Learning Bayesian network structures by estimation of distribution algorithms : An experimental analysis

Gregory Thibault, Stéphane Bonnevay, Alexandre Aussem

Learning Bayesian network structures by estimation of distribution algorithms : An experimental analysis

Thibault Grégory
Université de Lyon,
LIRIS-LIESP, Université Lyon 1,
F-69622 Villeurbanne France
gregory_thibault@yahoo.fr

Bonnevay Stéphane
Université de Lyon,
LIRIS, Université Lyon 1,
F-69622 Villeurbanne France
bonnevay@univ-lyon1.fr

Aussem Alexandre
Université de Lyon,
LIESP, Université Lyon 1,
F-69622 Villeurbanne France
aaussem@univ-lyon1.fr

Abstract— **Learning the structure of a Bayesian network (BN) from a data set is NP-hard. In this paper, we discuss a novel heuristic based on Estimation of Distribution Algorithms (EDA), a new paradigm for Evolutionary Computation that is used as a search engine in the BN structure learning problem. The purpose of this work is to study the parameter setting of the EDA and to fix a "good" set of parameters. For this purpose, the EDA-based procedure is applied on several benchmarks to recover the original structure from data. The quality of the learned structure is assessed using several performance indexes.**

Bayesian networks, evolutionary computation, EDA, machine learning.

I. Introduction

Bayesian networks are probabilistic graphical models that offer a coherent and intuitive representation of uncertain domain knowledge [1]. Formally, BN are directed acyclic graphs (DAG) modeling probabilistic dependencies among variables [2]. The graphical part of BN reflects the structure of a problem while local interactions among neighboring variables are quantified by conditional probability distributions. Learning a Bayesian network from data requires both identifying the model structure $\mathcal{G}$ and identifying the corresponding set of model parameter values. Given a fixed structure, however, it is straightforward to estimate the parameter values. As a result, research on the problem of learning Bayesian networks from data is focused on methods to identifying one or more “good” DAG structures from data. All independence constraints that hold in the joint distribution represented by any Bayesian network with structure $\mathcal{G}$ can be identified from the structure itself under certain conditions known as "faithfulness" (i.e., the d-separations in the DAG identify all and only the conditional independencies in the underlying distribution P). However, the problem of learning the most probable *a posteriori* Bayesian network (BN) from data is worst-case NP-hard [3].

There are two types of BN structure learning methods. Constraint-based (CB) methods search a database for conditional independence relations and constructs graphical structures called "patterns" which represent a class of statistically indistinguishable directed DAGs. Search-and-score methods perform a search in the space of legal structures. Compared to CB methods, they have the advantage of being able to flexibly incorporate users’ background knowledge and dealing with incomplete records in the database (e.g., EM technique). Very recently [4], a new powerful paradigm was proposed for Evolutionary Computation : Estimation of Distribution Algorithms (EDA) were proposed as new population-based stochastic search procedures. They use an explicit probability distribution, i.e., the algorithm directly handles this distribution which is used for the sampling of the search space. These EDA algorithms replace the crossover and mutation operators by learning the probability distribution of the best individuals of each generation and its posterior simulation. In this paper, we discuss in detail the EDA-based algorithm to search in the space of legal BN structures. We then carefully evaluate the strengths and limitations of the method on synthetic data generated from the well known Asia [5], Asia8 [6] and ALARM [7] benchmarks. The method was implemented in Matlab using parts of BNT Toolbox [8] and BNTSLP Toolbox [9].

II. Background

For the paper to be accessible to those outside the domain, we recall first the principle of Bayesian network. A Bayesian network (BN) is a tuple $<\mathcal{G}, P>$, where $\mathcal{G} = <\mathcal{V}, \mathcal{E}>$ is a directed acyclic graph (DAG) with nodes representing the random variables $\mathcal{V}$ and P a joint probability distribution on $\mathcal{V}$. In addition, $\mathcal{G}$ and P must satisfy the Markov condition: every variable, $X \in \mathcal{V}$, is independent of any subset of its non-descendant variables (ND_X) conditioned on the set of its parents $\mathbf{Pa}_i^{\mathcal{G}}$. We denote the conditional independence of the variable X and Y given $\mathbf{Z}$, in some distribution P with $Ind_P(X;Y|\mathbf{Z})$, dependence as $Dep_P(X;Y|\mathbf{Z})$. Using these notations, the Markov condition entails for each variable $X_i \in \mathcal{V} : Ind_P(X_i; ND_X|\mathbf{Pa}_i^{\mathcal{G}})$. The independence constraints implied by the Markov condition necessarily hold in the joint distribution represented by any Bayesian network with structure $\mathcal{G}$. They can be identified by the d-separation criterion of Pearl (1988) [2]. From the Markov condition, it is easy to prove [1] that the joint probability distribution P on the variables on $\mathcal{V}$ can be factored as follows :

$$P(\mathcal{V}) = P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i^{\mathcal{G}}) \quad (1)$$

