2008－BIO－12 (3)
2008／3／3

Super-Structure: A Decisive Framework to Rationalize and Improve Both Optimal and Heuristic Bayesian Network Structure Learning Algorithms

Eric Perrier perrier@ims.u-tokyo.ac.jp
Seiya Imoto imoto@ims.u-tokyo.ac.jp
Satoru Miyano miyano@ims.u-tokyo.ac.jp

Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan

論文要旨

データに基づくベイジアンネットワークの構造学習において精度向上は，遺伝子ネットワークのような巨大システムをモデル化する際に極めて重要となる．この目的のため，我々は，スコア関数に基づく探索に無向グラフによる super-structure を制約として用いる方法を提案する．つまり，探索する枝は super-structure に含まれているものに限る．さらに，データを用いた super-structure の近似法を導入し，制約付き最適アルゴリズム (COS) を構成する．COS により，最適アルゴリズムはより大きなネットワークに適用可能となる．また，トポロジカル順序に基づく発見的アルゴリズム (PERM) を導出できる．数値実験により，提案するアルゴリズムは，他の方法よりも精度の面で優れていることを示した．

Abstract

Improving the accuracy of Bayesian network learning from data is a decisive challenge to model huge systems such as genes networks. With this end, we propose to constraint the scoring function based algorithms with a super-structure encoded by an undirected graph. This restricts the possible edges of the networks to the ones it contains. Further, we introduce a basic method to approximate a super-structure from data. Then, we develop a constrained optimal search (COS) that extends exact algorithms to sensitively bigger graphs, and a heuristic hill climbing over topological orderings (PERM). Experimentally, these algorithms outperform significantly other approaches.

1 Introduction

It is impossible to understand large raw sets of data obtained from a huge number of correlated variables. Therefore, in order to simplify the comprehension of a system, various graphical models have been developed to summarize interactions between the variables in a synoptic graph. Among the existing models, Bayesian networks have been widely employed for decades in various domains including bioinformatics (Ott et al., 2004). This model is popular because it is robust with respect to noisy or incomplete data, it can easily include prior knowledge, and it is simple to use. Setting the parameters of a Bayesian network from the data is a well known task when its structure is known. Thereby, the bottleneck of modeling an unknown system is to infer the true structure. In fact, any directed acyclic graph (DAG) could be a true representation of the causal relations among the variables.

Despite of their relatively long usage, algorithms devoted to structure learning remain of a poor accuracy, especially when considering huge sets of genes. Those algorithms are regrouped in two distinct approaches: Independency Test (IT) based (Spirtes et al., 2000) and score function based algorithms (Chickering et al., 1995). Both approaches are suffering from severe drawbacks. On one hand, the IT sensitivity to noise leads to accumulate structural errors that greatly affect

the accuracy of the results. On the other hand, since the search space is of a super exponential size, algorithms cannot find the global optima of the score function; hence, they conduct only a greedy search and find locally optimal graphs. However, a recent comparative study carried out by Tsamardinos et al. (2006) showed the superiority of a hybrid algorithm, MMHC, both in terms of structure accuracy and speed. The method consists in approximating the skeleton of the true graph with an IT based technique, MMPC, and then, using this structure to constraint the search space of a classic hill climbing search.

Following this empirical study, we propose to formalize structural constraints over DAGs by defining the concept of a super-structure. This is a flexible constraint that does not fix the skeleton of graphs, but only limits possible edges to the one it contains. In other words, it is a "super-skeleton" and we use it to restrict the search to graphs whose skeleton is covered by it. We give a formal definition of super-structure in Section 3; from MMPC, we derived a method to approximate such structures (Perrier et al.) that we are using in our experiments.

Furthermore, in Section 4, we propose a constrained optimal search algorithm (COS) that globally maximizes the score function over the reduced search space. This algorithm proceeds faster than a general optimal search, since fewer cases have to be considered, which enables its application to bigger networks provided the super-structure is sufficiently sparse. Moreover, if the given constraint covers the true graph, our algorithm returns graphs more accurate than those of an optimal search.

