Learning Bayesian Networks Structure using Markov Networks

Christophe Gonzales and Nicolas Jouve
Laboratoire d'Informatique de Paris VI
8 rue du capitaine Scott 75015 Paris, France

Abstract

This paper addresses the problem of learning a Bayes net (BN) structure from a database. We advocate first searching the Markov networks (MNs) space to obtain an initial RB and, then, refining it into an optimal RB. More precisely, it can be shown that under classical assumptions our algorithm obtains the optimal RB moral graph in polynomial time. This MN is thus optimal w.r.t. inference. The process is attractive in that, in addition to providing optimality guarrantees, the MN space is substantially smaller than the traditional search spaces (those of BNs and equivalent classes (ECs)). In practice, we face the classical shortcoming of constraint-based methods, namely the unreliability of high-order conditional independence tests, and handle it using efficient conditioning set computations based on graph triangulations. Our preliminary experimentations are promising, both in terms of the quality of the produced solutions and in terms of time responses.

1 Introduction

Effective modeling of uncertainty is essential to most AI applications. For fifteen years probabilistic graphical models like Bayesian nets (BN) and Markov nets (MN), a.k.a. Markov random fields, (Cowell et al., 1999; Jensen, 1996; Pearl, 1988) have proved to be well suited and computationally efficient to deal with uncertainties in many practical applications. Their key feature consists in exploiting probabilistic conditional independences (CIs) to decompose a joint probability distribution over a set of random variables as a product of functions of smaller sets, thus allowing a compact representation of the joint probability as well as efficient inference algorithms (Allen and Darwiche, 2003; Madsen and Jensen, 1999). These independences are encoded in the graphical structure of the model, which is directed for BNs and undirected for MNs. In this paper, we propose an algorithm to learn a BN structure from a data sample of the distribution of interest.

There exist three main classes of algorithms for learning BN from data. The first one tries to determine the set of all probabilistic CIs using statistical independence tests (Verma and Pearl, 1990; Spirtes et al., 2000). Such tests have been criticized in the literature as they can only be applied with small conditioning sets, thus ruling out complex BNs. A more popular approach consists in searching the BN structures space, optimizing some score (BDeu, MDL, etc.) measuring how well the structure fits data (Heckerman et al., 1995; Lam and Bacchus, 1993). Unfortunately, the number of BNs is super-exponential in the number of random variables (Robinson, 1973) and an exhaustive comparison of all the structures is impossible. One way out is to use local search algorithms parsing efficiently the BN structure space, moving from one BN to the next one by performing simple graphical modifications (Heckerman et al., 1995). However, these suffer from the existence of multiple BNs representing the same set of CIs. This not only lengthens the search but may also trap it into local optima. To avoid these problems, the third class of approaches (Munteanu and Bendou, 2001; Chickering, 2002) searches the space of BN equivalence classes (EC). In this space, equivalent BNs are represented by a unique partially directed graph. In addition to speeding-up the search by avoiding useless moves from one BN to an equivalent one, this class of algorithms possesses nice theoretical properties. For instance, under DAG-isormorphism, a classical hypothesis on the data generative distribution, and in the limit of a large database, the GES algorithm is theoretically able to recover the optimal BN structure (Chickering, 2002). This property is remarkable as very few search algorithms are able to guarantee the quality of the returned structure.

Although, in theory, the EC space seems more attractive than the BN space, it suffers in practice from two problems: i) its neighborhood is exponential in the number of nodes (Chickering, 2002) and ii) the EC space size is roughly equal to that of the BN space (Gillispie and Perlman, 2001). It would thus be interesting to find a space preserving the optimality feature of the EC space exploited by GES while avoiding the above problems. For this purpose, the MN space seems a good candidate as, like the EC space, its equivalence classes are singletons. Moreover, it is exponentially smaller than the BN space and, by its undirected nature, its neighborhood is also exponentially smaller than that of EC. This suggests that searching the MN space instead of the EC space can lead to significant improvements.

To support this claim, we propose a learning algorithm that mainly performs its search in the MN space. More precisely, it is divided into three distinct phases. In the first one, we use a local search algorithm that finds an optimal MN searching the MN space. It is well-known that only chordal MNs are precisely representable by BNs (Pearl, 1988). As it is unlikely that the MN we find at the end of the first phase is actually chordal, its transformation into a BN must come along with the loss of some CIs. Finding a set of CIs that can be dispensed with to map the MN into a BN while fitting data as best as possible is not a trivial task. Phase 2 exploits the relationship between BNs and their moral graphs to transform the MN into a BN whose moral graph is not far from the MN obtained at the end of phase 1. Then, in a third phase, this BN is refined into one that better fits data. This last phase uses GES second step. The whole learning algorithm preserves GES theoretical optimality guarantee. Furthermore, the BN at the end of phase 2, which possesses interesting theoretical properties, is obtained in polynomial time. Phase 3 is exponential but has an anytime property. Preliminary experimental results suggest that our learning algorithm is faster than GES and produces better quality solutions.

