ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-567

An Efficient Bayesian Network Structure Learning Algorithm in the Presence of Deterministic Relations

Ahmed Mabrouk[1] and **Christophe Gonzales**[2] and **Karine Jabet-Chevalier**[1] and **Eric Chojnacki**[1]

Abstract. Faithfulness is one of the main hypotheses on which rely most Bayesian network (BN) structure learning algorithms. When some random variables are deterministically determined by others, faithfulness is ruled out and classical learning algorithms fail to discover many dependences between variables, hence producing incorrect BNs. Even state-of-the-art algorithms dedicated to learning with deterministic variables prove to be inefficient to discover many dependences/independences. For critical applications, e.g., in nuclear safety, such failure is a serious issue. This paper introduces a new hybrid algorithm, combining a constraint-based approach with a greedy search, that includes specific rules dedicated to deterministic nodes that significantly reduce the incorrect learning. Experiments show that our method significantly outperforms state-of-the-art algorithms.

1 INTRODUCTION

Bayesian networks (BN) [18] are defined by a graphical structure whose nodes represent random variables and by the set of conditional probability tables of each node in the graph given its parents. They encode the joint probability over all the nodes as the product of these conditional probabilities. As such, they enable compact representations of probabilities. In addition, they are supplied with fast inference engines that enable to answer efficiently various types of probabilistic queries (computation of marginal, *a priori*, *a posteriori* probabilities [15, 4], of most probable explanations [16], of maximum *a posteriori* [17], etc.). This explains the reason why they are a very popular framework for reasoning under uncertainty.

In the past twenty years, numerous efforts have been devoted to learn both their graphical structure and the parameters of their conditional probability tables from datasets [2, 9, 10, 19, 21, 23]. Different contexts have led researchers to propose tailored learning algorithms, e.g., algorithms taking into account monotonicity constraints [5]. However, there exist important critical applications for which no current BN learning algorithm proves to be satisfactory. Such a situation, which was actually our starting point, arises in nuclear safety, notably in problems of nuclear accident scenario reconstruction from sensors' partial observations. In this context, BNs and their inference engines seem well-suited for helping Decision Makers make the best decisions to limit as much as possible the consequences of the accident [18]. But learning such BNs from datasets is a complex task due to the presence of deterministic dependences between random variables that essentially represent equations modeling the various phenomena occurring in a damaged nuclear power plant during a severe accident. These deterministic relations rule out the *faithfulness* property on which rely the majority of learning algorithms. In essence, the absence of an arc in the BN implies a conditional independence in the underlying probability distribution. But learning algorithms derive from the dataset independence statements and, then, induce from them the absence of some arcs, hence exploiting the opposite implication. Faithfulness states that both implications hold, i.e., that the absence of an arc is equivalent to a probabilistic conditional independence. When there exist deterministic relations among random variables this equivalence fails to hold and learning algorithms can therefore fail to recover a correct BN structure.

Structure learning with deterministic relations has been tackled in several papers. In [22], it is suggested to remove those random variables that are deterministically determined by others. While this method is simple, it does not work in many cases because this tends to increase so much the sizes of the contingency tables used in the independence tests required to construct the BN's structure that these tests become meaningless. In addition, this can also induce undesirable changes in the independence model (e.g., creating new dependences). Other more sophisticated methods have been provided in the literature: in [20], the authors propose to modify the operators of MMHC [23] to prevent that some arcs adjacent to deterministic nodes be discarded from the BN's graphical structure. But, by doing so, they also add unnecessary arcs. [14] and [13] also either miss some arcs or add unnecessary ones. Missing arcs is a serious failure because the learnt model is inadequate and, in critical applications like nuclear safety, this can have serious consequences. Adding too many arcs is also an issue because it slows down probabilistic inference, which needs to be fast to help Decision Makers in real-time.

The approach we propose to overcome these problems first constructs the skeleton of the BN, i.e., its graphical structure in which arcs are substituted by edges, and, then to convert it into a directed graph which is refined by a greedy search. This kind of technique is known to be very effective [23]. The originality of our algorithm lies in the rules applied in its three steps, that exploit the features of the deterministic relations: i) to discard arcs that should not belong to the BN but that cannot be detected by scoring or independence tests; and ii) to orient optimally the arcs adjacent to deterministic nodes. In addition, our algorithm uses an original way to compute the conditional independences it needs to construct the BN's graphical structure.

The rest of the paper is organized as follows. Section 2 recalls the definitions and notations needed and explains why deterministic relations make structure learning a challenging task. Section 3 presents learning algorithms of the literature suited for deterministic relations. Then, in Section 4, we describe our approach and justify its correctness. Its effectiveness is highlighted through experiments in Section 5. Finally, some concluding remarks are given in Section 6.

[1] Institut de radioprotection et de sûreté nucléaire, France, email: firstname.lastname@irsn.fr
[2] Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7606, LIP6, France, email: Christophe.Gonzales@lip6.fr

2 BAYES NET STRUCTURE LEARNING AND DETERMINISTIC RELATIONS

In all the paper, boldface letters represent sets, and we only consider discrete random variables and complete datasets. Our goal is to learn from datasets the structure of a Bayesian network (BN):

Definition 1 *A BN is a pair* $(\mathcal{G}, \mathbf{\Theta})$ *where* $\mathcal{G} = (\mathbf{V}, \mathbf{A})$ *is a directed acyclic graph (DAG),* $\mathbf{V}$ *represents a set of random variables*[3]*,* $\mathbf{A}$ *is a set of arcs, and* $\mathbf{\Theta} = \{P(X|\mathbf{Pa}(X))\}_{X\in\mathbf{V}}$ *is the set of the conditional probability distributions of the nodes / random variables* X *in* $\mathcal{G}$ *given their parents* $\mathbf{Pa}(X)$ *in* $\mathcal{G}$*. The BN encodes the joint probability over* $\mathbf{V}$ *as:*

$$P(\mathbf{V}) = \prod_{X\in\mathbf{V}} P(X|\mathbf{Pa}(X)). \quad (1)$$

In all the paper, $\mathbf{V}$ will represent the set of nodes / variables of the BNs. By their graphical structure, BNs encode an independence model, i.e., a set of conditional independences between random variables, characterized by d-separation [18]:

