


# A Pre-Screening Approach for Faster Bayesian Network Structure Learning

Thibaud Rahier<sup>1</sup><sup>[0000-0002-0886-1249]</sup>, Sylvain Marié<sup>2</sup><sup>[0000-0002-5929-1047]</sup>,  
and Florence Forbes<sup>3</sup><sup>[0000-0003-3639-0226]</sup>

<sup>1</sup> Criteo AI Lab, 4 rue des Méridiens, 38130 Echirolles, France `{f.last}@criteo.com`

<sup>2</sup> Schneider Electric Industries, 160, avenue des Martyrs, 38000 Grenoble, France  
`{first.last}@se.com`

<sup>3</sup> INRIA, 655 Av. de l'Europe, 38330 Montbonnot-Saint-Martin, France  
`{first.last}@inria.fr`

**Abstract.** Learning the structure of Bayesian networks from data is a NP-Hard problem that involves optimization over a super-exponential sized space. Still, in many real-life datasets a number of the arcs contained in the final structure correspond to strongly related pairs of variables and can be identified efficiently with information-theoretic metrics. In this work, we propose a meta-algorithm to accelerate any existing Bayesian network structure learning method. It contains an additional arc pre-screening step allowing to narrow the structure learning task down to a subset of the original variables, thus reducing the overall problem size. We conduct extensive experiments on both public benchmarks and private industrial datasets, showing that this approach enables a significant decrease in computational time and graph complexity for little to no decrease in performance score.

**Keywords:** Bayesian networks · Structure learning · Information theory · Conditional entropy · Determinism · Functional relations · Screening.

## 1 Introduction

Bayesian networks are probabilistic graphical models that present interest both in terms of knowledge discovery and density estimation. Learning Bayesian networks 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 (BNSL) from observational data. Algorithms fall under two main categories: *constraint-based* and *score-based*.

Constraint-based structure learning algorithms rely on testing for conditional independence relations that hold in the data in order to reconstruct a Bayesian network encoding these independence relations. The *PC* algorithm by Spirtes et al. (2000) was the first practical application of this idea, followed by increasingly optimized approaches such as the *fast incremental association* (Fast-IAMB) algorithm.

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 with  $n$  nodes is of order  $2^{\mathcal{O}(n^2)}$ , which prevents exhaustive search when  $n$  is typically larger than 30. Most of the score-based algorithms used in practice therefore rely on heuristics, as the original approach from Cooper and Herskovits (1992) which assumes a prior ordering of the variables is known, 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, increasingly pushing the limits of state-of-the art in BNSL: some based on the search for an optimal ordering (Teyssier and Koller, 2005; Chen et al., 2008), others on optimizing the search task in accordance to a given score (Scanagatta et al., 2015; Nie et al., 2016), using integer programming (Cussens, 2011) or bio-inspired optimization heuristics (Kareem and Okur, 2019, 2021). See Scutari et al. (2019) for a recent review.

Meanwhile, data itself may contain determinism, for example in the fields of cancer risk identification (de Moraes et al., 2008) or nuclear safety (Mabrouk et al., 2014). Data is also increasingly collected and generated by software systems whether in social networks, smart buildings, smart grids, 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 semantic data models (El Kaed et al., 2016) where the same entity may be described with several attributes. It is therefore now common to find deterministically related variables 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 first remind the background of Bayesian network structure learning (Section 2) and bring forward the following contributions:

- we state some theoretical results bridging the gap between the notion of determinism and Bayesian network scoring (Section 3),
- we propose and study the complexity of the *quasi-determinism screening BNSL* (qds-BNSL) meta-algorithm, whose aim is to accelerate any existing BNSL algorithm by reducing the learning problem to a subset of the original variables via the detection of strong arcs (Section 4),
- we conduct experiments on both public benchmarks and private industrial datasets, demonstrating empirically that our meta-algorithm indeed accelerates the overall BNSL procedure with very low performance loss and also leads to sparser and therefore more interpretable graphs (Section 5).

## 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  $I \subset \llbracket 1, n \rrbracket$ , 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, we suppose that  $D$  is 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  $\cdot^D$  exponent (*e.g.*  $p^D$  refers to the empirical distribution with respect to  $D$ ). Finally,  $D_I$  refers to the restriction of  $D$  to the observations of  $\mathbf{X}_I$ .