III. Learning the structure of a BN

CB methods try to recover all the conditional independences in the data. But the number of statistical tests required is exponential with the number of variables. Therefore, most proposals are based on greedy search techniques. The second class of methods use a scoring function to evaluate the quality of network compared to the data. Next, these methods search for the best structure with metaheuristics among all the possible structures. But they are slow to converge. There are many different scores in the literature. The majority of them are sums of a likelihood term and a complexity term penalizing complex structures, as described in the *Penalized Maximum Likelihood* formula :

$$\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ij}\log\frac{N_{ijk}}{N_{ij}} - f(N)\sum_{i=1}^{n} q_i(r_i-1) \quad (2)$$

Indeed, a network representing the data as well as possible would be a directed graph completely connected [10], which would not be acyclic, and the model would be too complex to be used in practice. In practice, for the efficiency of the methods, the score is decomposable, i.e., may be computed locally for each variable. The BIC score [11] used in this work is decomposable.

IV. Estimation of Distribution Algorithms

Metaheuristics are combinatorial optimization algorithms which use a set of heuristics (a simple empirical rule to improve a solution) to solve a given problem. Among these methods, we note the evolutionary metaheuristics which base their draft on concepts inspired by evolutionary biology. The concept of evolution is present in many bio-inspired metaheuristics, e.g., genetic algorithms (GAs) [12] and more recently, estimation of distribution algorithms (EDA) [13].

Algorithm 1 EDA overview

```
function EDA($data$:training_base,$Fit$:fitness_function)
    $D_0 \leftarrow$ initialize distribution
    $l \leftarrow 1$
    repeat                                        ▷ main loop
        $P_l \leftarrow$ sample $M$ individuals from $D_l$
        $Fit$($P_l$,$data$)
        $C_l \leftarrow$ select $N$ individuals from $P_l$ $(N <= M)$
        $D_{l+1} \leftarrow$ estimate next distribution from $C_l$
        $l \leftarrow l + 1$
    until ¬ stopping
    return best solution found
end function
```

We will now explicit a toy-problem of machine-learning as an example to illustrate the principles of EDA, called one-max. The principle of this simple problem is to fill ones into a vector. It is very easy to solve but it make it possible us to gauge the efficiency of EDA.

TABLE I
First step

i	x1	x2	x3	f(x)
1	0	1	0	1
2	0	1	0	1
3	***1***	***0***	***1***	***2***
4	***1***	***0***	***1***	***2***
5	***0***	***1***	***1***	***2***
6	1	0	0	1
$p_0(x)$	0.5	0.5	0.5	
$p_1(x)$	***0.7***	***0.3***	***1***	

We start first with a vector of univariate marginal Bernoulli variables, representing the initial distribution of the individuals within the initial population. This 3-vector must make it possible a uniform generation of the initial population, because our knowledge is null at beginning (*a priori*). Then, from this one, we sample six solutions and we fit them (as see in table I). Next, we keep the three best solutions and re-evaluate the distribution from them. Finally, we reiterate this step until the ED vector is full of ones (table II is the next step).

TABLE II
Second step

i	x1	x2	x3	f(x)
1	***1***	***1***	***1***	***3***
2	***1***	***1***	***1***	***3***
3	***1***	***0***	***1***	***2***
4	1	0	1	2
5	1	0	1	2
6	0	0	1	1
$p_1(x)$	0.7	0.3	1	
$p_2(x)$	***1***	***0.7***	***1***	

In this example, only two or three generations are needed to converge towards the best solution, which proves that the algorithm has effectively extracted the knowledge. Even though the overview of the algorithm is the same, it has several variants including UMDA and PBIL which we will study. The variant which we have seen in the example is UMDA (Univariate Marginal Distribution Algorithm) [13]. The other variant, PBIL (Population Based Incremental Learning) [14] differs when we re-estimate the distribution : it uses both the selected population and the previous distribution in a Hebbian-inspired rule :

$$p_{l+1}(x) = (1-\alpha)p_l(x) + \alpha\frac{1}{N}\sum_{k=1}^{N} x_{k:M}^{l} (\alpha \in [0,1]) \quad (3)$$

In such a way, we are sure that the updating is not too stochastic, whereas with UMDA the noise due to the process' stochasticity might be learned.