Definition 2 *Two nodes* X *and* Y *are said to be d-separated in* $\mathcal{G}$ *by a set of nodes* $\mathbf{Z} \subseteq \mathbf{V}\setminus\{X, Y\}$*, which is denoted by* $X \perp_G Y|\mathbf{Z}$*, if, for every trail (undirected path) linking* X *and* Y *in* $\mathcal{G}$*, there exists a node* S *on the trail such that one of the following conditions holds:*

1. *S has converging arrows on the trail and neither S nor any of its descendants are in $\mathbf{Z}$;*
2. *S does not have converging arrows on the trail and $S \in \mathbf{Z}$.*

Definition 2 induces an alternative definition for BNs: a pair $(\mathcal{G}, \mathbf{\Theta})$ is a BN if each node $X \in \mathbf{V}$ is *probabilistically* independent of its non-descendants in $\mathcal{G}$ given its parents. This property is called the *local Markov property*. In the rest of the paper, we will denote by $X \perp_P Y|\mathbf{Z}$ the probabilistic independence of X and Y given $\mathbf{Z}$, i.e., the case when $P(X|Y, \mathbf{Z}) = P(X|\mathbf{Z})$.

Learning the graphical structure $\mathcal{G}$ of the BN from data consists of learning its independence model, hence the set of conditional independences it models. These are determined by computing either scores (BIC, BDeu, etc.) [9] or statistical independence tests (χ^2, G^2, etc.) [21, 19] on the dataset. Of course, each such computation only enables to determine whether two variables X and Y are *probabilistically* independent given a set $\mathbf{Z}$, but, *per se*, it does not determines whether X and Y are d-separated by $\mathbf{Z}$ (since we do not know yet graph $\mathcal{G}$). Ideally, we would like to have $X \perp_G Y|\mathbf{Z} \iff X \perp_P Y|\mathbf{Z}$ because it would imply that $\mathcal{G}$ contains an arc between X and Y if and only if X and Y are probabilistically dependent given any set $\mathbf{Z}$ [18], which can be tested using the dataset.

Unfortunately, this equivalence, which is referred to in the literature as *DAG-faithfulness*, does not always hold. It is known to hold for strictly positive probability distributions P but if $\mathbf{V}$ contains deterministic nodes, P cannot be strictly positive. As a matter of fact, a random variable X is said to be deterministic if there exists a subset $\mathbf{W} \subseteq \mathbf{V}\setminus\{X\}$ and a deterministic function f such that $X = f(\mathbf{W})$. Therefore, if $X = f(\mathbf{W})$, then $P(X = x, \mathbf{W} = \mathbf{w}) = 0$ whenever $x \neq f(\mathbf{w})$. Actually, whenever a deterministic node has a child in $\mathcal{G}$, DAG-faithfulness is lost. As an example, consider the BN of Fig. 1 where $\mathbf{V} = \{X, Y, Z, U, W\}$ and assume that $X = f(Y, Z, U)$ is the only deterministic node in $\mathbf{V}$. By d-separation, we have that $X \not\perp_G W|\{Y, Z, U\}$. However, $X \perp_P W|\{Y, Z, U\}$ because the values of Y, Z, U determine that of X, hence inducing that $P(X|W, Y, Z, U) = P(X|Y, Z, U)$.

[3] By abuse of notation, we use interchangeably $X \in \mathbf{V}$ to denote a node in the BN and its corresponding random variable.

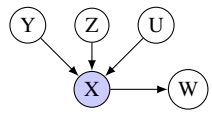

Figure 1. A BN with one deterministic relation $X = f(Y, Z, U)$

Learning DAG-unfaithful BNs is more difficult than DAG-faithful ones. Actually, under unfaithfulness, to keep the characterization of the BN's independence model by d-separation, it must be the case that $X \perp_G Y|\mathbf{Z} \implies X \perp_P Y|\mathbf{Z}$. However, as mentioned above, from the dataset, we can only infer statements of type $X \perp_P Y|\mathbf{Z}$. Hence, for learning, this is the converse implication which is needed.

3 RELATED WORKS

Several approaches have been proposed in the literature to address the problem of structure identification when some random variables are deterministic. For example, in [7], d-separation has been adapted to cope with deterministic variables, hence resulting in D-separation. The latter only adds in Definition 2 the third condition that, for a deterministic node S defined by $S = f(\mathbf{W})$, with $\mathbf{W} \subset \mathbf{Z}$, S does not have converging arrows on the trail. Unfortunately, applying this definition instead of that of d-separation is not sufficient to decrease the difficulty of learning the BN structure in the presence of deterministic random variables.

In Spirtes *et al.* [22], it is considered that deterministic variables are not essential for the independence model since they only express redundant information. Therefore, it is proposed to filter them out from data before learning the structure of the BN. However, it may be pointed out that, with datasets of limited sizes, such an approach can fail to learn correctly the structure of the BN. For instance, assume that the graphical structure $\mathcal{G}$ of a BN is such that $\mathbf{V} = \{X, W, Y_1, \ldots, Y_k, Z_1, \ldots, Z_r\}$ and that all these variables are Boolean. Assume in addition that X is deterministically defined as the exclusive OR of $Y_1, \ldots, Y_k$ and that W depends stochastically on $X, Z_1, \ldots, Z_r$. Then, by removing X, W depends on $Y_1, \ldots, Y_k, Z_1, \ldots, Z_r$, which, to be detected, requires an independence test over a contingency table of 2^{k+r+1} cells instead of 2^{r+2} cells for the dependence test of W with $X, Z_1, \ldots, Z_r$. As another example, consider a BN whose variables are X, Y, Z, T, with $X = f(Y, Z)$, and whose arcs are $Y \to X$, $Z \to X$, $X \to T$. Then X d-separates $\{Y, Z\}$ from T whereas, by removing X, this conditional independence cannot be taken into account in the BN.

