26**	**-21.74**	**-30.43**
link	1146	**-1.83**	**-0.44**	-22.43
munin1	208	**0.00**	**0.96**	-9.62
munin2	879	**0.00**	**0.00**	**-13.31**
munin3	898	**0.00**	**0.00**	**-7.80**
munin4	903	**0.00**	**0.00**	**-8.53**
pathfinder	161	**-4.35**	-8.70	**-24.22**
win95pts	115	**0.00**	**-0.87**	-12.17

arcs while still scoring less than 5% below state of the art: 20 newsgroups, book, kddcup 2000, hepar 2 and pathfinder.

6 DISCUSSION

We have seen that both in theory and in practice, the quasi-determinism screening approach enables a decrease in computational time and complexity for a small decrease in graph performance. This tradeoff is all the more important that there actually are strong one to one relationships in the data, that can be detected during the screening phase, thus enabling a decrease in the number of variables to be considered by the state of the art structure learning algorithm during the second phase of Algorithm 2.
Optimal cases for this algorithm take place when we get $n_r(\epsilon)$ significantly smaller than n for ϵ reasonably small (this is what happens for datasets 20 newsgroup, munin2-4 and webkb among others).
We have also tested our algorithm on industrial descriptive metadatasets, for which most of the variables possess empirically-deterministic parents: in this case we have $n_r(\epsilon = 0)$ very small with respect to n (typically $n_r(\epsilon = 0) \approx 2 - 3$ for $n \approx 80$). This is a very specific case (data stored in relational databases in almost perfect third normal form), which is however interesting as it is highly advantageous for *qds-BNSL*, which can be up to 20 times faster than state of the art for no performance loss.

Our main research perspective is to be able to anticipate how good the tradeoff may be before running any algorithm all the way through, saving us from trying *qds-BNSL* on datasets in which there are absolutely no strong one to one relationships, and enabling us to choose an optimal value of ϵ on datasets for which there is a lot of potential for computational time win with controlled performance loss.
The bound presented in Proposition 4 concerns the MLL score and is far from tight in practice. However, if we could find a tight bound on the BIC score of the graphs generated by *qds-BNSL* (that should depend notably on $K = \max_{1 \leq i \leq n} |Val(X_i)|$), it would be much easier to estimate the most promising value of ϵ.

Besides, we still have potential to improve the *qds-BNSL* algorithm, by paralellizing the computation of $\mathbb{H}^D$, and implementing it in `C` instead of `R`.

Finally, we have some insights on ways to generalize our quasi-determinism 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 therefore use techniques that detect determinism in a broader sense than only one to one, 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), 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: instead of considering the information-theoretic quantity $H^D(X|Y)$ to describe the strength of the relationship $Y \to X$, one could choose $\frac{H^D(X|Y)}{H^D(X)}$, which represents the proportion of X's entropy that is explained by Y. Moreover, $\frac{H^D(X|Y)}{H^D(X)} \leq \epsilon$ can be rewritten as $\frac{MI^D(X,Y)}{H(X)} \geq 1-\epsilon$, which gives another lighting to quasi-determinism screening: for a given variable X, this comes down to finding a variable Y such that $MI^D(X,Y)$ is high. This is connected to the idea of Chow and Liu (1968), and later Cheng et al. (1997), for whom pairwise empirical mutual information is central. This alternate definition of $\epsilon-$quasi-determinism does not change the algorithms and complexity considerations described in section 4. Lastly, we could consider other definitions of entropy as the ones presented by Rényi et al. (1961).

References

Bouckaert, R. (1995). *Bayesian belief networks: from inference to construction.* 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*.

El Kaed, C., Leida, B., and Gray, T. (2016). Building management insights driven by a multi-system semantic representation approach. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 520–525. IEEE.

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.

Koo, D. D., Lee, J. J., Sebastiani, A., and Kim, J. (2016). An internet-of-things (iot) system development and implementation for bathroom safety enhancement. *Procedia Engineering*, 145:396–403.

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 algorithms. *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.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

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. *Proceedings of the 28th conference on Uncertainty in artificial intelligence*.

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 *Fifth IEEE international conference on Data Mining*, 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, pages 2186–2191.

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.