Finally, since the complexity of constrained exact algorithms remains exponential, we develop a heuristic algorithm PERM that enables to consider bigger networks. In fact, given a super-structure constraint, we can define and calculate efficiently the score of a topological ordering as we show in Section 5. Subsequently, the algorithm can conduct a greedy hill climbing over the space of orderings, which enables a wider browsing of the search space since an ordering represents an exponential number of graphs. Experiments confirm the superiority of PERM over every heuristic search for a wide range of networks.

2 Definitions and Preliminaries

We will use upper-case letters to denote random variables (e.g., X_i, V_i) and lower-case letters for the state or value of the corresponding variables (e.g., x_i, v_i). Bold-face will be used for sets of variables (e.g., $\mathbf{Pa}_i$) or values (e.g., $\mathbf{pa}_i$). We will deal only with discrete probability distributions and complete datasets for simplicity. Given a set $\mathbf{X}$ of n random variables, we would like to study their probability distribution P_0. To model this system, we will use Bayesian networks:

Definition 1. (Neapolitan, 2003) *Let P be a discrete joint distribution probability of the random variables in some set $\mathbf{X}$, and $G = (\mathbf{X}, \mathbf{E})$ be a directed acyclic graph (DAG). We call (G, P) a* Bayesian network (BN) *if it satisfies the Markov condition, i.e., each variable is independent of any subset of its non-descendant variables conditioned on its parents.*

We will denote the set of the parents of a variable X_i in a graph G by $\mathbf{Pa}_i$. To study $\mathbf{X}$, we are given a set of data $\mathbf{D}$ following the distribution P_0, and we try to learn a graph G, such that (G, P_0) is a faithful Bayesian network. This means that all and only the conditional independencies true in the distribution P_0 are entailed by the Markov condition applied to G (Neapolitan, 2003). Moreover, in order to learn G, we are given a scoring criterion that evaluates how well the graph fits the data; in our experiments we use the Bayesian information criterion (BIC) (Schwartz, 1978). It is usually costly to evaluate; however, due to the Markov condition, it can be evaluated locally:

$$Score(G, \mathbf{D}) = \sum_{i=1}^{n} score(X_i, \mathbf{Pa}_i, \mathbf{D})$$

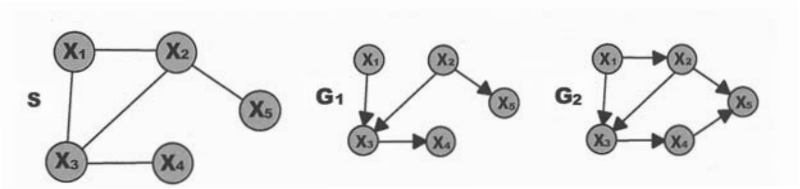

Figure 1: In a search constrained by S, G_1 could be considered since its skeleton is a sub graph of S. Conversely, G_2 is not considered because $\langle X_4, X_5 \rangle \notin \mathbf{E}_S$.

This property is essential to enable efficient calculation, particularly with large graphs. Finally, our task is to maximize the score function over the space of DAGs. In addition, we enunciate the following theorem that is useful to identify the skeleton of G:

Theorem 1. (Spirtes et al., 2000) *In a faithful BN (G, P) on variables $\mathbf{X}$, there is an edge between the pair of nodes X and Y if and only if X depends on Y conditioning on every subset $\mathbf{Z}$ included in $\mathbf{X} \setminus \{X, Y\}$.*

This result is crucial for IT based approach. It implies that we can estimate the skeleton of G by performing conditional independency tests (Tsamardinos et al., 2006) using a significance level α to decide if the value of the test implies independency or dependency.

3 Super-Structure

Since reducing the search space by constraining on the skeleton of considered networks is confirmed to be a decisive step in network learning, we propose to define the concept of a super-structure by:

Definition 2. *An undirected graph $S = (\mathbf{X}, \mathbf{E}_S)$ is said to be a* super-structure *of a DAG $G = (\mathbf{X}, \mathbf{E}_G)$, if the skeleton of G, $(\mathbf{X}, \mathbf{E}_{G'})$ is covered by S (i.e., $\mathbf{E}_{G'} \subseteq \mathbf{E}_S$).*

Further, we propose to constraint algorithms by reducing the search space to the space of DAGs that admit a given super-structure, S, as shown in Figure 1.