A Bayesian network is an object  $\mathcal{B} = (G, \theta)$  where (1)  $G = (V, A)$  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  $V = \llbracket 1, n \rrbracket$  where each node  $i \in V$  is associated with the random variable  $X_i$ , and  $\pi^G(i) = \{j \in V \text{ s.t. } (j, i) \in A\}$  is the set of  $i$ 's parents in  $G^4$  and (2)  $\theta = \{\theta_i\}_{i \in V}$  is a set of parameters. Each  $\theta_i$  defines the local conditional distribution of  $X_i$  given its parents in the graph,  $P(X_i | \mathbf{X}_{\pi(i)})$ . 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-descendants given its parents*.

## 2.2 Score-based approach to Bayesian network structure learning

For a given *scoring* function  $s : DAG_V \rightarrow \mathbb{R}$ , where  $DAG_V$  is the set of all possible DAG structures with node set  $V$ , score-based BNSL aims at solving the following combinatorial optimization problem:

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

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 (Koller and Friedman, 2009): the size of  $DAG_V$  is said to be super-exponential in  $|V|$ . Most scoring functions used in practice are based on the likelihood function. The most straightforward being the *max log-likelihood* score, that we now present.

*The max log-likelihood (MLL) score* Let  $l_D(\theta) = \log(p_\theta(D))$  be the log-likelihood of the set of parameters  $\theta$  given the dataset  $D$ . For a given DAG structure  $G \in DAG_V$ , we define the MLL score of  $G$  with respect to  $D$  as:

$$s_D^{MLL}(G) = \max_{\theta \in \Theta_G} l_D(\theta).$$

where  $\Theta_G$  is the set of all  $\theta$ 's such that  $\mathcal{B} = (G, \theta)$  is a well defined Bayesian network. The MLL score is very straightforward and intuitive, but it favors denser

<sup>4</sup> The exponent  $G$  may be dropped for clarity when the referred graph is obvious from context

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:  $s_D^{MLL}(G_1) \leq s_D^{MLL}(G_2)$ . This problem is generally solved by using a score that induces a goodness-of-fit versus complexity tradeoff, such as BIC (Schwarz et al., 1978), which is a penalized version of the MLL score, or BDe (Heckerman et al., 1995), which is derived from the marginalization of the likelihood, which implicitly penalizes the model’s complexity through a Dirichlet prior on the parameters. In this paper, we will use the BDe score to evaluate a BN structure’s quality, as it is done in several papers as Teyssier and Koller (2005), or Nie et al. (2016). This score is known to be a good indicator of generalization performance, which is what we are aiming to optimize. In the following sections, we propose to look for such a solution by constructing sparse graphs with minimal MLL.

### 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,

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

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  $D$*

$G \in \text{DAG}_V$  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  $X_1, \dots, X_n$  observed in a dataset  $D$ ,

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

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

The next proposition follows then straightforwardly. We remind that a tree is a DAG in which each node has exactly one parent, except for 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):

$$s_D^{MLL}(T) = \max_{G \in DAG_V} s_D^{MLL}(G).$$

It is worth noticing that complete DAGs also maximize the MLL score. The main interest of Proposition 1 resides in the fact that, under the (strong) assumption that a deterministic tree exists, 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 assumption 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  $T_1, \dots, T_p$  are  $p$  disjoint deterministic trees wrt  $D_{V_{T_1}}, \dots, D_{V_{T_p}}$  respectively and 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  $F$  with respect to  $D$ , we define  $\mathcal{R}(F) = \{i \in V \mid \pi^F(i) = \emptyset\}$  the set of  $F$ 's roots (the union of the roots of each of its trees).

**Proposition 2** Suppose  $F$  is a deterministic forest wrt  $D$ . Let  $G_{\mathcal{R}(F)}^*$  be a solution of the BNSL optimization problem (1) for  $\mathbf{X}_{\mathcal{R}(F)}$  and the MLL score i.e.

$$s_{D_{\mathcal{R}(F)}}^{MLL}(G_{\mathcal{R}(F)}^*) = \max_{G \in DAG_{\mathcal{R}(F)}} s_{D_{\mathcal{R}(F)}}^{MLL}(G).$$

Then,  $G^* = F \cup G_{\mathcal{R}(F)}^*$  is a solution of (1) for  $\mathbf{X}$ , i.e.

$$s_D^{MLL}(G^*) = \max_{G \in DAG_V} s_D^{MLL}(G).$$

As opposed to Proposition 1, the assumptions of Proposition 2 are always formally verified: if there is no determinism in the dataset  $D$ , then  $\mathcal{R}(F) = V$ , and every tree  $T_k$  is formed of a single root node. In that case, solving problem (1) for  $G_{\mathcal{R}(F)}^*$  is the same as solving it for  $G^*$ . Of course, we are interested in the case where  $\mathcal{R}(F) < n$ , as this enables us to focus on a smaller structure learning problem while still having the guarantee to learn the optimal Bayesian network with regards to the MLL score.