The paper is organized as follows. Section 2 provides some background on MNs and BNs. Then, Section 3 describes how we obtain the optimal MN. Section 4 shows how it can be converted into a BN. Finally Section 5 mentions some related work and presents some experimental results.

2 Background

Let $\mathcal{V}$ be a set of random variables with joint probability distribution $P(\mathcal{V})$. Variables are denoted by capitalized letters (other than P) and sets of variables (except $\mathcal{V}$) by bold letters.

Markov and Bayes nets are both composed of: i) a graphical structure whose nodes are the variables in $\mathcal{V}$ (hereafter, we indifferently refer to nodes and their corresponding variables) and ii) a set of numerical parameters. The structure encodes probabilistic CIs among variables and then defines a family of probability distributions. The set of parameters, whose form depends on the structure, defines a unique distribution among this family and assesses it numerically. Once a structure is learned from data, its parameters are assessed, generally by maximizing data likelihood given the model.

A MN structure is an undirected graph while a BN one is a directed acyclic graph (DAG). MNs graphically encode CIs by separation. In an undirected graph $\mathcal{G}$, two nodes X and Y are said to be *separated* by a disjoint set $\mathbf{Z}$, denoted by $X \perp_{\mathcal{G}} Y \mid \mathbf{Z}$, if each chain between X and Y has a node in $\mathbf{Z}$. BNs criterion, called d-separation, is defined similarly except that it distinguishes *colliders* on a chain, i.e., nodes with their two adjacent arcs on the chain directed toward them. In a DAG, X and Y are said to be *d-separated* by $\mathbf{Z}$, denoted by $X \perp_{\mathcal{B}} Y \mid \mathbf{Z}$, if for each chain between X and Y there exists a node S s.t. if S is a collider on the chain, then neither S nor any of its descendant is in $\mathbf{Z}$, else S is in $\mathbf{Z}$. Extending Pearl's terminology (Pearl, 1988), we will say that a graph $\mathcal{G}$, directed or not, is an *I-map* (*I* standing for *independency*), implicitly relative to P, if $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z} \Longrightarrow \mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z}$. We will also say that a graph $\mathcal{G}$ is an I-map of another graph $\mathcal{G}'$ when $\mathbf{X} \perp_{\mathcal{G}} \mathbf{Y} \mid \mathbf{Z} \Longrightarrow \mathbf{X} \perp_{\mathcal{G}'} \mathbf{Y} \mid \mathbf{Z}$. When two structures are I-maps of one another, they represent the same family and are thus said to be *equivalent*.

The complete graph, which exhibits no separation assertion, is a structure containing all the possible distributions and is, as such, always an I-map. Of course, in a learning prospect, our aim is to recover from data an I-map as sparse as possible. More precisely, in both MN and BN frameworks, an I-map $\mathcal{G}$ is said to be *optimal* (w.r.t. inclusion) relatively to P if there exists no other I-map $\mathcal{G}'$ such that i) $\mathcal{G}$ is an I-map of $\mathcal{G}'$ and ii) $\mathcal{G}$ is not equivalent to $\mathcal{G}'$.

Though they are strongly related, BNs and MNs are not able to represent precisely the same independence shapes. MNs are mostly used in the image processing and computer vision community (Pérez, 1998), while BNs are particularly used in the AI community working on modeling and prediction tools like expert systems (Cowell et al., 1999). In the latter prospect, BNs are mostly preferred for two reasons. First, the kind of independences they can express using arcs orientations are considered more interesting than the independence shapes only representable by MNs (Pearl, 1988). Secondly, as BNs parameters are probabilities (while those of MNs are non-negative potential functions without any real actual meaning), they are considered easier to assess, to manipulate and to interpret. In BNs, the direction is exploited only through *V-structures*, that is, three-node chains whose central node is a collider the neighbors of which are not connected by an arc. This idea is expressed in a theorem (Pearl, 1988) stating that two BNs are equivalent if and only if they have the same V-structures and the same *skeleton*, where the skeleton of a DAG is the undirected graph resulting from the removal of its arcs direction.

MNs are unable to encode V-structure information since it is a directed notion. As a DAG without V-structure is chordal (i.e. triangulated), it is not surprising that a result (Pearl et al., 1989) states that independences of a family of distributions can be represented by both MNs and BNs frameworks if and only if the associated structure is chordal. In the general case where the graph is not chordal, it is possible to get an optimal MN from a BN by *moralization*. The *moral graph* of a DAG is the undirected graph obtained by first adding edges between non adjacent nodes with a common child (i.e. the extremities of V-structures) and then removing the arcs orientations. It is easily seen that the moral graph of a BN is an optimal undirected I-map of this graph, even if some independences of the BN have been necessarily loosed in the transformation. The converse, i.e. getting a directed I-map from a MN, is less easy and will be addressed in Section 4.