Thereby, we separate the problem of structure learning in two sub-problems. First, we have to learn an undirected graph S that contains at least true causal neighborhoods of each variable. Then, given this super-structure, we search for a constrained DAG that maximizes the score function (global optima if it is possible). Finally, an advantage of S is that it can be more easily learnt from data than the true skeleton by using an IT method. This is because extra edges are authorized in a super-structure; hence, higher values of the significance level are allowed, which reduces the rate of false negative discoveries, i.e. missing edges. We derived from MMPC (Tsamardinos et al., 2006) a routine to approximate super-structures in Perrier et al.. This is a basic technique poorly approximating S; however, we use it in our experiments, awaiting better methods that remain to be proposed.

4 Super-Structure Constrained Optimal Search

Here, we propose to constraint the optimal search (OS) of Ott et al. (2004) with a given super-structure $S = (\mathbf{X}, \mathbf{E}_S)$. We will refer to the neighborhood of a variable X_i in S by $\mathbf{N}(X_i)$ that is the set of nodes potentially connected to X_i ($\{X_j \mid \langle X_i, X_j \rangle \in \mathbf{E}_S\}$). Our task is to find a graph that globally maximizes the score function in the reduced search space. For this end, by

using the score locality, we introduce the function $F = (F_s, F_p)$ defined by:

Definition 3. *$\forall X_i \in \mathbf{X}$, and $\mathbf{A} \subseteq \mathbf{N}(X_i)$, F_s (and F_p) are defined by:*

$$\begin{aligned} F_s(X_i, \mathbf{A}) &= \max_{\mathbf{B} \subseteq \mathbf{A}} score(X_i, \mathbf{B}, \mathbf{D}) \\ F_p(X_i, \mathbf{A}) &= \operatorname*{argmax}_{\mathbf{B} \subseteq \mathbf{A}} score(X_i, \mathbf{B}, \mathbf{D}) = \mathbf{B}^* \end{aligned}$$

In other words, F_s returns the best score possible for X_i, considering any possible parent sets, and F_p gives this precise optimal parent set, $\mathbf{B}^*$, over subsets of $\mathbf{A}$. With a super-structure constraint, F needs only to be defined on the neighborhood of each X_i and not on $\mathbf{X} \setminus \{X_i\}$. Further, F can be calculated recursively by using the following formulas:

$$F_s(X_i, \emptyset) = score(X_i, \emptyset) \quad \text{and} \quad F_p(X_i, \emptyset) = \emptyset \tag{1}$$

$$F_s(X_i, \mathbf{A}) = \max(score(X_i, \mathbf{A}), \max_{X_j \in \mathbf{A}} (F_s(X_i, \mathbf{A} \backslash \{X_j\})) \tag{2}$$

$$F_p(X_i, \mathbf{A}) = \operatorname{argmax}(score(X_i, \mathbf{A}), \max_{X_j \in \mathbf{A}} (F_s(X_i, \mathbf{A} \backslash \{X_j\})) \tag{3}$$

Therefore, the total cost of calculating F is lower than $O(n2^m)$, where m is the maximal degree of S (i.e., $m = \max_{X_i \in \mathbf{X}} |\mathbf{N}(X_i)|$); it can be considered as linear if m is bounded by a small constant. This is a decisive improvement compared to OS that needs $O(n2^{n-1})$ dynamical steps. Moreover, to guarantee acyclic graphs, we must build a graph G such that there exists at least one topological ordering w over $\mathbf{X}$ for G. To do this we define the function $M = (M_s, M_g)$ by:

Definition 4. *$\forall \mathbf{A} \subseteq \mathbf{X}$, let the following functions be*

$$\begin{aligned} M_s(\mathbf{A}) &= \max_N Score(N) \text{ ,with } N = (\mathbf{A}, \mathbf{E}) \text{ a DAG on } \mathbf{A} \\ M_g(\mathbf{A}) &= \operatorname*{argmax}_N Score(N), \text{ with } N \text{ as before} \end{aligned}$$