As seen in Section 2, the main issue with the MLL score is that it favors complete graphs. However, a deterministic forest  $F$  containing  $p$  trees is very sparse ( $n - p$  arcs), and even if the graph  $G_{\mathcal{R}(F)}^*$  is dense, the graph  $G^* = F \cup G_{\mathcal{R}(F)}^*$  may still satisfy sparsity conditions.

## 4 Structure learning with quasi-determinism screening

### 4.1 Quasi-determinism

When it comes to BNSL 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 narrow the structure learning problem down 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, one does not always 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  $\epsilon$ : 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  $\epsilon$  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  $X_i \rightarrow X_j$  is  $\epsilon$ -*qd* wrt  $D$  iff  $H^D(X_j|X_i) \leq \epsilon$ .

It has been seen in Proposition 2 that a deterministic forest is the subgraph of an optimal DAG with respect to the MLL score, while still satisfying sparsity conditions. Such a forest is therefore very promising with regards to the fit-complexity tradeoff (typically evaluated by scores such as BDe or BIC).

Combining this intuition with the  $\epsilon$ -*qd* criteria presented in Definition 4, we propose the quasi-determinism screening approach to BNSL, defined in the next subsections. An alternate definition with a relative  $\epsilon$  is proposed in Section 6.

## 4.2 Quasi-determinism screening algorithm

Algorithm 1 details how to find the simplest  $\epsilon$ -qd forest  $F_\epsilon$  from a dataset  $D$  and a threshold  $\epsilon$ . Here *simplest* refers to the complexity in terms of number of parameters, which in the case of categorical variables corresponds to the number of states: for each variable  $X_i$  that has at least one  $\epsilon$ -qd parent, we select the one that has the lowest number of states from the set of all potential  $\epsilon$ -qd parents  $\pi_\epsilon(i)$  (line 11).

This algorithm takes for input  $D$  (a dataset containing  $M$  *i.i.d* realizations of  $\mathbf{X}$ ) and  $\epsilon$  (a threshold for quasi-determinism). The routine on lines 4-9 makes sure no

---

### Algorithm 1 Quasi-determinism screening (qds)

---

**Input:**  $D$ ,  $\epsilon$

- 1: Compute empirical conditional entropy matrix  $\mathbb{H}^D = (H^D(X_i|X_j))_{1 \leq i, j \leq n}$
- 2: **for**  $i = 1$  to  $n$  **do**
- 3:     compute  $\pi_\epsilon(i) = \{j \in \llbracket 1, n \rrbracket \setminus \{i\} \mid \mathbb{H}_{ij}^D \leq \epsilon\}$
- 4: **for**  $i = 1$  to  $n$  **do**
- 5:     **if**  $\exists j \in \pi_\epsilon(i)$  *s.t.*  $i \in \pi_\epsilon(j)$  **then**
- 6:         **if**  $\mathbb{H}_{ij}^D \leq \mathbb{H}_{ji}^D$  **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**
- 11:      $\pi_\epsilon^*(i) \leftarrow \underset{j \in \pi_\epsilon(i)}{\operatorname{argmin}} |Val(X_j)|$  (select one index in case of tie)
- 12: Compute forest  $F_\epsilon = (V_{F_\epsilon}, A_{F_\epsilon})$  where  $V_{F_\epsilon} = \llbracket 1, n \rrbracket$  and  $A_{F_\epsilon} = \{(\pi_\epsilon^*(i), i) \mid i \in \llbracket 1, n \rrbracket \text{ s.t. } \pi_\epsilon^*(i) \neq \emptyset\}$

**Output:**  $F_\epsilon$

---

cycle is introduced by the screening phase, and guarantees the next proposition holds (ensuring that Algorithm 1 is indeed well defined):