Like most works aiming to recover an optimal structure from data, we will assume that the underlying distribution P is *DAG-isomorph*, i.e. that there exists a DAG $\mathcal{B}^*$ encoding exactly the independences of P (s.t. $\mathbf{X} \perp_{\mathcal{B}^*} \mathbf{Y} \mid \mathbf{Z} \Longleftrightarrow \mathbf{X} \perp\!\!\!\perp_P \mathbf{Y} \mid \mathbf{Z}$). Under this assumption, $\mathcal{B}^*$ and equivalent BNs are obviously optimal. Using the axiomatic in (Pearl, 1988), it can be shown that, in the MN framework, the DAG-isomorphism hypothesis entails the so-called intersection property, leading to the existence of a unique optimal MN, say $\mathcal{G}^*$ (Pearl, 1988). Moreover, $\mathcal{G}^*$ is the moral graph of $\mathcal{B}^*$.

3 Markov network search

Searching the BN or the EC space is often performed as an optimization process, that is, the algorithm looks for a structure optimizing some goodness of fit measure, the latter being a decomposable scoring function that involves maximum likelihood estimates. For MNs, the computation of these estimates is hard and requires time-expensive methods (Murray and Ghahramani, 2004) (unless the MN is triangulated, which is not frequent). Hence score-based exploration strategies seem inappropriate for MN searches. Using statistical CI tests and combining them to reveal the MN's edges seems a more promising approach. This is the one we follow here.

```
Algorithm LEARNMN
Input : database
Output : moral graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_2)$
  1. $\mathcal{E}_1 \leftarrow \emptyset$
  2. foreach edge $(X,Y) \notin \mathcal{E}_1$ do
         search, if necessary, a new $\mathbf{S_{XY}}$
         s.t. $X \perp_{(\mathcal{V},\mathcal{E}_1)} Y \mid \mathbf{S_{XY}}$
  3. if $\exists\, (X,Y) \notin \mathcal{E}_1$ s.t. $X \not\perp\!\!\!\perp Y | \mathbf{S_{XY}}$ then
  4.     add edge $(X,Y)$ to $\mathcal{E}_1$ and go to line 2
  5. $\mathcal{E}_2 \leftarrow \mathcal{E}_1$
  6. foreach edge $(X,Y) \in \mathcal{E}_2$ do
         search, if necessary, a new $\mathbf{S_{XY}}$
         s.t. $X \perp_{(\mathcal{V},\mathcal{E}_2 \setminus \{(X,Y)\})} Y \mid \mathbf{S_{XY}}$
  7. if $\exists\, (X,Y) \in \mathcal{E}_2$ s.t. $X \perp\!\!\!\perp Y | \mathbf{S_{XY}}$ then
  8.     remove edge $(X,Y)$ from $\mathcal{E}_2$ and go to line 6
  9. return $\mathcal{G} = (\mathcal{V}, \mathcal{E}_2)$
End of algorithm
```

Unlike most works where learning a MN amounts to independently learn the Markov blanket of all the variables, we construct a MN in a local search manner. Algorithm LEARNMN consists in two consecutive phases: the first one adds edges to the empty graph until it converges toward a graph $\mathcal{G}_1 =$

$(\mathcal{V}, \mathcal{E}_1)$. Then it removes from $\mathcal{G}_1$ as many edges as possible, hence resulting in a graph $\mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_2)$.

At each step of phase 1, we compute the dependence of each pair of non-adjacent nodes (X, Y) conditionally to any set $\mathbf{S_{XY}}$ separating them in the current graph. We determine the pair with the strongest dependence (using a normalized difference to the χ^2 critical value) and we add the corresponding edge to the graph and update separators.

Lemma 1. *Assuming DAG-isomorphism and sound statistical tests, graph $\mathcal{G}_1$ resulting for the first phase of* LEARNMN *is an I-map.*

Sketch of proof. At the end of phase 1, $\forall\ (X, Y) \notin \mathcal{E}_1$, there exists $\mathbf{S_{XY}}$ s.t. $X \perp_{\mathcal{G}_1} Y \mid \mathbf{S_{XY}}$ and $X \perp\!\!\!\perp Y \mid \mathbf{S_{XY}}$. By DAG-isomorphism, it can be shown that $\mathcal{G}_1$ contains the skeleton of $\mathcal{B}^*$, and, then, that it also contains its moralization edges. □

In the second phase, which takes an I-map $\mathcal{G}_1$ as input, LEARNMN detects the superfluous edges in $\mathcal{G}_1$ and iteratively removes them.

Proposition 1. *Assuming DAG-isomorphism and sound statistical tests, the graph $\mathcal{G}_2$ returned by* LEARNMN *is the optimal MN $\mathcal{G}^*$.*

Sketch of proof. At the end of phase 2, $\forall\ (X, Y) \in \mathcal{G}_2$, there exists $\mathbf{S_{XY}}$ s.t. $X \perp_{(\mathcal{V}, \mathcal{E}_2 \setminus \{(X,Y)\})} Y \mid \mathbf{S_{XY}}$ and $X \not\perp\!\!\!\perp Y \mid \mathbf{S_{XY}}$. We show using DAG-isomorphism that this phase cannot delete any edge in $\mathcal{G}^*$ and, then, that it discards all other edges. □