By using these functions, an optimal graph is simply referred by $M_g(\mathbf{X})$, and its score is $M_s(\mathbf{X})$. The main advantage of these functions is that they can be calculated recursively by using the following initialization:

$$\forall X_i \in \mathbf{X} : \begin{array}{l} M_s(\{X_i\}) = score(X_i, \emptyset) \\ M_g(\{X_i\}) = (\{X_i\}, \emptyset) \text{ , the graph containing only } X_i \end{array} \tag{4}$$

While considering $\mathbf{A} \subseteq \mathbf{X}$, with $|\mathbf{A}| = k > 1$, we should find the last element of an optimal ordering over $\mathbf{A}$, X_{i^*} in order to derive the recursive formulas. This element cannot be a parent of any other node in $\mathbf{A} \backslash \{X_{i^*}\}$ but can have all of them as parents. Hence, it is derived as follows:

$$X_{i^*} = \operatorname*{argmax}_{X_j \in \mathbf{A}} (F_s(X_j, \mathbf{B}_j) + M_s(\mathbf{B}_j)) \quad \text{with } \mathbf{B}_j = \mathbf{A} \backslash \{X_j\} \tag{5}$$

In other words, X_{i^*} is the element among every X_j in $\mathbf{A}$ that is the best for placing at the bottom in a topological ordering on $\mathbf{A}$. For more details, lectors should refer Ott et al. (2004). From this equation, we can conclude that:

$$M_s(\mathbf{A}) = F_s(X_{i^*}, \mathbf{B}_{i^*}) + M_s(\mathbf{B}_{i^*}) \tag{6}$$

$$\begin{aligned} M_g(\mathbf{A}) = {} & (\mathbf{A}, \mathbf{E}) \text{ with the edge } (X_a, X_b) \in \mathbf{E} \text{ if and only if:} \\ & - (X_a, X_b) \in \mathbf{E}^* \text{ with } M_g(\mathbf{B}_{i^*}) = (\mathbf{B}_{i^*}, \mathbf{E}^*) \text{ an optimal graph on } \mathbf{B}_{i^*} \\ & - \text{Or } (X_a, X_b) = (X_j, X_{i^*}), \text{ with } X_j \in F_p(X_{i^*}, \mathbf{B}_{i^*}) \end{aligned} \tag{7}$$

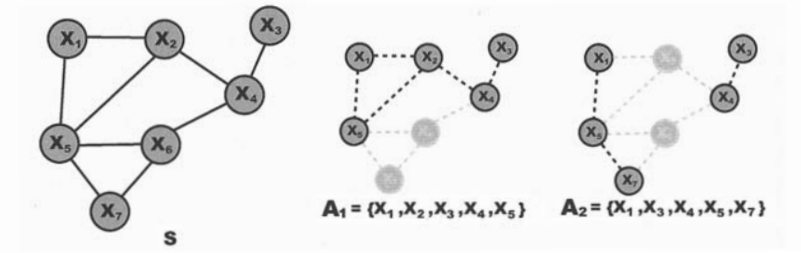

Figure 2: Given S, $\mathbf{A}_1$ is a connected subset, while $\mathbf{A}_2$ is not, because for example, there is no path in $\mathbf{A}_2$ between X_7 and X_4.

Here, $\mathbf{B}_{i^*}$ is equal to $\mathbf{A}\backslash\{X_{i^*}\}$. Moreover, it is not necessary to calculate explicitly M for every $\mathbf{A}$. To show this, we need to define the notion of *connectivity*.

Definition 5. *Given an undirected graph $S = (\mathbf{X}, \mathbf{E}_S)$, we say that a subset $\mathbf{A} \subseteq \mathbf{X}$ is a* connected subset, *or* connex, *if and only if $\mathbf{A} \neq \emptyset$ and $\forall (X_i, X_j) \in \mathbf{A}^2$, $X_i = X_j$, or there exist an undirected path of nodes in $\mathbf{A}$ between X_i and X_j (i.e., there exist $[X_{i_1}, \cdots, X_{i_p}]$ a list of elements in $\mathbf{A}$, with $p \geq 2$, $X_{i_1} = X_i$ and $X_{i_p} = X_j$ such that $\forall k$ in $[1, \cdots, p-1]$, $\langle X_{i_k}, X_{i_{k+1}} \rangle \in \mathbf{E}_S$, cf. Figure 2).*