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

## 4.3 Learning Bayesian networks using quasi-determinism screening

We now present Algorithm 2, which uses quasi-determinism screening to accelerate Bayesian network structure learning. This algorithm takes the following input:  $D$  (a dataset containing  $M$  realizations of  $\mathbf{X}$ ),  $\epsilon$  (a threshold for quasi-determinism) and **ref-BNSL** (a BNSL baseline algorithm, taking for input a dataset, and returning a Bayesian network structure). The extension of the def-

---

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

---

**Input:**  $D, \epsilon, \text{ref-BNSL}$

- 1: Compute  $F_\epsilon$  by running **Algorithm 1** with input  $D$  and  $\epsilon$
- 2: Identify  $R(F_\epsilon) = \{i \in \llbracket 1, n \rrbracket \mid \pi^{F_\epsilon}(i) = \emptyset\}$ , the set of  $F_\epsilon$ 's roots.
- 3: Compute  $G_{R(F_\epsilon)}^*$  by running **ref-BNSL** on  $D_{R(F_\epsilon)}$
- 4:  $G_\epsilon^* \leftarrow F_\epsilon \cup G_{R(F_\epsilon)}^*$

**Output:**  $G_\epsilon^*$

---

inition of determinism to quasi-determinism (Definition 3) prevents us to have ‘hard’ guarantees as those presented in Proposition 2. However, we are able to explicit bounds for the MLL score of a graph  $G_\epsilon^*$  returned by Algorithm 2, as stated in the following Proposition.

**Proposition 4** *Let  $\epsilon, D$  and **ref-BNSL** be rightful input to Algorithm 2, and  $G_\epsilon^*$  the associated output.*

*Then, if **ref-BNSL** is exact (i.e. always returns an optimal solution) with respect to the MLL score, we have the following lower bound for  $s_D^{MLL}(G_\epsilon^*)$ :*

$$s_D^{MLL}(G_\epsilon^*) \geq \left( \max_{G \in \text{DAG}_V} s_D^{MLL}(G) \right) - Mn\epsilon.$$

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

$$s_D^{MLL}(G_\epsilon^*) \xrightarrow{\epsilon \rightarrow 0} \max_{G \in \text{DAG}_V} s_D^{MLL}(G).$$

In other words,  $\epsilon \mapsto s_D^{MLL}(G_\epsilon^*)$  is continuous in 0, and Proposition 4 generalizes Proposition 2.

Algorithm 2 is promising, notably if for small  $\epsilon$  we have  $|R(F_\epsilon)|$  significantly smaller than  $n$ . In that case, **ref-BNSL**, that only has to be run on  $D_{R(F_\epsilon)}$ , can be expected to be much faster and more accurate than if it is run on the entire dataset  $D$ .

#### 4.4 Complexity analysis

*Complexity of baseline BNSL learning algorithms* The number of possible DAG structures being super exponential in the number of nodes, BNSL 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 **ref-BNSL** be a reference Bayesian network structure learning algorithm and  $C_{\text{ref}}(M, n)$  be its complexity.  $C_{\text{ref}}(M, n)$  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  $D$ ), which implies  $M \frac{n(n-1)}{2}$  operations.
2. lines 4-9:  $\mathcal{O}(n^2)$ . Going through  $\mathbb{H}^D$  once.
3. lines 10-12:  $\mathcal{O}(n^2)$ . Going through  $\mathbb{H}^D$  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  $F_\epsilon$  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:  $C_{ref}(M, n_r(\epsilon))$ . Run of **ref**-BNSL on reduced dataset  $D_{R(F_\epsilon)}$  with  $n_r(\epsilon)$  columns.

This yields  $C_{Alg2}(M, n) = \mathcal{O}(Mn^2) + C_{ref}(M, n_r(\epsilon))$ .

We are interested in how much it differs from  $C_{ref}(M, n)$ , which depends mainly on:

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

$C_{ref}(M, n)$  is known to be typically exponential in  $n$  for the best exact structure learning algorithms, as those presented by Silander and Myllymäki (2006) or Cussens (2011), and it is expected to be significantly larger than  $\mathcal{O}(Mn^2)$  for high-performing heuristics. We therefore expect an important decrease in computational time when running Algorithm 2 compared to its baseline version, as long as  $n_r(\epsilon)$  is sufficiently smaller than  $n$ . In the next section, we run a reference structure learning algorithm and Algorithm 2 on benchmark datasets in order to confirm this intuition.