It is well known that the main shortcoming of independence test-based methods is the unreliability of high-order tests, so we must keep sets $\mathbf{S_{XY}}$ as small as possible. Finding small $\mathbf{S_{XY}}$ requires a more sophisticated approach than simply using the set of neighbors of one of X or Y. Actually, the best set is the smallest one that cuts all the paths between X and Y. This can be computed using a min-cut in a max-flow problem (Tian et al., 1998). Although the set computed is then optimal w.r.t. the quality of the independence test, it has a major drawback: computing a min-cut for all possible pairs of nodes (X, Y) is too prohibitive when the graph involves numerous variables. Moreover, pairs of close nodes require redundant computations. An attractive alternative results from the similarities between min-cuts and junction trees: a min-cut separates the MN into several connected components, just as a separator cuts a junction tree into distinct connected components. Hence, we propose a method, based on join trees (Jensen, 1996), which computes the separators for all pairs at the same time in $O(|\mathcal{V}|^4)$, as compared to $O(|\mathcal{V}|^5)$ in (Tian et al., 1998). Although the separators we compute are not guaranteed to be optimal, in practice they are most often small. Here is the key idea: assume $\mathcal{G}_0$ is triangulated and a corresponding join tree $\mathcal{J}_0$ is computed. Then two cases can obtain: i) X and Y belong to the same clique, or ii) they belong to different cliques of $\mathcal{J}_0$. In the second case, let $\mathbf{C_0}, \ldots, \mathbf{C_k}$ be the smallest connected set of cliques such that $X \in \mathbf{C_0}$ and $Y \in \mathbf{C_k}$, i.e., $\mathbf{C_0}$ and $\mathbf{C_k}$ are the nearest cliques containing X and Y. Then any separator on the path $\mathbf{C_0}, \ldots, \mathbf{C_k}$ cuts all the paths between X and Y in $\mathcal{G}_0$ and, thus, can act as an admissible set $\mathbf{S_{XY}}$. Assuming the triangulation algorithm produces separators as small as possible, selecting the smallest one should keep sets $\mathbf{S_{XY}}$ sufficiently small to allow independence tests. As for the first case, there is no separator between X and Y, so the junction tree is helpless. However, we do not need assigning to $\mathbf{S_{XY}}$ all the neighbors of X or Y, but only those that belong to the same biconnected component as X and Y (as only those can be on the chains between X and Y). Such components can be computed quickly by depth first search algorithms. However, as we shall see, the triangulation algorithm we used determines them implicitly.

In order to produce triangulations in polynomial time (determining an optimal one is NP-hard), we used the simplicial and almost-simplicial rules advocated by (Eijkhof et al., 2002) as well as some heuristics. These rules give high-quality triangulations but are time-consuming. So, to speed-up the process, we used incremental triangulations as suggested by (Flores et al., 2003). The idea is to update the triangulation only in the maximal prime subgraphs of $\mathcal{G}_0$ involved in the modifications resulting from LEARNMN's lines 4 and 8. In addition to the join tree, a join tree of max prime subgraphs is thus maintained by the algorithm. It turns out that merging in this tree the adjacent cliques that are linked by a separator containing more than one node of $\mathcal{V}$ precisely produces $\mathcal{G}_0$'s biconnected components.

Once the join tree constructed, extracting for each pair of nodes (X, Y) not belonging to the same clique its minimal separator is achieved by the collect/distribute algorithms below. In the latter, messages $\mathbf{M_{ij}}$ transmitted from a clique $\mathbf{C_j}$ to $\mathbf{C_i}$ are vectors of pairs $(X, \mathbf{S})$ such that the best conditioning sets between nodes Y in $C_i \backslash (C_i \cap C_j)$ and X is $\mathbf{S}$. An illustrative example is given on Figure 1: messages near solid arrows result from COLLECT($\{BCD\}$,$\{BCD\}$) and those in dashed arrows from DISTRIBUTE($\{BCD\}$,$\{BCD\}$,$\emptyset$).

Algorithm COLLECT
Input : pair $(\mathbf{C_i}, \mathbf{C_j})$
Output : message $\mathbf{M_{ij}}$

```
1. $\mathbf{M_{ij}} \leftarrow$ an empty vector message
2. **foreach** neighbor $\mathbf{C_k}$ of $\mathbf{C_i}$ except $\mathbf{C_j}$ **do**
3.   $\mathbf{M_{ik}} \leftarrow$ Collect$(\mathbf{C_k}, \mathbf{C_i})$
4.   **foreach** pair $(X, \mathbf{S})$ in $\mathbf{M_{ik}}$ **do**
5.     **foreach** node $Y$ in $\mathbf{C_i} \backslash (\mathbf{C_i} \cap \mathbf{C_k})$ **do**
6.       **if** $\mathbf{S_{XY}} \neq \mathbf{S}$ **then**
7.         $\mathbf{S_{XY}} \leftarrow \mathbf{S}$
8.     **done**
9.     add $(X, min(\mathbf{C_i} \cap \mathbf{C_k}, \mathbf{S}))$ to $\mathbf{M_{ij}}$
10.  **done**
11. **done**
12. **foreach** node $X$ in $\mathbf{C_i} \backslash (\mathbf{C_i} \cap \mathbf{C_j})$ **do**
13.   add $(X, \mathbf{C_i} \cap \mathbf{C_j})$ to $\mathbf{M_{ij}}$
14. **return** $\mathbf{M_{ij}}$
```