Rodrigues de Morais *et al.* [20] addressed structure learning through an original determination of the Markov blanket (**MB**) of each node X of $\mathbf{V}$, i.e., the minimal set of nodes $\mathbf{Z}$ s.t., given $\mathbf{Z}$, X is independent from the rest of the network (in a BN, the **MB** of X is the set set of its parents, children and children's other parents). The idea relies on the identification of only the set of parents and children (**PC**) of the nodes, as specified in [23], i.e., $\mathbf{PC}(X)$ is equal to $\mathbf{MB}(X)$ minus the parents of the children of X. In Tsamardinos *et al.* [23], $\mathbf{PC}(X)$ is constructed incrementally, starting from an empty set and adding into it one by one new nodes Y such that $X \not\perp_P Y|\mathbf{Z}$ for all sets $\mathbf{Z}$ included in the current set $\mathbf{PC}(X)$. In a second step, for each variable $Y \in \mathbf{PC}(X)$, if there exists a set $\mathbf{Z} \subseteq \mathbf{PC}(X) \setminus \{Y\}$ such that $X \perp_P Y|\mathbf{Z}$, then Y is removed from $\mathbf{PC}(X)$. In a DAG-faithful context, two different BNs representing the same independence model have precisely the same **PC** sets and it is shown in [23] that a BN should contain an arc between nodes X

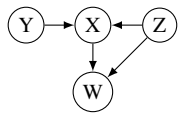

Figure 2. An example where $W \in \mathbf{PC}(Y)$ and $Y \notin \mathbf{PC}(W)$

and Y if and only if $Y \in \mathbf{PC}(X)$ *and* $X \in \mathbf{PC}(Y)$. The *and* condition is necessary to avoid false positive problems: consider actually the example of Fig. 2. Node Z cannot be added to $\mathbf{PC}(Y)$ because, if the BN is faithful, $Y \perp_P Z$. Node X is not d-separated of Y given $\emptyset$ and given $\{W\}$, so $X \in \mathbf{PC}(Y)$. Similarly, $W \in \mathbf{PC}(Y)$ because W and Y are not d-separated given $\emptyset$ and $\{X\}$. Conversely, $Y \notin \mathbf{PC}(W)$ because $X, Z \in \mathbf{PC}(W)$ and $Y \perp_P W|\{X, Z\}$. The "and" condition is thus compulsory to avoid adding arc $Y \rightarrow W$ in the graph. However, in a DAG-unfaithful context, this condition fails to produce the correct BN. For instance, if X is a deterministic node in Fig. 2, i.e., $X = f(Y, Z)$, then $Y, Z \in \mathbf{PC}(X)$ but this will rule out W belonging to $\mathbf{PC}(X)$ because $X \perp_P W|\{Y, Z\}$ since X does not bring any more information to W when Y and Z are already known. Therefore the "and" condition will prevent the existence of arc $X \rightarrow W$. To avoid this problem, it is suggested in [20] to substitute the "and" operator by an "or" operator. But as shown previously, this "or" operator will add arc $Y \rightarrow W$, which is not necessary.

In Luo [14], association rules miners are used to detect deterministic relations. Those are used in a constraint based algorithm (inductive causation (IC)) to build the BN as follows:

1. X and Y are connected by an arc iff $X \not\perp_P Y|\mathbf{Z}$ for every set $\mathbf{Z} \subseteq \mathbf{V} \setminus \{X, Y\}$ such that neither X nor Y are determined by $\mathbf{Z}$;
2. the unshielded triple $(X \rightarrow W \leftarrow Y)$ is considered as a V-structure iff the exists a set $\mathbf{Z} \not\supseteq \{W\}$ such that $X \perp_P Y|\mathbf{Z}$ and neither X nor Y are determined by $\mathbf{Z}$.

This method, while quite efficient, is unable to remove some arcs. For instance, if $\mathbf{V} = \{X, Y, Z, T\}$, with $T = f(Y, Z)$ and if the BN has a diamond shape with X at the top, T at bottom and Y, Z on the middle, then rule 1 above prevents discarding arc $X \rightarrow T$.

Finally, Lemeire *et al.* [13] provides the definition of a new class of unfaithfulness called "Information Equivalence" that follows from a broader class of relations than just deterministic ones. In addition to the latter, this class also contains equivalence partitions where each one is a source of unfaithfulness. Node sets $\mathbf{X}$ and $\mathbf{Y}$ are called information equivalent w.r.t. node Z if there exists a subset $\mathbf{W} \subseteq \mathbf{V} \setminus (\mathbf{X} \cup \mathbf{Y} \cup \{Z\})$ for which:

$$\mathbf{X} \not\perp_P Z|\mathbf{W}, \quad \mathbf{Y} \not\perp_P Z|\mathbf{W}, \quad \mathbf{X} \perp_P Z|\mathbf{Y} \cup \mathbf{W}, \quad \mathbf{Y} \perp_P Z|\mathbf{X} \cup \mathbf{W}. \quad (2)$$

The algorithm developed in [13] consists of testing, for each independence $X \perp_P Y|\mathbf{Z}$ observed, whether an equivalence information can be found by testing additionally whether $X \perp_P \mathbf{Z}|\{Y\} \cup \mathbf{W}$ and $Y \perp_P \mathbf{Z}|\{X\} \cup \mathbf{W}$ hold and whether Eq. (2) is satisfied. If this is the case, the arc between X and Y is not removed from $\mathcal{G}$. Information equivalence encompasses dependences due to deterministic relations but, in practice, we observed that independence tests often fail to reveal true information equivalences.