## 5 Experiments

### 5.1 Experimental setup

*Data* Table 1 summarizes the data used in our experiments. We considered the largest open-source categorical datasets among those presented<sup>5</sup> by Davis and Domingos (2010) and available on the UCI repository (Dheeru and Karra Taniskidou, 2017): `20_newsgroup`, `adult`, `book`, `coverttype`, `kddcup_2000`, `msnbc`, `msweb`,

<sup>5</sup> <http://alchemy.cs.washington.edu/papers/davis10a/>

plants, reuters-52 and uscensus. Moreover, as it was done by Scanagatta et al. (2016), we chose the largest Bayesian networks available in the literature<sup>6</sup>, for each of which we simulated 10000 observations: andes, hailfinder, hepar 2, link, munin 1-4, pathfinder and win95pts.

We also include two industrial datasets containing descriptive metadata on which we have privileged access, priv-metadata 1 and priv-metadata 2.

**Table 1.** Datasets presentation

name	short name	$n$	$M$
20 newsgroups	20ng	930	11293
adult	adult	125	36631
book	book	500	8700
covertypes	covertypes	84	30000
kddcup 2000	kddcup	64	180092
msnbc	msnbc	17	291326
msweb	msweb	294	29441
plants	plants	69	17412
reuters 52	r52	941	6532
uscensus	uscensus	68	2458285
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
priv-metadata 1	priv-meta1	43	1000
priv-metadata 2	priv-meta2	41	1000

*Programming details and choice of ref-BNSL* Most of the code associated with this project was done in R, enabling an optimal exploitation of the `bnlearn` package from Scutari (2010), which is a very good reference among open-source packages dealing with Bayesian networks structure learning.

We need a BSNL algorithm to obtain a baseline performance. After carefully evaluating several algorithms implemented in the `bnlearn` package, we chose to use *Greedy Hill Climbing with random restarts and a tabu list*, as it consistently outperformed other built-in algorithms both in time and score, in addition to

<sup>6</sup> <http://www.bnlearn.com/bnrepository/>

being also used as a benchmark algorithm in the literature, notably by Teyssier and Koller (2005). In this section, we refer to this algorithm as **ref**-BNSL.

*Choice of  $\epsilon$  for **qds**-BNSL* An approach to choosing  $\epsilon$  in the case of the **qds**-BNSL algorithm is to pick values for  $n_r(\epsilon)$ , and manually find the corresponding values for  $\epsilon$ . For a given dataset and  $\rho \in [0, 1]$ , we define  $\epsilon_\rho = n_r^{-1}(\lfloor \rho n \rfloor)$ . In other words,  $\epsilon_\rho$  is the value of  $\epsilon$  for which the number of roots of the qd forest  $F_\epsilon$  represents a proportion  $\rho$  of the total number of variables (more details in Rahier (2018)). The computation of  $\epsilon_\rho$  is not problematic: once  $\mathbb{H}^D$  is computed and stored, evaluating  $n_r(\epsilon)$  is done in constant time, and finding one of  $n_r(\cdot)$ 's quantiles is doable in  $\mathcal{O}(\log(n))$  operations (dichotomy), which is negligible compared to the overall complexity of the screening. In the case of the **priv-metadata** datasets, choosing  $\epsilon = 0$  leads to a dramatic decrease of the number of variables that are considered by the baseline algorithm, since these datasets contain several truly deterministic relationships by design.

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

- BDe score of Section 2 with a uniform prior and equivalent sample size (ESS) equal to 5, inspired from Teyssier and Koller (2005) and referred to as BDeu.
- Number of arcs of the learned Bayesian network.

The BDeu score naturally penalizes overly complex models (in terms of number of parameters), it is however interesting to look at the number of arcs, as it is a straightforward way to evaluate how complex a Bayesian network appears to a human expert (and thus how interpretable this structure is).

- Computing time  $t_{run}$  (all algorithms were run on the same machine).
- It is essential to remark that **ref**-BNSL is used both to obtain a baseline performance and inside **qds**-BNSL. In both cases, it is run with the same settings until convergence. The comparison of computing times is thus fair.