In fact, when $\mathbf{A}$ is unconnected, we can simply derive the value of M from the results of $\mathcal{C}(\mathbf{A}) = \mathbf{C}_1, \cdots, \mathbf{C}_p$ that is the *family of maximal connected subsets* of $\mathbf{A}$ (Perrier et al.). This result is summarized by the following formulas:

$$M_g(\mathbf{A}) = (\mathbf{A}, \mathbf{E}), \text{ with } \mathbf{E} = \bigcup_{i=1}^{p} \mathbf{E}_i \text{ where } M_g(\mathbf{C}_i) = (\mathbf{C}_i, \mathbf{E}_i) \tag{8}$$

$$M_s(\mathbf{A}) = \sum_{i=1}^{p} M_s(\mathbf{C}_i) \tag{9}$$

Consequently, only and exactly the connected subsets of $\mathbf{X}$ ($Con(S)$) are needed to calculate $M_g(\mathbf{X})$ that is an optimal constrained DAG. We can now present the pseudo code of COS (where the underlined parts correspond to what have been changed from a classic OS):

Algorithm 1 (COS).

(a*) Initialize $\forall X_i \in \mathbf{X}$, $F_s(X_i, \emptyset)$ and $F_p(X_i, \emptyset)$ with (1)

(b*) For each $X_i \in \mathbf{X}$
  For $k = 1$ to $\underline{|\mathbf{N}(X_i)|}$ and each $\mathbf{A} \subseteq \underline{\mathbf{N}(X_i)}$ with $|\mathbf{A}| = k$
    Calculate $\underline{F_s(X_i, \mathbf{A})}$ and $\underline{F_p(X_i, \mathbf{A})}$ using (2) and (3)

(c*) Initialize $\forall X_i$, $M_s(\{X_i\})$ and $M_g(\{X_i\})$ [and $M_w(\{X_i\})$] using (4)

(d*) For $k = 2$ to n and $\underline{\forall \mathbf{A} \in Con(S)}$ with $|\mathbf{A}| = k$
  Select X_{i^*} in $\mathbf{A}$ $\underline{\text{using (5) and (9)}}$
  Then, define $M_s(\mathbf{A})$ and $\underline{M_g(\mathbf{A})}$ using (6), $\underline{(9)}$, (7), and $\underline{(8)}$

(e*) Return $M_g(\mathbf{X})$ $\underline{\text{using (8) if needed}}$

Therefore, the total complexity of Algorithm 1 is simply $O(n2^m + |Con(S)|)$ instead of $O((\frac{n}{2}+1)2^n)$. In Perrier et al. we defined a tree-like structure H that represents efficiently and unambiguously $Con(S)$. We derived as well an upper bound and the average behavior of $|Con(S)|$ depending on S. The complexity of COS remains exponential; however bigger networks can be considered if their structure is enough sparse.

5 PERM: A Greedy Search Over Orderings

Despite the fact that optimality cannot be achieved for huge networks, there is still some optimal results that can be calculated for any network size when constraining with a super-structure S. In fact, since $\mathbf{Pa}_i \subseteq \mathbf{N}(X_i)$, it is possible to calculate F in pre-processing for a total cost bounded by $O(n2^m)$ if the maximal degree m of S is not too big. From this, we define the score of a topological ordering w by:

$$Score(w) = \sum_{i=1}^{n} F_s(\mathbf{Pred}_w(X_i) \cap \mathbf{N}(X_i), X_i) \tag{10}$$

Here, $\mathbf{Pred}_w(X_i)$ represents the set of variables that precede X_i in the ordering w, that is $\{X_j \text{ such that } w(X_j) < w(X_i)\}$. Then, we can obtain the optimal constrained graph $G^*_w = (X, \mathbf{E}^*_w)$ that admits w as topological ordering, by defining the parent set of each variable on the following manner:

$$\mathbf{Pa}_i = \{X_j \text{ such that } (X_j, X_i) \in \mathbf{E}^*_w\} = F_p(\mathbf{Pred}_w(X_i) \cap \mathbf{N}(X_i), X_i) \tag{11}$$