To summarize, there exist algorithms that try to deal with deterministic relations but all of them have serious shortcomings. In those methods, deterministic relations are perceived as sources of problems. We developed our approach with an opposite view: we try to exploit deterministic relations as new opportunities to help the learning algorithm to add or remove arcs in a principled manner.

4 A NEW LEARNING ALGORITHM SUITED FOR DETERMINISTIC RELATIONS

Like several other BN learning algorithms, ours is an hybrid that first determines the skeleton of the BN (or at least an approximation) and that, next, transforms it into a BN and refines it through a score-based search algorithm. In all the steps, we dedicate special rules to deterministic nodes. In the sequel, we will assume that DAG-unfaithfulness results only from the deterministic nodes and that, whenever $X = f(\mathbf{Z})$, all $Z \in \mathbf{Z}$ are parents of X in the BN.

4.1 First phase: learning the BN's skeleton

The skeleton of a BN is its graphical structure $\mathcal{G}$ in which the arcs have been transformed into (undirected) edges. As a consequence, there exists an arc $X \rightarrow Y$ in the BN if and only if there exists an edge $X - Y$ in the skeleton. Learning the skeleton prior to refine it into a BN is computationally attractive because the skeleton's undirected nature makes it faster to learn than BNs. There exist several algorithms for discovering the skeleton of a BN, for instance the PC algorithm[4] [21], Max-Min Parents Children [23] or Three-Phase Dependency Analysis [1]. For the first step of our method, we designed a variant of PC adapted to exploit deterministic relations. We chose PC because it is simple and efficient and also because the use of statistical independence tests are well suited for deterministic relations. To test whether $X \perp_P Y|\mathbf{Z}$, we compute a G^2-statistics, i.e.,

$$G^2(X, Y|\mathbf{Z}) = 2 \sum_{x \in X} \sum_{y \in Y} \sum_{\mathbf{z} \in \mathbf{Z}} N_{xy\mathbf{z}} \ln \frac{N_{xy\mathbf{z}} N_{\mathbf{z}}}{N_{x\mathbf{z}} N_{y\mathbf{z}}}, \quad (3)$$

where $N_{xy\mathbf{z}}, N_{x\mathbf{z}}, N_{y\mathbf{z}}, N_{\mathbf{z}}$ represent the number of occurrences of each tuple $(x, y, \mathbf{z}), (x, \mathbf{z}), (y, \mathbf{z}), \mathbf{z}$ in an N-sized dataset respectively. G^2 is known to provide more robust independence tests than χ^2. Note that G^2 also corresponds to $2N$ times the conditional mutual information between X and Y given $\mathbf{Z}$.

Basically, the idea of PC is to start with a complete undirected graph $\mathcal{G}$. Then, the algorithm iterates computations of independence tests between pairs X, Y given sets $\mathbf{Z}$ that are adjacent to X or Y. Whenever it finds that $X \perp_P Y|\mathbf{Z}$, edge $X - Y$ is removed from $\mathcal{G}$.

However, as shown in the preceding section, when $X = f(\mathbf{Z})$, the G^2-test always indicates that $X \perp_P Y|\mathbf{Z}$, for any Y, even if Y strongly depends on X (see Fig. 1). Therefore, such a G^2-test should never be used when X or Y is a deterministic node depending on $\mathbf{Z}$. This calls for a method to detect deterministic nodes. For this purpose, we exploit the conditional entropy $H(X|\mathbf{Z})$ of X given $\mathbf{Z}$:

$$\begin{aligned} H(X|\mathbf{Z}) &= -\sum_{x \in X} \sum_{\mathbf{z} \in \mathbf{Z}} P(x, \mathbf{z}) \log P(x|\mathbf{z}) \\ &= -\frac{1}{N} \sum_{x \in X} \sum_{\mathbf{z} \in \mathbf{Z}} N_{x\mathbf{z}} \log \frac{N_{x\mathbf{z}}}{N_{\mathbf{z}}}. \end{aligned} \quad (4)$$

It is well known that X is a deterministic node defined by $X = f(\mathbf{Z})$ if and only if $H(X|\mathbf{Z}) = 0$. Therefore our first adaptation of PC consists of computing the conditional entropies $H(X|\mathbf{Z})$ and $H(Y|\mathbf{Z})$ and, if one of them is equal to 0 (or is below a threshold), we just do not use the G^2-test over $(X, Y|\mathbf{Z})$ to remove edge $X - Y$. It should be noted here that computing these conditional entropies only marginally increases the overall learning's computation time. Actually, the most time consuming task is the parsing of the dataset in

[4] Note that the PC algorithm is not related with the parent-child algorithm **PC** mentioned in the preceding section.

order to evaluate observation counts $N_{xy\mathbf{z}}, N_{x\mathbf{z}}, N_{y\mathbf{z}}, N_{\mathbf{z}}$. The computations of Eqs. (3) and (4) are significantly faster than this task.