We present the obtained results for our selected baseline algorithm **ref**-BNSL, and 3 versions of **qds**-BNSL. For each dataset, we selected  $\epsilon \in \{\epsilon_{0.9}, \epsilon_{0.75}, \epsilon_{0.5}\}$ , corresponding to a restriction of **ref**-BNSL to 90%, 75% and 50% of the original variables respectively (for the **priv-metadata** datasets, these three choices of  $\epsilon$  are merged into the single choice  $\epsilon = 0$ , which results in a decrease of more than 50% of the original variables).

The results are shown in Table 2, one group of columns per evaluation criterion, and each value is the median of 10 runs with different seeds. In each table, the median value of the criterion is displayed for **ref**-BNSL (**ref**), and the relative difference is displayed for the three versions of **qds**-BNSL we consider (**qds** $_{\epsilon_{0.9}}$ , **qds** $_{\epsilon_{0.75}}$  and **qds** $_{\epsilon_{0.5}}$ ).

## 5.2 Results

*Score* It appears in Table 2 that the decrease in BDeu score is smaller than 5% for all the considered datasets when 90% of the variables remain after the

**Table 2.** For algorithms **ref**, **q<sub>0.5</sub>**, **q<sub>0.75</sub>** and **q<sub>0.9</sub>**, and benchmark datasets, we display the Bayesian Network’s (1) BDeu score averaged by observation, (2) learning time (including prescreening) and (3) number of arcs. Every result that corresponds to a BDeu score less than 5% smaller than **ref**-BNSL’s score is boldfaced.

dataset	BDeu score				Computation time				Number of arcs			
	<b>ref</b>	<b>q<sub>0.9</sub></b> (%)	<b>q<sub>0.75</sub></b> (%)	<b>q<sub>0.5</sub></b> (%)	<b>ref</b> (s)	<b>q<sub>0.9</sub></b> (%)	<b>q<sub>0.75</sub></b> (%)	<b>q<sub>0.5</sub></b> (%)	<b>ref</b> (nb)	<b>q<sub>0.9</sub></b> (%)	<b>q<sub>0.75</sub></b> (%)	<b>q<sub>0.5</sub></b> (%)
20ng	-143	<b>-0.7</b>	<b>-2.1</b>	<b>-4.8</b>	21495	-1.6	-43	-73	3136	-4.5	-15	-32
adult	-13	<b>-0.2</b>	<b>-0.1</b>	<b>-4.0</b>	102	-6.6	-22	-61	371	+3.2	+7.0	-14
book	-35	<b>-0.8</b>	<b>-1.7</b>	<b>-4.6</b>	7600	-24	-40	-71	2196	-11	-19	-40
covertype	-14	<b>-0.2</b>	<b>-1.2</b>	-12	565	-6.8	-33	-71	337	-0.9	-11	-38
kddcup	-2.4	<b>-0.3</b>	<b>-1.0</b>	<b>-3.8</b>	2167	-11	-33	-74	285	-5.3	-19	-39
msnbc	-6.2	<b>-0.1</b>	<b>-2.6</b>	<b>-4.6</b>	252	-21	-61	-86	102	-7.8	-33	-64
msweb	-9.8	<b>+0.0</b>	<b>-0.1</b>	<b>-1.0</b>	4701	-6.3	-9.9	-55	1264	-2.5	-3.6	-35
plants	-13	<b>-2.6</b>	-7.6	-21	455	-47	-62	-84	320	-6.2	-18	-42
r52	-95	<b>-0.8</b>	<b>-2.0</b>	-6.1	18630	-14	-38	-77	2713	-3.6	-9.1	-25
uscensus	-23	<b>-0.3</b>	<b>-1.8</b>	-10	21782	-0.4	-32	-78	220	-10	-20	-38
andes	-93	<b>-0.5</b>	-6.2	-17	898	-2.2	-27	-70	336	-0.9	-7.1	-23
hailfinder	-50	<b>-0.1</b>	<b>-2.7</b>	-10	46	-5.3	-17	-55	64	-1.6	+6.2	-16
hepar2	-33	<b>-0.3</b>	<b>-1.4</b>	<b>-3.2</b>	76	-4.0	-43	-70	92	-3.3	-22	-30
link	-216	<b>+0.1</b>	<b>+1.1</b>	-17	7240	-12	-11	-61	1146	-1.8	-0.4	-22
munin1	-41	<b>-0.1</b>	<b>-0.2</b>	-9.9	497	-7.4	-17	-59	208	+0.0	+1.0	-9.6
munin2	-172	<b>-0.0</b>	<b>-0.0</b>	<b>-1.8</b>	7093	-20	-22	-44	879	+0.0	+0.0	-13
munin3	-165	<b>+0.0</b>	<b>+0.0</b>	<b>-1.1</b>	11558	-37	-29	-54	898	+0.0	+0.0	-7.8
munin4	-186	<b>-0.0</b>	<b>-0.0</b>	<b>-3.9</b>	8550	-7.9	-13	-39	903	+0.0	+0.0	-8.5
pathfinder	-27	<b>-0.7</b>	<b>-0.7</b>	<b>-4.9</b>	231	-14	-35	-69	161	-4.3	-8.7	-24
win95pts	-9.2	<b>+0.1</b>	<b>-1.1</b>	-9.2	132	-6.0	-31	-69	115	+0.0	-0.9	-12
priv-meta1	-8.72	<b>+1.1</b>	<b>+1.1</b>	<b>+1.1</b>	13794	-99	-99	-99	70	-41	-41	-41
priv-meta2	-8.72	<b>+12.5</b>	<b>+12.5</b>	<b>+12.5</b>	4346	-99	-99	-99	102	-59	-59	-59