V. Application to BNSL

In this section, UMDA and PBIL are applied to the BNSL problem (Bayesian Network Structure Learning) to extend our previsous work with genetic algorithms [15]. The limitations of GA on BNSL have been described by Larrañaga [16]. The algorithm applied to BNSL is similar to the one described in [17], except that a probability value is associated to an edge in the Bayesian network. The solutions can be represented in different ways : [17] suggest a vector representation of the DAG : $[a_{1,2}, a_{1,3}, a_{1,4}, ..., a_{i,(i-1)}, a_{i,(i+1)}, ..., a_{n,(n-1)}]$ where $a_{i,j}$ equals 1 if the DAG has an (i, j) edge. Another vector representation $[a_{1,2}], a_{1,3}, ..., a_{2,3}, a2, 4, ..., a_{(n-1),n}]$ allows us to take into account an order on the variables. Our algorithms was tested in a general case, without *a priori* knowledge, therefore, without any order imposed on the variables.

TABLE III

Adjacency matrix

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

Our representation, more efficient, is the complete connexity matrix of the DAG. In this way, there is not needed to converse an array in a corresponding matrix. That is, the diagonal must be null. The four elements to define for the implementation of the algorithm are :

- the sampling function according to a given distribution,
- the function that estimates the underlying distribution of a population,
- the function selecting the best individuals,
- the evaluation function (fitness).

Since we use the connexity matrix to represent a solution, we must make sure that the individuals generated by the distribution have no cycle. Every generation, we remove all cycles by randomly choosing edges which belong to the cycles. In addition to the type of algorithm, we have determined four other parameters :

- population size,
- selection size, i.e. the number of individuals needed for the estimation of the distribution,
- selection type,
- and the parameter which determines whether or not we use the previous population for the selection step.

The uniform stochastic selection comes from the work on the AG and allows a multiple selection of the best individuals, and occasionally a few bad ones, in a way proportional to the rank of the individual in the population. Its principle is similar to the roulette : the reproduction expectancies are placed on a virtual roulette (or rope), such that the occupied zone is proportional to the expectancy. The roulette is turned a first time, which selects a first individual, then it is incremented by regular steps until it makes a full turn. At each step match a newly selected individual. The value of the increment therefore depends on the number of individuals wanted after the selection (sum of expectancies / size of selection). The expectancy function used here is one of the most popular : $\frac{1}{sqrt(rank)}$ where $rank$ equals 1 for the best individual, i for the i-th best, etc.

VI. Experiments

In this section, we apply the method on small complete synthetic data sets generated from given DAGs. So as to measure the quality of the results according to the EDA parameterization, we have followed an experiment plan of “factorial design” type. This plan consists in executing the set of every possible combination of algorithmic parameters and then to measure the quality of the results. There are therefore seven parameters which we will change. The two first correspond to the entry type which we will present to the algorithm, which is the network used to generate the data, as well as the size of the learning database. The other five correspond to the proper parameterization of the algorithm :

- ALGO : type of algorithm : UMDA or PBIL (with $\alpha = 0.25, 0.50$ or 0.75),
- SPOP : population size : 10 or 25 times the number of nodes in the network,
- TSEL : selection type : elitist or stochastic uniform,
- SSEL : selection size : 10 or 30% of the population,
- USEO : use of the previous population or not.

We notice that each simulation is performed 5 times in order to estimate the standard deviation of the studied criteria (the algorithm is highly stochastic). Finally, the following stoping criteria are used :

- 80 generations maximum,
- stop if there was no improvement in the last 15 generations,
- stop when the convergence rate (defined below) reached 100%,
- stop after 5 minutes (to reduce the maximum time spent for these 2880 simulations).

The testing station has a 2.4 GHz Pentium III processor with 512 Mb of RAM using Matlab.

A. Benchmarks

We will use three classical benchmarks : ASIA networks [5] (8 variables), ASIA8 (8 independent copies of ASIA therefore 64 variables) [5] and ALARM [7] (37 non-binary variables). Asia8 is obtained by tiling eight copies of the smaller Asia network as discussed in [6]. The tiling is performed in a way that maintains the structural and probabilistic properties of the original network in the tiled network.