End of algorithm

Algorithm DISTRIBUTE
Input : pair $(\mathbf{C_i}, \mathbf{C_j})$, message $\mathbf{N_{ij}}$
Output :

```
1. **foreach** pair $(X, \mathbf{S})$ in $\mathbf{N_{ij}}$ **do**
2.   **foreach** node $Y$ in $\mathbf{C_i} \backslash (\mathbf{C_i} \cap \mathbf{C_j})$ **do**
3.     **if** $\mathbf{S_{XY}} \neq \mathbf{S}$ **then**
4.       $\mathbf{S_{XY}} \leftarrow \mathbf{S}$
5.   **done**
6. **done**
7. **foreach** neighbor $\mathbf{C_k}$ of $\mathbf{C_i}$ except $\mathbf{C_j}$ **do**
8.   $\mathbf{N} \leftarrow$ empty message vector
9.   **foreach** pair $(X, \mathbf{S})$ in $\mathbf{N_{ij}}$ **do**
10.    add $(X, min(\mathbf{C_i} \cap \mathbf{C_k}, \mathbf{S}))$ to $\mathbf{N}$
11.  call Distribute $(\mathbf{C_k}, \mathbf{C_i}, \mathbf{N})$
12.  $\mathbf{N'} \leftarrow$ message $\mathbf{M_{ik}}$ sent during collect
13.  **foreach** pair $(X, \mathbf{S})$ in $\mathbf{N'}$ **do**
14.    add $(X, min(\mathbf{C_i} \cap \mathbf{C_k}, \mathbf{S}))$ to $\mathbf{N_{ij}}$
15. **done**
```

End of algorithm

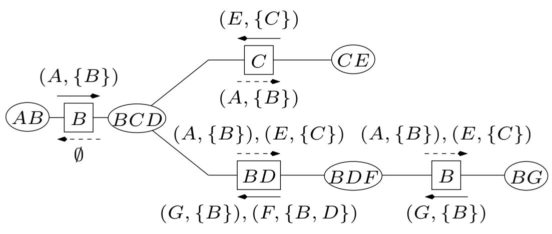

Figure 1: Messages sent during Collect/diffusion.

Because we use a polynomial algorithm to triangulate the maximal prime subgraphs, the complexity of the whole process recovering the optimal MN is also polynomial. Hence, under DAG-isomorphism, we obtain the moral graph of the optimal BN in polynomial time. As the first step of inference algorithms consists in moralizing the BN, and triangulating this moral graph to produce a secondary structure well-suited for efficient computations, the MN we obtain is optimal w.r.t. inference.

4 From the Markov Net to the Bayes Net

Once the MN is obtained, we transform it into a BN $\mathcal{B}_0$ using algorithm MN2BN.

Algorithm MN2BN
Input : UG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output : DAG $\mathcal{B} = (\mathcal{V}', \mathcal{A})$

```
1. $\mathcal{A} \leftarrow \emptyset$ ; $\mathcal{V}' \leftarrow \mathcal{V}$
2. **While** $|\mathcal{V}| > 1$ **Do**
3.   Choose $X \in \mathcal{V}$ in a single clique of $\mathcal{G}$
4.   $\mathbf{E} \leftarrow \{Y \in \mathcal{V} : (Y, X) \in \mathcal{E}\}$
5.   **For all** $Y \in \mathbf{E}$ **Do**
6.     $\mathcal{E} \leftarrow \mathcal{E} \backslash \{(Y, X)\}$ and $\mathcal{A} \leftarrow \mathcal{A} \cup \{(Y \rightarrow X)\}$
7.   $\mathcal{V} \leftarrow \mathcal{V} \backslash \{X\}$
8.   **For all** $(Y, Z) \in \mathbf{E} \times \mathbf{E}$ **Do**
9.     $\mathcal{G} \leftarrow$ DEMORALIZE$(\mathcal{G}, (Y, Z))$
10. **Done**
11. **Return** $\mathcal{B} = (\mathcal{V}', \mathcal{A})$
```

End of algorithm

Algorithm DEMORALIZE
Input : UG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; Edge $(X, Y) \in \mathcal{E}$
Output : UG $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$

```
1. $\mathcal{E}' \leftarrow \mathcal{E} \backslash \{(X, Y)\}$
2. Search $\mathbf{Z} \subset \mathcal{V}$ s.t. $X \perp_{(\mathcal{V}, \mathcal{E}')} Y \mid \mathbf{Z}$
3. **If** $X \perp\!\!\!\perp Y | \mathbf{Z}$ **Then Return** $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$
4. **Else Return** $\mathcal{G}' = (\mathcal{V}, \mathcal{E})$
```