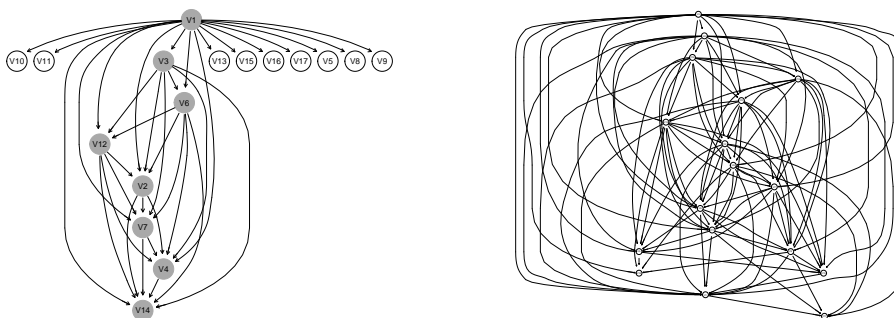
pre-screening (**qds** <sub>$\epsilon_{0.9}$</sub> ), and for most of them when 75% of the variables remain (**qds** <sub>$\epsilon_{0.75}$</sub> ). This is also observed with  $\epsilon_{0.5}$  for datasets that contain a lot of very strong pairwise relationships as **kddcup**, **msweb**, or **munin 2-4**. For **priv-metadata** datasets, our approach increases the score (slightly for **priv-metadata1** and of more than 12% for **priv-metadata2**).

*Computing time* Table 2 shows a significant decrease in computational time for **qds**-BNSL, which is all the more important as  $\epsilon$  is large. In the best cases, we have both a very small decrease in BDeu score, and an important decrease in computational time. We suspect that this is also due to the presence of many strong pairwise relationships. For example, the algorithm **qds**-BNSL with  $\epsilon = \epsilon_{0.5}$  is 55% faster for **msweb**, and 54% for **munin 3**, while implying only around 1% decrease in score compared to **ref**-BNSL. If we allow a 5% score decrease, **qds**-BNSL can be up to 70% faster (**20 newgroups**, **book**, **msnbc**, **kddcup**, **hepar2**, **pathfinder**). On the industrial datasets this computational time decrease is astonishing: it is of

more than 2 **orders of magnitude** in the case of `priv-metadata 1` and more than 3 **orders of magnitude** for `priv-metadata 2`<sup>7</sup>. 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.

*Complexity* As showed by Table 2, Bayesian networks learned with `qds-BNSL` are consistently less complex than those learned with `ref-BNSL`. Several graphs learned with `qdsε0.5` are more than 30% sparser while still scoring less than 5% below the baseline algorithm: 20 `newsgroups`, `book`, `kddcup 2000`, `msnbc`, `msweb` and `hepar 2`.

Figure 1 displays two Bayesian networks learned on the `msnbc` dataset.