The size of the learning databases was varied following 3 key values : 500 cases where the classical algorithms always show learning difficulties considering the under-representation of the dependencies, 5000 cases which represents a comfortable size for a good learning and a base of 1000 cases for an intermediate size (under which the results are not conclusive). Data was generated within probabilistic logic sampling method

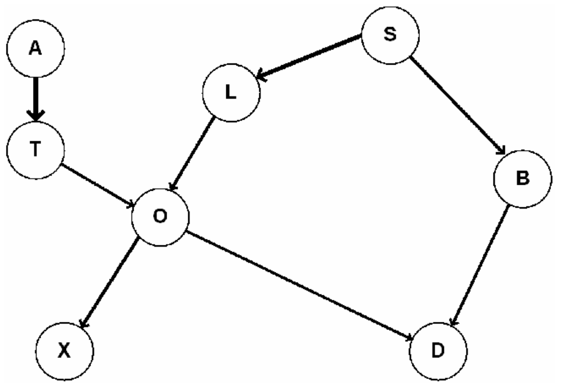

Fig. 1. The Chest-clinic network. A stands for 'visit to Asia', T 'tuberculosis', O 'tuberculosis OR cancer', S 'smoking', B 'bronchitis', D 'dyspnea', X 'X-ray' and L 'lung cancer'.

[18]. Eight criteria are selected to gauge the quality of the reconstructed structure :

- editing distance : the number of edges which must be added, removed or reversed to find the network which generated the data,
- the number of extra edges,
- the number of missing edges,
- the number of badly-oriented edges (note that the editing distance is the sum of the three values following it),
- the number of correct edges,
- the number of iterations at the end of the execution,
- the BIC (**B**ayesian **I**nformation **C**riterion) score of the solution (which is also used as the fitness function),
- the convergence rate of the algorithm defined hereafter.

Let us recall that the distribution is represented by a square probability matrix with zero diagonal (to avoid loops). Each probability is initialized at 0.5. We thereby define $conv$, the convergence function of a probability thus :

$$\begin{array}{llll} conv : & [0,1] & \rightarrow & [0,1] \\ & p & \mapsto & conv(p) = (2 \times (^1/_2 - p))^2 \end{array} \quad (4)$$

By generalizing this function to the set of the probability matrix (*Mx*), we obtain :

$$\begin{array}{llll} conv : & \mathit{Mx} & \rightarrow & [0,1] \\ & M & \mapsto & conv(M) = \frac{\sum_{(i,j)} conv(M_{i,j}) - N}{N \times (N-1)} \end{array} \quad (5)$$

Where N is the size of the graph. Notice that this measure is similarly to a variance computation around 0.5 (instead of the mean). When the convergence rate equals 1, it means that each probability of the matrix equals 0 or 1 and therefore that all the individuals generated are identical (unless they have cycles, which will be eliminated in a stochastic process).

B. Analysis of the results

1) general idea: To analyze the great amount of results, we have compared for each couple (network, base) the 8 criteria for different configurations. For example, we discuss the use of conservation of the population by comparing results with USEO=true and USEO=false.

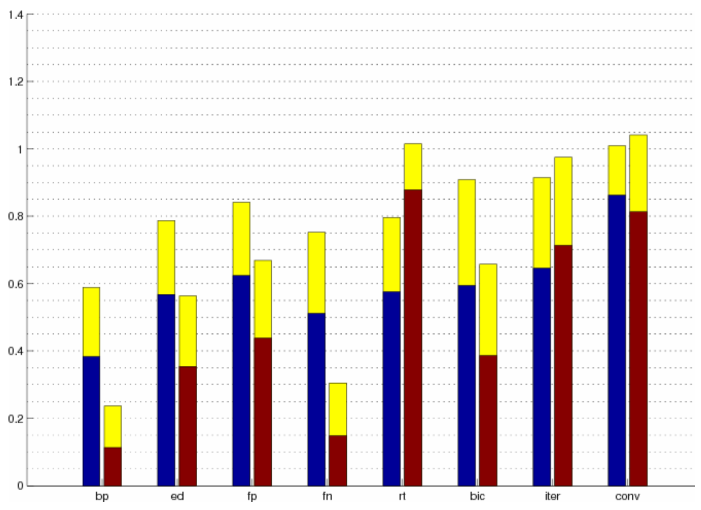

Fig. 2. Example : grouped bars

2) separation: Separation is the step which consists in isolating a set of parameters and to compare the results for the set of combinations of these parameters. For example, knowing that there are 4 different algorithms and 2 selection types, the separation for these two parameters will consist in comparing 8 types of results (because there are 8 couples of possible parameters) on each couple (network, base). For reader-friendly reasons the data output will be drawn as grouped bars : each group represents a criterion and each of these bars a sample of the set of parameters (8 bars in the example). See figure VI-B.2 for an example of separation on TSEL only.