End of algorithm

Before stating the properties of MN2BN, let us illustrate it with an example. Consider the graphs in Figure 2. Assuming a DAG-isomorph distribution, $\mathcal{B}^*$ represents the optimal BN. Suppose we obtained the optimal MN $\mathcal{G}^*$: we now apply MN2BN to it. In $\mathcal{G} = \mathcal{G}^*$, F and R belong to a single clique. Assume MN2BN chooses F. After saving its neighbors set in $\mathcal{G}$ (l.4), F is removed from $\mathcal{G}$ (l.7) as well as its adjacent edges (l.6), which results in a new current graph $\mathcal{G} = \mathcal{G}_1$. The empty DAG $\mathcal{B}$ is altered by adding arcs from these neighbors toward F, hence resulting in the next current DAG $\mathcal{B} = \mathcal{B}_1$. By directing F's adjacent arcs toward F, we may create

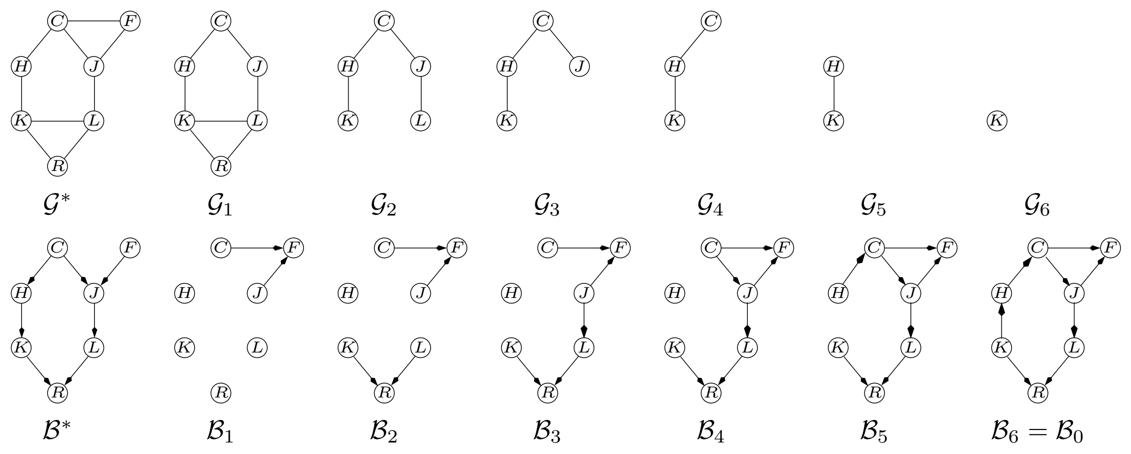

Figure 2: A MN2BN run example

V-structures, hence some of the edges remaining in $\mathcal{G}$ between F's neighbors may correspond to moralization edges and may thus be safely discarded. To this end, for each candidate edge, DEMORALIZE tests whether its deletion would introduce spurious independence in the DAG by searching a separator in the current remaining undirected graph $\mathcal{G}$. If not, we discard it from $\mathcal{G}$. In our example, there is only one candidate edge, (C, J). We therefore compute a set separating C from J in $\mathcal{G}_1$ without (C, J), say K, and test if $C \perp\!\!\!\perp J \mid K$. As this independence does not hold in a distribution exactly encoded by $\mathcal{B}^*$, the test fails and the edge is kept. For the second iteration, only node R belongs to a single clique. The process is similar but, this time, moralization edge (K, L) can be discarded. The algorithm then goes on similarly and finally returns $\mathcal{B}_6 = \mathcal{B}_0$. It is easily seen that $\mathcal{B}_0$ is a DAG which is an I-map of $\mathcal{B}^*$ and that its moral graph is $\mathcal{G}^*$. Note that by mapping $\mathcal{G}^*$ into $\mathcal{B}_6$, not all moralization edges have been identified. For instance, (C, F) was not.

Now, let us explain why MN2BN produces DAGs closely related to the optimal BN we look for.

Proposition 2. *Assuming a DAG-isomorph distribution and sound statistical tests, if* MN2BN *is applied to $\mathcal{G}^*$, the returned graph $\mathcal{B}_0$ is a DAG that is an I-map of $\mathcal{B}^*$, and its moral graph is $\mathcal{G}^*$. Moreover, the algorithm runs in polynomial time.*

Sketch of proof. By induction on $\mathcal{V}$, we prove that, at each iteration, $\mathcal{G}$ is the moral graph of a subgraph of $\mathcal{B}^*$. This entails the existence of a node in a single clique at each iteration. Then we prove that given a DAG $\mathcal{B}$, its moral graph $\mathcal{G}$ and a node X belonging to a single clique of $\mathcal{G}$, there exists a DAG $\mathcal{B}'$ s.t.: i) $\mathcal{B}'$ is an I-map of $\mathcal{B}$, ii) $\mathcal{G}$ is the moral graph of $\mathcal{B}'$ and iii) X has no child in $\mathcal{B}'$. Finally, we prove the proposition by induction on $\mathcal{V}$. □

$\mathcal{B}_0$ encodes at least as many independences as $\mathcal{G}^*$ and possibly more if some moralization edges have been discarded. In the case where MN2BN selects nodes in the inverse topological order of $\mathcal{B}^*$, $\mathcal{B}_0$ is actually $\mathcal{B}^*$. However, this should not happen very often and recovering $\mathcal{B}^*$ requires in general to refine $\mathcal{B}_0$ by identifying all remaining hidden V-structures. Under DAG-isomorph distribution and sound statistical tests, the second phase of GES (Chickering, 2002) is known to transform an I-map of the generative distribution into $\mathcal{B}^*$. Applied to $\mathcal{B}_0$, it can therefore be used as the refinement phase of our algorithm. This one has an exponential complexity but benefits from an anytime property, in the sense that it proceeds by constructing a sequence of BNs sharing the same moral graph and the quality of which converges monotonically from $\mathcal{B}_0$ to $\mathcal{B}^*$. This last phase preserves the GES score-based optimality.