However, while the use of the conditional entropy overcomes the problem of unfaithfulness due to erroneous inductions from G^2-tests performed on deterministic nodes, it can also raise another problem of statistical indistinguishably [14]: there exist cases where conditional independences can be explained by both d-separation and by deterministic relations. For instance, in Fig. 3, the conditional independence between B and D given $\{A, C\}$ can result from B and D being d-separated by $\{A, C\}$ but also from D being a deterministic node defined by $D = f(A, C)$. By the preceding paragraph, our algorithm first checks whether $H(D|\{A, C\}) = 0$. As this is the case since $D = f(A, C)$, edge $B - D$ will not be removed from $\mathcal{G}$ due to a G^2 test involving D given $\{A, C\}$. In such a case, [14] keeps this edge as well. However, as will be shown in Proposition 1, edge $B - D$ can always be safely removed from $\mathcal{G}$:

Proposition 1 *Let X be a deterministic node defined by $X = f(\mathbf{Z})$ and let Y be a node such that, in the skeleton $\mathcal{G}$ learnt by the algorithm, $Y \notin \mathbf{Z}$ and $\mathbf{Z} \subseteq \mathbf{Adj}(Y)$, where $\mathbf{Adj}(Y)$ refers to the set of neighbors of Y in $\mathcal{G}$. Then edge $X - Y$ can be removed from $\mathcal{G}$.*

Proof. Our goal is to learn a skeleton $\mathcal{G}$. Hence to any of its edges must correspond an arc in the BN. Three cases can obtain:

- assume that, in the BN, there exists an arc $X \rightarrow Y$ and that all the nodes, say Z, of $\mathbf{Z}$ are such that $Z \rightarrow Y$. Then $\{X\} \cup \mathbf{Z} \subseteq \mathbf{Pa}(Y)$. Define $\mathbf{K}$ as $\mathbf{Pa}(Y) \setminus (\{X\} \cup \mathbf{Z})$. Then $P(Y|\mathbf{Pa}(Y)) = P(Y|\{X\} \cup \mathbf{Z} \cup \mathbf{K}) = P(Y|\mathbf{Z} \cup \mathbf{K})$ because X does not bring any additional information once $\mathbf{Z}$ is known. Therefore arc $X \rightarrow Y$ can be removed from the BN as well as edge $X - Y$ from $\mathcal{G}$.
- assume that, in the BN, there exists an arc $X \rightarrow Y$ and that there exists at least one node $Z \in \mathbf{Z}$ such that $Y \rightarrow Z$. Then X, Y, Z forms a directed cycle since $Z \in \mathbf{Z}$ is also a parent of X as $X = f(\mathbf{Z})$. This is impossible since BNs are DAGs.
- assume that, in the BN, there exists an arc $Y \rightarrow X$. Then $\{Y\} \cup \mathbf{Z} \subseteq \mathbf{Pa}(X)$. Define $\mathbf{K}$ as $\mathbf{Pa}(X) \setminus (\{Y\} \cup \mathbf{Z})$. Then $P(X|\mathbf{Pa}(X)) = P(X|\{Y\} \cup \mathbf{Z} \cup \mathbf{K}) = P(X|\mathbf{Z})$ because, once $\mathbf{Z}$ is known, X is determined. Hence arc $Y \rightarrow X$ can be removed from the BN as well as edge $X - Y$ from $\mathcal{G}$. ■

At the end of the first phase of our algorithm, for each deterministic node X, all edges $X - Y$ for nodes Y satisfying the conditions of Proposition 1 are discarded. This proposition is applied only at the end of the learning phase because we start with a complete graph $\mathcal{G}$ hence, at the beginning, every node Y satisfies the conditions.

Another variant with the original PC algorithm that was already advocated in [21] and [11] is the use of min cutsets to determine the conditioning sets in our independence tests: PC usually tests the independence between X and Y first by computing $G^2(X, Y|\emptyset)$ then, if this test fails (i.e., if it does not provides evidence that X is independent of Y), conditional tests $G^2(X, Y|\mathbf{Z})$ are performed iteratively with sets $\mathbf{Z}$, adjacent to X or Y, of increasing sizes until either a conditional independence is proved or the size of the database does not allow any more meaningful statistical test. In our algorithm, we not only performed these tests with $\mathbf{Z}$ of increasing sizes, we also computed a minimal cutset $\mathbf{Cut}_{XY}$ in $\mathcal{G}$ separating X and Y, i.e., a set of nodes $\mathbf{Cut}_{XY}$ which, when removed from $\mathcal{G}$ make X and Y belong to two different connected components. This can be easily done by computing a min cut in a max-flow problem. If the size of $\mathbf{Cut}_{XY}$ is smaller than that of sets $\mathbf{Z}$ PC would have used, we just perform $G^2(X, Y|\mathbf{Cut}_{XY})$. This results in the construction of smaller contingency tables and, as such, this increases the accuracy of the statistical test. Moreover, the additional time required to compute $\mathbf{Cut}_{XY}$ can be balanced by the number of G^2-tests it allows to avoid. Of course, for large graphs, computing min cutsets can be expensive. In this case, we resort to a technique advocated in [8]: basically, graph $\mathcal{G}$ is incrementally triangulated, hence resulting in a junction tree incrementally constructed [6]. Then, some cliques $\mathbf{C}_X$ and $\mathbf{C}_Y$ containing X and Y are identified as well as the minimal-size separator $\mathbf{S}_{XY}$ on the trail between $\mathbf{C}_X$ and $\mathbf{C}_Y$. Set $\mathbf{S}_{XY}$ is a cutset and, thus, to test the independence between X and Y, it is sufficient to compute $G^2(X, Y|\mathbf{S}_{XY})$. In practice, incremental triangulations are very fast to perform and sets $\mathbf{S}_{XY}$ are relatively small.