By using (10), it is possible to calculate in linear time the score of an ordering, and thus, to compare efficiently orderings. Therefore we develop a greedy hill climbing search over the space of orderings (PERM) that proceeds in the following manner. It starts by calculating F and selects a randomly generated ordering w_0. At each step, it considers every ordering $w_{i,j}$ obtained from w_0 after swapping two elements, i.e., $w_{i,j} = \tau_{i,j} \circ w_0$, where $\tau_{i,j}$ swaps i and j, and $\circ$ is the operator of composition. If a swap τ^* improves the score of w_0, it updates $w_0 = \tau^* \circ w_0$ and continues the search. Otherwise it returns $G^*_{w_0}$ by using (11). In the following implementation, $\Delta_{i,j}$ is the score difference between $w_{i,j}$ and w_0, and Δ_{max} is $\max_{i,j} \Delta i, j$.

Algorithm 2 (PERM).

(a) $\forall X_i \in \mathbf{X}$ and $\forall \mathbf{A} \subseteq \mathbf{N}(X_i)$ calculate $F(X_i, \mathbf{A})$ using (1), (2), and (3)

(b) Initialize w_0 randomly

(c) For $i = 1$ to n and for $j = i + 1$ to n: calculate $\Delta_{i,j}$ using (10)

(d) If $\Delta_{max} > 0$, then $w_0 = \tau^* \circ w_0$ and repeat from step (c)

(e) Otherwise return $G^*_{w_0}$ using (11)

Algorithm 2 is a typical hill climbing search; however, it is original by the fact that the search space is the one of orderings and not the one of DAGs. Since an ordering represent an exponential number of DAGs, it is more powerful than other greedy hill climbing algorithms because the search space is more widely and efficiently browsed. Moreover, the resulting graph is optimal for its topological ordering. Finally, nothing can be asserted about the complexity of PERM, with the exception that the algorithm ends since the score of w_0 is strictly improved at each step, and that the step (c) requires $O(n^3)$ memory accesses.

6 Results and Conclusion

In the following experiments, we consider five algorithms, including OS (Ott et al., 2004), our structure constrained optimal search COS (Algorithm 1), state-of-the-art hill climbing HC, PERM (Algorithm 2), and MMHC (Tsamardinos et al., 2006). MMHC was selected because it appears to be actually the most efficient greedy search in terms of speed and accuracy. We compare these five algorithms by using three criteria. The first one is the score of the resulting graph (here we use BIC). The second one is the Structural Error Ratio (SER) that helps evaluating the

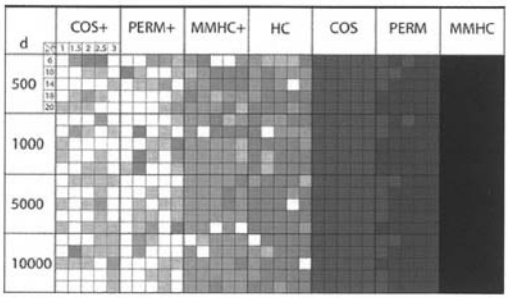

Figure 3: Classification of algorithms by score for every parameters n,$\tilde{m}$ and d.

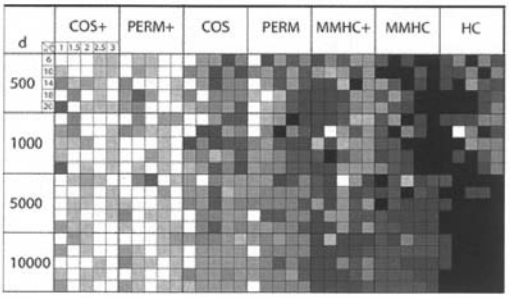

Figure 4: Classification of algorithms by SER for every parameters n,$\tilde{m}$ and d.

accuracy of the resulting graph. To take into account the fact that equivalent graphs have the same score, we build the CPDAGs of both original and learnt DAGs (Chickering, 2002b), and we compare their structure. Thereby, we define the SER as being the number of extra, missing, and wrongly oriented edges divided by the total number of edges in the original graph. In fact, we decided to penalize wrongly oriented edges only by half, because we consider that errors in the skeleton are more "grave" than those in edges orientation. Finally, we measure the time of execution of each algorithm to estimate their complexity.