3) normalization, mean computation, standard deviation: In order to compare on a same figure the 8 criteria, they are normalized in the following way : for each couple (network, base), and for each criterion, we determine the best and the worst values (if we want to maximize (e.g. BIC, conv) or minimize (e.g. ED) the criterion). Then, each value is normalized : 1 for the best, 0 for the worst. Even though we lose numerical information in this transformation, we keep the proportions and it is now possible to compare a set of values on different scales. Each simulation is performed 5 times, the height of the bar is proportional to the mean of these 5 results. Moreover, on top of each bar a second bar corresponding to the standard deviation is added. This representation enables us to easily read and compare several combinations of parameters. You can consult in the annex the set of possible figures.

VII. Results

We have first noticed that the type of uniform stochastic selection always gave worse results, whatever the network and the size of the learning base. That's the reason why is prefered using the elitist selection. The next step of analysis consisted in selecting the 'elitist' selection type, i.e. to only be interested in the simulations where the selection was defined as such, then to compare the results again for other parameters. According to this selection, it's now possible to fix the size of selection : the 10% selection always gives better results than the 30% one. We therefore proceeded the same way as above.

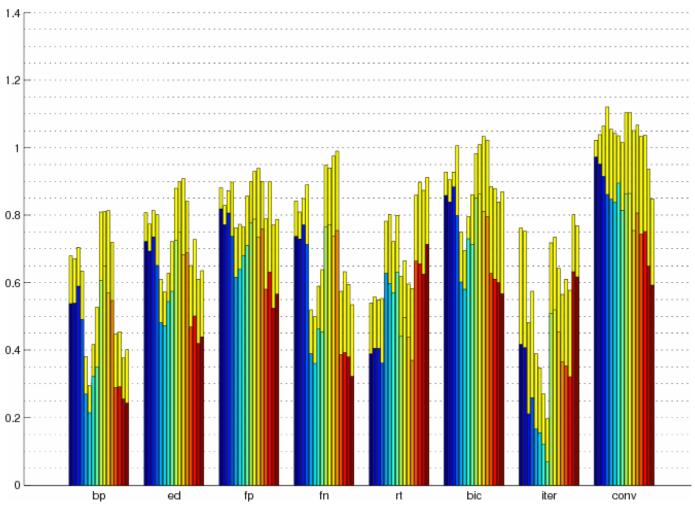

Fig. 3. separation for {SPOP;TSEL;SSEL;USEO}

With the same method, we were not able to fix another parameter which could be determined ; on the other hand, we notice that SPOP, the population size, gives better results on the ASIA and ALARM networks with 500 and 1000 cases when SPOP equals 25 times the number of nodes but that for ALARM with 5000 cases and ASIA8 the results are better with 10 times the number of nodes. Indeed, this coincides with the case where the algorithms stopped because of maximum allocated time. It is therefore justified to think that 10 gives better results and converges faster, then that 25 gives better results later (slower but better convergence). This correlates with our intuition : the bigger the population, the greater the exploration, but on the other hand, the execution time is bigger too. So, we set SPOP to 25. The USEO parameter does not seem to intervene on the algorithm when we analyze the BIC score obtained and the reconstruction measures (editing distance, well-placed edges, etc.). As previously, though, we can deduce a rule by observing the number of iterations and the convergence rate. We notice that on the slow benchmarks (e.g. ASIA8), the first algorithm had the time for 10 to 15% less iterations and has a lower convergence rate than the second algorithm. We deduce that the first algorithm is slow and that it obtains roughly equivalent results with less iteration than the second. Since time is not a barrier in our study, we will set USEO to 'true' and will always use the previous population for the selection (see figure VII).