With real-world databases, LEARNMN can fail to recover precisely $\mathcal{G}^*$. In this case, departing from our theoretical framework, we lose our guarantee of accuracy. We also lose the guarantee to always find a node belonging to a single clique. However, MN2BN can easily be modified to handle this situation: if no node is found on line 3, just add edges to $\mathcal{G}$ so that a given node forms a clique with its neighbors. Whatever the node chosen, $\mathcal{B}_0$ is guaranteed to be an I-map of the distribution represented by the input MN. However, the node should be carefully chosen to avoid deviating too much from the MN passed in argument to MN2BN, as each edge addition hides conditional independences.

5 Related works and experiments

In this paper, we have presented an algorithm that exploits attractive features of the MN space to quickly compute an undirected I-map close to the optimal one. This graph is then directed and fed to GES second phase to obtain an optimal BN. The key idea is to learn as much as possible the polynomially accessible information of the generative distribution, namely its undirected part, and postpone to the end the task concentrating the learning computational complexity, namely the V-structure recovery. Most algorithms of the constraint-based approach, like IC (Verma and Pearl, 1990) or PC (Spirtes et al., 2000), deal at some stage with undirected learning but they do not explicitly separate this stage from the directed one. That is, they generally search a power set to extract V-structures information before achieving a whole undirected search. In (Dijk et al., 2003), the two phases are disjoint but the authors are more concerned with finding the skeleton (which is not an I-map) rather than the moral graph. As they only allow CI tests conditioned by a set of size no greater than 1, the search is mostly delegated to a directed score-based search. (Cheng et al., 2002) makes much stronger distribution assumptions.

Like GES, under DAG-isomorphism and sound CI test hypotheses, our method is able to find the optimal BN it looks for. However, these hypotheses probably do not hold in practice. To assess the discrepancy between theory and real world, we performed a series of experiments on classical benchmarks of the Bayes Net Repository[1]. For each BN, we generated by logic sampling 10 databases of size 500, 2000 and 20000. For each database, we ran LEARNMN, GES (the WinMine toolkit software) and a K2 implemented with a BIC score and fed with the original BN topological ordering. K2 is thus already aware of some crucial learning information that LMN and GES must discover. For GES, we recorded as time response only the time required for GES phase 1 but, as the toolkit does not provide the network found at the end of phase 1, we used the better one resulting from phase 2. For both GES and K2, we did not consider directly the output BN but its moral graph, to compare it with the MN output by LMN. Note that the GES toolkit was unable to handle some of the graphs. Both time and quality results reported are means and standard deviations for the 10 databases. The first table shows running times in seconds for LMN and GES (on a 3.06GHz PC). The second one shows for each algorithm the number of edges spuriously added (+) and missed (-) w.r.t. the original BN moral graph (whose number of edges appears in the first row).

[1]`http://compbio.cs.huji.ac.il/Repository`

According to these preliminary tests, we consider our approach as promising. W.r.t GES, we find that LMN is not time-consuming, as GES running time grows much faster than that of LMN when database size increases. As for networks quality, it is noticeable that, for all algorithms, the smaller the database, the weaker the dependences represented in the database and hence the fewer the edges recovered. LMN finds clearly more edges than GES and K2, despite the latter's ordering knowledge which gives it a strong advantage. LMN adds fewer spurious edges than GES but more than K2 (which exploits its ordering knowledge). We think that LMN did not detect some of the unnecessary edges because nodes involved had so many neighbors that χ^2 tests were meaningless, despite our efforts. This suggests alternating multiple phases of edge additions and deletions, so as to avoid situations deteriorating due to big clusters of neighbors. We also should try to tune the χ^2 confidence probability.

References

D. Allen and A. Darwiche. 2003. Optimal time-space tradeoff in probabilistic inference. In *proceedings of IJCAI*, pages 969–975.