**Fig. 1.** Bayesian network learned on `msnbc` with `qds-BNSL`, left (resp. `ref-BNSL`, right). BDeu: -6.48 (resp. -6.19), Nb of arcs: 37 (resp. 102), Running time: 36s (resp. 252s)

They provide an interesting example of the sparsity induced by `qds-BNSL`. After the `qdε0.5`-screening phase, half of the variables (corresponding to the nodes in white) are considered to be sufficiently explained by `V1`. They are therefore not taken into account by `ref-BNSL`, which is run only on the variables corresponding to the nodes in gray (more details in Rahier et al. (2018)).

In the `msnbc` case, this learning problem restriction implies only a small decrease in the final graph’s generalization performance (as seen in BDeu scores), while being 7 times faster to compute and enabling a significantly better readability.

In this processed version of the `msnbc` dataset (Davis and Domingos, 2010), each variable contains a binary information regarding the visit of a given page from the `msnbc.com` website<sup>8</sup>. The Bayesian network displayed in Figure 2 shows in a compact way the influence between the different variables. For instance,

<sup>7</sup> These results were so extreme that they could not be fully captured in the ‘percentage’ format choice of Table 2.

<sup>8</sup> more details: <http://archive.ics.uci.edu/ml/machine-learning-databases/msnbc-ml/msnbc.data.html>

we see that visits of the website’s pages corresponding to nodes in white (*e.g.* ‘weather’ (V8), ‘health’ (V9) or ‘business’ (V11)) are importantly influenced by whether the user has also visited the frontpage (V1). For example, learned parameters show that a user who did not visit the website’s frontpage (V1) is about 10 times more likely to have visited the website’s ‘summary’ page (V13) than a user who did visit the frontpage. Such information is much harder to read from the graph learned with `ref-BNSL` displayed in Figure 1 right.

## 6 Concluding remarks

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 score. This tradeoff is all the more advantageous as there actually are strong pairwise 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 baseline structure learning algorithm during the second phase of Algorithm 2. Optimal cases for this meta-algorithm take place when  $n_r(\epsilon)$  is significantly smaller than  $n$  for  $\epsilon$  reasonably small compared to the variable’s entropies. Among benchmark datasets this is reasonably frequent (*e.g.* `20 newsgroup`, `msnbc`, `munin2-4`, `webkb`), and we argue it is extremely frequent among industrial datasets, as we have shown in our `priv-metadata` datasets which are only a small sample of the kind of datasets in which we can find very strong (even completely deterministic) relations.

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

Our main research perspectives are (1) to understand how one can anticipate how good the score/computation time/complexity trade-off can be before running any algorithm all the way through, saving us from running `qds-BNSL` on datasets in which there are no strong pairwise relationships to be detected, (2) find a principled way to choose  $\epsilon$  and (3) tighten the bound of Proposition 4 and generalize it to the `BDeu` score.

In another directions, 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 pairwise, 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), or more recently Papenbrock et al. (2015) who evaluate 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  $Y \rightarrow X$ , 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 insight 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 original 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.

## Bibliography

- 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.
- Cussens, J. (2011). Bayesian network learning with cutting planes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI'11*, pages 153–160, Arlington, Virginia, United States. AUAI Press.
- 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 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*.
- Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository.
- 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.
- Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3).
- 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.
- Kareem, S. W. and Okur, M. C. (2019). Bayesian network structure learning based on pigeon inspired optimization. *International Journal of Advanced Trends in Computer Science and Engineering*, 8:131–137.
- Kareem, S. W. and Okur, M. C. (2021). Falcon optimization algorithm for bayesian networks structure learning. *Computer Science*, 22(4).



- 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.
- 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*. IOS Press.
- Nie, S., de Campos, C. P., and Ji, Q. (2016). Learning Bayesian networks with bounded tree-width via guided search. In *AAAI*, pages 3294–3300.
- 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.
- Rahier, T., Marie, S., Girard, S., and Forbes, F. (2018). Screening strong pairwise relationships for fast Bayesian network structure learning *2nd Italian-French Statistics Seminar-IFSS*.
- Rahier, T. (2018). Bayesian networks for static and temporal data fusion *Université Grenoble Alpes*, PhD thesis.
- 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. (2010). Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*.
- Scutari, M., Graafland, C., and Gutiérrez, J. (2019). Who learns better bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115.
- Silander, T. and Myllymäki, P. (2006). A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI’06.
- Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press.
- Teyssier, M. and Koller, D. (2005). Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 584–590. AUAI Press.