We still have to determine the algorithm, but the results are not useful : it is impossible to determine a rule. On the other hand, for the slow benchmarks the best to worse algorithms are UMDA, PBIL75, PBIL50 and PBIL25. For the fast benchmarks, it is the opposite. To better analyze these

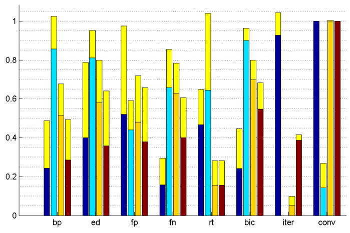

Fig. 4. separation for ALGO when others parameters are fixed

results, a second series of tests, with more allocated time, has been realized, taking the previously fixed parameters as base. After this, we can fix the last parameter : ALGO. PBIL variant with $\alpha = 0.25$ seems to give better results as other variants, as we can see figure VII.

Now, we can explicit the best set of "good" parameters for EDA applied on BNSL :

- TSEL : elitist rather than stochastic uniform,
- SSEL : 10 percent of the population size rather than 30,
- SPOP : 10 times numbers of nodes rather than 25,
- UESO : true,
- ALGO : PBIL25.

The EDA's approaches are shown to perform well as we can see in [17] : the algorithms converge in a few number of generation than GA (not described here). The ASIA network is recovered except the weak link $A \rightarrow T$. The best algorithm was applied on ALARM with 10000 cases in the database. We notice that 32 edges were correctly found on 46, and 9 were reversed. Finally, a total of 41 oriented edges were correctly found in the skeleton, which is a promising result given that the order was supposed unknown.

VIII. Conclusion

In this paper, we proposed an experimental analysis of the behavior of an estimation of distribution algorithm on Bayesian network structure learning problem according to the parameter setting. This study was carried out by an experimental "factorial design" in order to select the best configuration for this particular problem. Comparisons with other scoring and constraint-based methods are currently been undertaken and will be persented shortly.

References

[1] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2004.
[2] T. Verma and J. Pearl, "Causal networks : Semantics and espressiveness," in *Proceedings of the 4th Workshop on Uncertainty in Artificial Intelligence*. Association for Uncertainty in Artificial Intelligence, 1988, pp. 352–359.
[3] D. M. Chickering and C. Meek, "Monotone dag faithfulness : A bad assumption." *Technical Report MSR-TR-2003-16, Microsoft Research, Redmond WA*, 2003.
[4] G. P. H. Mülhenbein, "From recombination of genes to the estimation of distribution i. binary parameters," in *Parallel Problem Solving from Nature*, ser. Lectures Notes in Computer Science 1411, vol. tome PPSN IV, 1996, pp. 178–187.
[5] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application on expert systems," *J. Royal Statistical Society B*, pp. 157–224, 1988.

[6] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing bayesian network structure learning algorithm," *Machine Learning*, vol. 65, no. 1, pp. 31–78, 2006.

[7] I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper, "The alarm monitoring system: a case study with two probabilistic inference techniques for belief networks," in *Second European Conference on Artificial Intelligence in Medicine*, 1989, pp. 247–256.

[8] K. Murphy, "The bayesnet toolbox for matlab," in *Computing Science and Statistics: Proceedings of Interface*, vol. 33, 2001. [Online]. Available: http://www.ai.mit.edu/ murphyk/Software/BNT/bnt.html

[9] P. Leray and O. Francois, "BNT structure learning package: Documentation and experiments," Laboratoire PSI, Tech. Rep., 2004. [Online]. Available: $http : //bnt.insa - rouen.fr/programmes/BNT\_StructureLearning\_v1.3.pdf$

[10] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[11] G. Schwartz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[12] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

[13] G. P. H. Mülhenbein, "From recombination of genes to the estimation of distribution i. binary parameters," in *Parallel Problem Solving from Nature*, ser. Lectures Notes in Computer Science 1411, vol. tome PPSN IV, 1996, pp. 178–187.

[14] S. Baluja, "Population-based incremental learning : A method for integrating genetic search based function optimization and competitive learning," CMU-CS-94-163, Carnegie Mellon University, Tech. Rep., 1994.

[15] G. Thibault and S. Bonnevay, "Recherche de structure d'un réseau bayésien par métaheuristiques," in *Meta'06 : First Workshop on Metaheuristics*, Hammamet, Tunisia, November 2-4 2006.

[16] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Combinatorial optimisation by learning and simulation of bayesian networks," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 343–352.

[17] R. Blanco, I. Inza, and P. Larrañaga, "Learning bayesian networks in the space of structures by estimation of distribution algorithms," *International journal of intelligent systems*, vol. 18, pp. 205–220, 2003.

[18] M. Henrion, "Propagating uncertainty in bayesian networks by probabilistic logic sampling," *Uncertainty in artificial intelligence*, vol. 2, pp. 149–163, 1988.