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

D.M. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J.

		alarm		barley		carpo		hailfinder		insurance		mildew		water	
size		Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev
20000	LMN	3.65	0.82	9.38	0.73	11.96	2.67	15.56	1.31	3.59	0.1	3.29	0.37	7.48	0.78
	GES	22.36	1.72			75.91	4.19			16.74	4.22			18.36	3.91
2000	LMN	0.98	0.28	1.46	0.46	1.79	0.68	2.36	0.65	1.05	0.12	0.56	0.08	1.37	0.21
	GES	1.36	0			7.27	0.52			1.5	0			1.55	0
500	LMN	0.47	0.15	0.31	0.01	0.71	0.27	1.24	0.39	0.63	0.2	0.28	0.02	0.44	0.12
	GES	0.64	0			0.64	0			0.5	0			0.45	0

		alarm (65)		barley (126)		carpo (102)		hailfinder (99)		insurance (70)		mildew (80)		water (123)	
size		Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev	Avg.	Stdev
20000	GES +	13.6	2.69			26.6	3.85			17.6	5.14			17.9	2.34
	GES -	39.7	1.1			38.7	4.22			44.4	3.17			94.6	2.15
	LMN +	2.1	1.3	15.7	1.1	9.4	1.28	9.7	3	0.4	0.49	3.2	0.75	0	0
	LMN -	9.6	1.11	60.1	2.02	23.2	1.89	10.1	0.94	15.5	1.63	22.3	1.55	52.3	1.85
	K2 +	3.2	0.75	3	0	3.1	1.58	0	0	0	0	1	0	0	0
	K2 -	19.7	1.79	87.8	0.6	31.1	2.43	26	0	19.3	1.1	59	0	71.6	2.58
2000	GES +	14	0.63			11.8	3.6			11.8	2.32			3.9	1.14
	GES -	62	0.77			66.4	2.2			59.2	1.47			110.3	1.73
	LMN +	3.4	0.8	13.3	2.19	7.5	3.35	2.2	1.17	0.3	0.46	4.5	1.2	0.2	0.4
	LMN -	21.3	2.33	79	1.73	33.3	2.61	15.4	1.28	24.9	2.12	40.4	1.2	75	3.87
	K2 +	3.2	0.6	3.1	0.3	2.3	0.78	0	0	0	0	0	0	0	0
	K2 -	34.9	1.37	98.3	0.78	45.7	3.72	50.2	1.83	38.2	1.47	64.4	1.28	98.2	2.48
500	GES +	3.9	1.64			10.3	2.28			5.3	1			0.5	0.5
	GES -	65	0			89.1	2.07			65.1	1.14			118.2	1.25
	LMN +	7.1	2.26	8.2	1.25	5.9	2.12	2	1.26	1.1	0.94	4.4	1.5	0.4	0.92
	LMN -	31.5	2.01	90.2	1.08	54.2	2.52	33.6	3.07	34.7	1.68	56.8	0.98	88.3	2.24
	K2 +	3.7	0.78	3.4	0.49	4.5	2.29	0.1	0.3	0.3	0.46	0	0	0	0
	K2 -	42.7	1	105.2	0.98	60.8	2.44	65.5	1.2	47.2	0.98	74.8	0.75	101.6	1.2

Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer-Verlag.

S. van Dijk, L. van der Gaag, and D. Thierens. 2003. A skeleton-based approach to learning Bayesian networks from data. In *Proceedings of PKDD*.

F. van den Eijkhof, H.L. Bodlaender, and A.M.C.A. Koster. 2002. Safe reduction rules for weighted treewidth. Technical Report UU-CS-2002-051, Utrecht University.

J. Flores, J. Gamez, and K. Olesen. 2003. Incremental compilation of bayesian networks. In *Proceedings of UAI*, pages 233–240.

S.B. Gillispie and M.D. Perlman. 2001. Enumerating Markov equivalence classes of acyclic digraphs models. In *Proceedings of UAI*, pages 171–177.

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

F.V. Jensen. 1996. *An Introduction to Bayesian Networks*. Springer-Verlag, North America.

W. Lam and F. Bacchus. 1993. Using causal information and local measures to learn Bayesian networks. In *Proceedings of UAI*, pages 243–250.

A.L. Madsen and F.V. Jensen. 1999. Lazy propagation: A junction tree inference algorithm based on lazy inference. *Artificial Intelligence*, 113:203–245.

P. Munteanu and M. Bendou. 2001. The EQ framework for learning equivalence classes of Bayesian networks. In *Proceedings of the IEEE International Conference on Data Mining*, pages 417–424.

I. Murray and Z. Ghahramani. 2004. Bayesian learning in undirected graphical models: Approximate MCMC algorithms. In *Proceedings of UAI*.

J. Pearl, D. Geiger, and T.S. Verma. 1989. The logic of influence diagrams. In R.M. Oliver and J.Q. Smith, editors, *Influence Diagrams, Belief Networks and Decision Analysis*. John Wiley and Sons.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman.

P. Pérez. 1998. Markov random fields and images. *CWI Quarterly*, 11(4):413–437.

R.W. Robinson. 1973. Counting labeled acyclic digraphs. In F. Harary, editor, *New directions in Graph Theory*, pages 239–273. Academic Press.

P. Spirtes, C. Glymour, and R. Scheines. 2000. *Causation, Prediction, and Search*. Springer Verlag.

J. Tian, A. Paz, and J. Pearl. 1998. Finding minimal d-separators. Technical Report TR-254, Cognitive Systems Laboratory.

T.S. Verma and J. Pearl. 1990. Causal networks : Semantics and expressiveness. In *Proceedings of UAI*, pages 69–76.