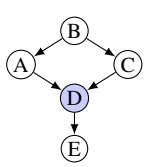

Figure 3. A BN with a deterministic relation $D = f(A, C)$

The last variant we introduced concerns deterministic variables: the proposition below shows that some edges can be removed from $\mathcal{G}$ without requiring the computation of a G^2-independence test:

Proposition 2 *Let X and Y be two deterministic nodes, i.e., $X = f(\mathbf{W})$ and $Y = f(\mathbf{Z})$. Assume that $X \notin \mathbf{Z}$ and that $Y \notin \mathbf{W}$. Then edge $X - Y$ can be safely removed from $\mathcal{G}$.*

Proof. Let us assume that the BN we learn contains arc $X \rightarrow Y$. Then, $\{X\} \cup \mathbf{Z} \subseteq \mathbf{Pa}(Y)$. But then, $P(Y|\mathbf{Pa}(Y)) = P(Y|\mathbf{Z})$ since $Y = f(\mathbf{Z})$. Therefore, removing from the BN all the arcs in $\mathbf{Pa}(Y) \setminus \mathbf{Z}$ results in a new BN that represents the same distribution. Therefore, it is safe to discard arc $X \rightarrow Y$. By symmetry, it would also be safe to discard arc $Y \rightarrow X$. $\mathcal{G}$ being a skeleton, it is thus safe to remove edge $X - Y$ from $\mathcal{G}$. ■

Therefore, each time our algorithm discovers a new deterministic node, it applies Proposition 2 to remove all the edges between this node and the previously discovered deterministic nodes satisfying the conditions. Overall, our algorithm for computing the BN's skeleton when there exist deterministic nodes is described in Algo. 1.

4.2 Second phase: orientation and refinement

The next step of our learning algorithm consists of orienting the edges in order to get an initial BN that will subsequently be refined. In [21], PC first identifies using the computed "SepSet" the BN's V-structures, i.e., triples of nodes (X, Y, Z) that are connected as $X - Y - Z$ and such that $X \not\perp_P Z|\mathbf{Y}$ and $X \perp_P Z|\mathbf{W}$ for some $\mathbf{W} \not\supseteq \{Y\}$. For these triples, the BN should contain the following arcs: $X \rightarrow Y \leftarrow Z$. A BN is fully characterized by its skeleton and the set of its V-structures. When deterministic nodes exist, it is suggested in [14] to add a extra condition on sets $\mathbf{W}$: if X or Y is a deterministic node defined by some set $\mathbf{S}$, then $\mathbf{W} \not\supseteq \mathbf{S}$. Our algorithm follows the same rule.

In our framework, we add, prior to the V-structures orientation, the following "causal" constraint: for each deterministic node $X = f(\mathbf{W})$, all the edges $W - X$, for $W \in \mathbf{W}$, are converted into arcs $W \rightarrow X$[5]. But deterministic relations can be further exploited; ex-

[5] If we detect two deterministic nodes that are in a one-to-one mapping, we simply discard one of them to avoid redundancies.

Input: a dataset $\mathcal{D}$
Output: a skeleton $\mathcal{G}$
Start with a complete undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{A})$
DR $\leftarrow \emptyset$ // the set of deterministic relations found so far
SepSet $\leftarrow \emptyset$ // the set of independences $X \perp_P Y|\mathbf{Z}$ found so far
$i \leftarrow 0$ // the size of the conditioning sets tested
repeat
  foreach *edge* $X - Y$ *in* $\mathcal{G}$ **do**
    if *there exists a set* $\mathbf{Z} = \mathbf{Cut}_{XY}$ *or* $\mathbf{S}_{XY}$ *of size* $\leq i$, *or* $\mathbf{Z}$ *adjacent to* X *or to* Y *of size* i, *s.t.* $X \perp_P Y|\mathbf{Z}$ **then**
      if $\nexists\, \mathbf{S}, \mathbf{T} \subseteq \mathbf{Z}$ *s.t.* $(X|\mathbf{S}) \in DR$ *or* $(Y|\mathbf{T}) \in DR$ **then**
        NewDeterministicRelation $\leftarrow$ false
        if $H(X|\mathbf{Z}) = 0$ **then**
          find the smallest $\mathbf{S} \subseteq \mathbf{Z}$ s.t. $H(X|\mathbf{S}) = 0$
          DR $\leftarrow$ DR $\cup \{(X|\mathbf{S})\}$
          apply Proposition 2 to remove edges
          NewDeterministicRelation $\leftarrow$ true
        if $H(Y|\mathbf{Z}) = 0$ **then**
          find the smallest $\mathbf{T} \subseteq \mathbf{Z}$ s.t. $H(Y|\mathbf{T}) = 0$
          DR $\leftarrow$ DR $\cup \{(Y|\mathbf{T})\}$
          apply Proposition 2 to remove edges
          NewDeterministicRelation $\leftarrow$ true
        if *NewDeterministicRelation* $=$ *false* **then**
          $\mathbf{A} \leftarrow \mathbf{A} \setminus \{X - Y\}$
          SepSet $\leftarrow$ SepSet $\cup \{(X, Y|\mathbf{Z})\}$
until *all nodes in* $\mathbf{V}$ *have at most* i *neighbors*;
use Proposition 1 to remove unnecessary edges
return *undirected graph* $\mathcal{G} = (\mathbf{V}, \mathbf{A})$

Algorithm 1: Learning the skeleton of the BN

amples of such exploitation can be found, e.g., in [12] and [3]. Here, we propose to leverage deterministic relation features to deduce some orientations that are necessary. As an illustration, consider skeleton $\mathcal{G}$ of Fig. 4.a where $X = f(U, Y, Z)$ is a deterministic node. Our "causal" constraint imposes the orientation of Fig. 4.b. But, as shown in the proposition below, the only reasonable orientation for the remaining edges adjacent to X are those given in Fig. 4.c.

Proposition 3 *Let* $\mathcal{G}$ *be a skeleton and let* X *be a deterministic node defined by* $X = f(\mathbf{W})$. *Let* $\mathbf{Z} = \mathbf{Adj}(X) \setminus \mathbf{W}$ *in* $\mathcal{G}$. *Then for each* $Z \in \mathbf{Z}$, *edge* $X - Z$ *must be oriented as* $X \rightarrow Z$.