Here, we refer to the average degree of a graph by $\tilde{m}$. Given n and $\tilde{m}$, we generate a random ordering of the n variables, and on this basis, we build a DAG that admits it as topological ordering by randomly adding $\lfloor \frac{n\tilde{m}}{2} \rfloor$ edges to obtain the correct average degree. For simplicity, we decided to consider only discreet variables, and practically all of them are Boolean. Consequently, to obtain Bayesian networks, for each variable X_i, we generate conditional probabilities $P(X_i = 0 | \mathbf{Pa}_i = \mathbf{pa}_i^k)$ for every possible $\mathbf{pa}_i^k$ by choosing a random real number in $]0,1[$. Finally, a set of d data is artificially generated from the Bayesian network, by following its entailed probability distribution. On this manner, for every $n \in [6, 10, 14, 18, 20]$, $\tilde{m} \in [1, 1.5, 2, 2.5, 3]$, $d \in [500, 1000, 5000, 10000]$, we generated 30 different models, tested every algorithm and averaged the criteria. Thus, in total we considered 3000 different models.

Moreover, since the significance level we used in our experiments is low ($\alpha = 0.05$), the approximated super-structure is lacking some true edges. This implies that the score of constrained algorithms is penalized by errors in the pre-processing phase. That is why we extend every constrained algorithm by adding a post-processing phase. It consists in applying an unconstrained greedy hill climbing starting from the results of these algorithms. We name MMHC+, PERM+ and COS+ these post-processed versions. Such process usually improves the score but not always the accuracy.

Due to the shortage of place, we cannot report all the results. Therefore, we chose to summarize the results by the synoptic Figures 3 and 4. To make these Figures, for every triplet $(n, \tilde{m}, d)$ we ranked the algorithms by their score (respectively, their SER), the first one being the one that maximized the score (respectively, minimized the SER). OS was not considered since it is always the best algorithm. When two algorithms were equal, they were given the same rank j, the next rank becoming $j + 2$. At last, we represent visually these 100 rankings by affecting a color to each rank: the clearer being for the best algorithm, and the darker, for the worst. Thereby, we can quickly compare the quality of algorithms by watching these figures, and see how parameters affect them.

Following our experiments, in general the classification of the algorithms by their score is:

OS, COS+, PERM+, MMHC+, HC, COS, PERM, and MMHC. Similarly, concerning SER, the most frequent order is: OS, COS+, PERM+, COS, PERM, MMHC+, MMHC, and HC. The first interesting remark that we can notice is that there is not a strict link between structure accuracy and score. For example, HC returns nearly always graphs having a higher score than COS or PERM. This is because, as we said, the super-structure was poorly approximated by MMPC, and some of the missing edges could improve the score a lot. However, even with this poor constraint, our algorithms perform better in terms of SER; this means that the super-structure forbids many wrong edges that also improve the score. In fact, IT and scoring criteria have their own sensitivity, and by using both approaches we can augment our chances to find a good model. Moreover, it is also probably due to the fact that for COS, and partially for PERM, the results are constrained optima, which implies that they escaped from many local optima.

Therefore, these experiments illustrate the interest of structural constraint and optimal searches. Further, in terms of complexity, COS is of several order faster than OS, although the latter was given the size of the maximal parent set for each graph. It can be used safely for networks containing about forty variables. However, for bigger networks, it is applicable only if the skeleton of the graph is fairly sparse. Conversely, PERM is nearly as fast as MMHC, the fastest search actually in use. Its superiority is confirmed for bigger networks, despite the fact that the SER is slightly increasing when n augment. Unfortunately, it is not feasible yet when the approximated super-structure has a too high maximal degree m. We are actually thinking about other greedy strategies that could improve frther our results, and also, about new and more efficient methods to learn a super-structure.

References

D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, pages 445–498, 2002b.

D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128, 1995.

R. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

S. Ott, S. Imoto, and S. Miyano. Finding optimal models for small gene networks. *Pacific Symposium on Biocomputing*, 9:557–567, 2004.

E. Perrier, S. Imoto, and S. Miyano. Finding optimal bayesian network given a super-structure. submitted.

G. Schwartz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, second edition, 2000.

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