Proof. Assume that the BN contains arc $Z \rightarrow X$. In the BN, let $\mathbf{K} = \mathbf{Pa}(X) \setminus (\mathbf{W} \cup \{Z\})$. Then the joint probability modeled by the BN is $P(\mathbf{V}) = P(X|\mathbf{Pa}(X)) \times \prod_{V \in \mathbf{V} \setminus \{X\}} P(V|\mathbf{Pa}(V)) = P(X|\mathbf{W}) \times \prod_{V \in \mathbf{V} \setminus \{X\}} P(V|\mathbf{Pa}(V))$ because, once $\mathbf{W}$ is known, $\mathbf{K}$ and Z do not bring any additional information on X. Therefore arc $Z \rightarrow X$ can be removed from the BN without altering the probability distribution. But this is impossible because edge $X - Z$ belonging to skeleton $\mathcal{G}$ implies that X and Z are conditionally dependent given any other set of nodes. Hence, the arc between X and Z in the BN is necessarily $X \rightarrow Z$. ■

With all those rules, our algorithm converts the skeleton into a Completed Partially Directed Acyclic Graph (CPDAG). The remaining edges are then converted in a similar fashion as PC does (when PC is unable to orient edges, we select arbitrarily an orientation). At this point, our algorithm has constructed an initial BN. This one is then refined using any search algorithm [23]. The only constraint we add on this algorithm is that it never modifies the arcs adjacent to the deterministic nodes nor add new ones. As a matter of fact, due to unfaithfulness induced by determinism, classical algorithms will tend to erroneously modify the neighborhood of the deterministic nodes.

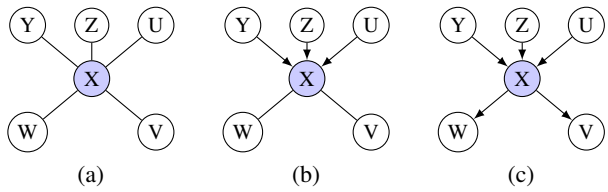

Figure 4. A skeleton $\mathcal{G}$ with a deterministic node $X = f(Y, Z, U)$

5 EXPERIMENTATIONS

In this section, we highlight the effectiveness of our method by comparing it with MMHC [23] (which is not suited to cope with deterministic nodes), "OR+Inter.IAMB" [20] and the algorithm of Luo [14] (substituting its association rule miners by conditional entropy tests to detect deterministic nodes, in order to improve its effectiveness). For this purpose, we generated randomly BNs containing 10 to 50 nodes and 12 to 92 arcs. Nodes had at most 6 parents and their domain size was randomly set between 2 and 6. Finally, the number of deterministic nodes was chosen arbitrarily between 1 and 15. From these BNs, we generated samples of sizes ranging from 1000 to 50000. Overall, 750 datasets were generated. Each BN produced by MMHC, OR+Inter.IAMB, Luo and our algorithm on these dataset has been converted into a CPDAG corresponding to its Markov equivalence class (i.e., a skeleton in which V-structures are oriented). This makes comparisons more meaningful since two BNs represent exactly the same independence model iff they belong to the same Markov equivalence class, i.e., they have the same CPDAGs. Finally, the CPDAGs were compared against those of the true BNs using two metrics: the true positive rate (TPR) and the "accuracy". TPR is defined as the number of arcs / edges present in the learnt CPDAG that also exist in the true BN's CPDAG (with the same orientations for arcs) divided by the number of arcs / edges in the true BN's CPDAG. This metric thus describes the power of the learning algorithm to correctly recover the dependences between the random variables. The accuracy is defined as the number of dependences and independences correctly recovered, i.e., the number of pairs of variables which are either unlinked or linked similarly in both CPDAGs, divided by the number of pairs of random variables. This metric thus quantifies how good is the learning algorithm to both discover dependences and independences.

Figure 5 displays the averages and standard deviations (error bars) over the 750 datasets of the TPR and accuracy metrics for the four algorithms. Curves being similar, we averaged the results for BNs with 10, 15, 25 nodes on the left and 35, 40, 50 nodes on the right. As can be seen, our algorithm always substantially outperform the other two, even asymptotically for large datasets. The TPR of our method is actually about 15% and 25% higher than that of Or+Inter.Iamb and Luo respectively. The overall accuracy is slightly lower but still significant. The performance of our algorithm follows mainly from its rules dedicated to deterministic nodes: by using conditional entropy, it avoids discarding edges that are needed and, by Proposition 3, it correctly orient the edges in the neighborhood of deterministic nodes. This explains why its TPR is higher than the other methods. For the accuracy, in addition to correctly recovering dependences, Propositions 1 and 2 enabled our algorithm to remove arcs that the other

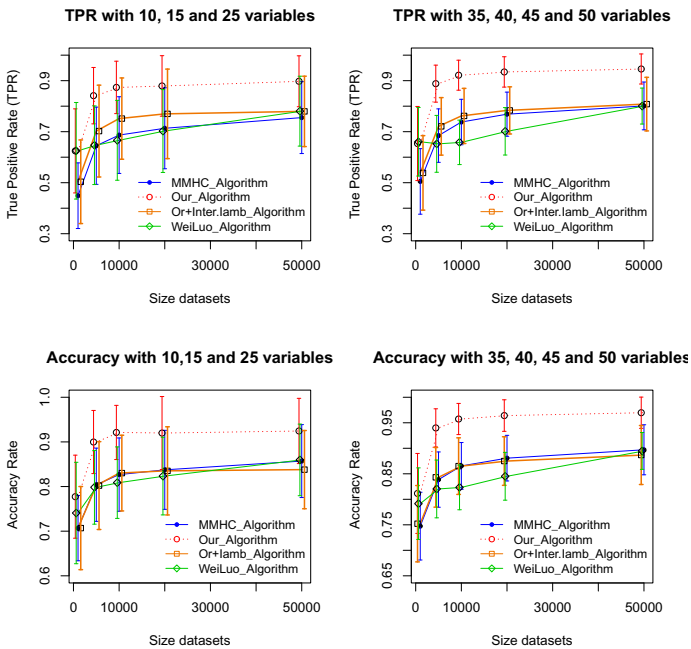

Figure 5. Averages (and standard deviations) of the TPR and accuracy metrics for BNs with 10 to 50 nodes in function of the sample sizes.

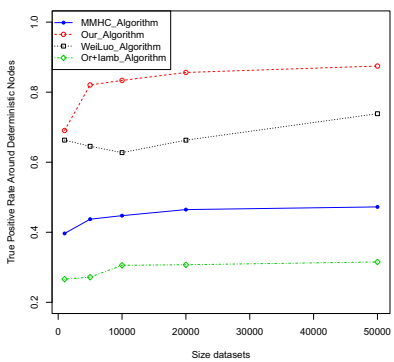

Figure 6. Averages of the TPRs computed only around deterministic nodes, for all the BNs, in function of the sample sizes.

methods were unable to remove, hence making it more suited for recovering independences. Finally, our original selection of the conditioning sets in the G^2-tests also helped discovering conditional independences.

Figure 6 provides TPRs around deterministic nodes, i.e., TPRs computed only with the edges / arcs for which at least one extremal node is deterministic. Again, our algorithm significantly outperforms the others, hence highlighting the efficiency of our rules, notably that of Proposition 3.

6 CONCLUSION

In this paper, we have proposed a new algorithm for learning the structure of BNs when some variables are deterministic. This algorithm relies on very effective dedicated rules whose mathematical correctness has been proved. As shown in the experimentations, these rules enable our algorithm to significantly outperform state-of-the-art algorithms. For future works, we plan to extend our algorithm for learning the structures of dynamic Bayesian networks, notably non-stationary ones, and to adapt it to other sources of unfaithfulness (e.g., equivalence partitions[13]).

References

[1] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu, 'Learning Bayesian networks from data: An information-theory based approach', *Artificial Intelligence*, **137**(1-2), 43–90, (2002).

[2] G. F. Cooper and E. Herskovits, 'Bayesian method for the induction of probabilistic networks from data', *Machine Learning*, **9**, 309–347, (1992).

[3] P. Daniusis, D. Janzing, J. Mooij, J. Zscheischler, B. Steudel, K. Zhang, and B. Schölkopf, 'Inferring deterministic causal relations', in *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 143–150, (2010).

[4] A. Darwiche, 'Recursive conditioning', *Artificial Intelligence Journal*, **125**(1-2), 5–41, (2001).

[5] A. Feelders, 'A new parameter learning method for Bayesian networks with qualitative influences', in *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 117–124, (2007).

[6] J. Flores, J. Gamez, and K. Olesen, 'Incremental compilation of Bayesian networks', in *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 233–240, (2003).

[7] D. Geiger, T. Verma, and J. Pearl, 'Identifying independence in Bayesian networks', *Networks*, **20**, 507–534, (1990).

[8] C. Gonzales and N. Jouve, 'Learning Bayesian networks structure using Markov networks', in *Proc. of the European Workshop on Probabilistic Graphical Models (PGM)*, pp. 147–154, (2006).

[9] D. Heckerman, 'A tutorial on learning with Bayesian networks', Technical Report TR-95-06, Microsoft Research, (1995).

[10] D. Heckerman, D. Geiger, and D. M. Chickering, 'Learning Bayesian networks: The combination of knowledge and statistical data', *Machine Learning*, **20**, 197–243, (1995).

[11] J.Abellan, M. Gomez-Olmedo, and S. Moral, 'Some variations on the PC algorithm', in *Proc. of the European Workshop on Probabilistic Graphical Models (PGM)*, (2006).

[12] D. Janzing, P. Hoyer, and B. Schölkopf, 'Telling cause from effect based on high-dimensional observations', in *Proc. of the International Conference on Machine Learning (ICML)*, pp. 479–486, (2010).

[13] J. Lemeire, S. Meganck, F. Cartella, and T. Liu, 'Conservative independence-based causal structure learning in absence of adjacency faithfulness', *International Journal of Approximate Reasoning*, **53**(9), 1305–1325, (2012).

[14] W. Luo, 'Learning Bayesian networks in semi-deterministic systems', in *Proc. of the Canadian Conference on AI*, volume 4013 of *Lecture notes in computer science*, pp. 230–241, (2006).

[15] A.L. Madsen and F.V. Jensen, 'LAZY propagation: A junction tree inference algorithm based on lazy inference', *Artificial Intelligence*, **113**(1–2), 203–245, (1999).

[16] D Nilsson, 'An efficient algorithm for finding the M most probable configurations in probabilistic expert systems', *Statistics and Computing*, **8**(2), 159–173, (1998).

[17] J. D. Park and A. Darwiche, 'Complexity results and approximation strategies for MAP explanations', *Journal of Artificial Intelligence Research*, **21**, 101–133, (2004).

[18] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kauffmann, 1988.

[19] J. Pearl, *Causality: Models, Reasoning and Inference*, Cambridge University Press, Cambridge, England, 2000.

[20] S. Rodrigues de Morais, A. Aussem, and M. Corbex, 'Handling almost-deterministic relationships in constraint-based Bayesian network discovery: Application to cancer risk factor identification', in *Proc. of the European Symposium on Artificial Neural Networks (ESANN)"*, pp. 101–106, (2008).

[21] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction and Search*, Bradford Book, 2nd edn., 2001.

[22] P. Spirtes, R. Scheines, C. Meek, T. Richardson, C. Glymour, H. Hoijtink, and A. Boomsma, *TETRAD 3: Tools for Causal Modeling – User's Manual*, 1996.

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