17

Evolutionary Methods for Learning Bayesian Network Structures

Thierry Brouard, Alain Delaplace and Hubert Cardot
Université Francois-Rabelais de Tours - Laboratoire Informatique
France

1. Introduction

Bayesian networks (BN) are a family of probabilistic graphical models representing a joint distribution for a set of random variables. Conditional dependencies between these variables are symbolized by a Directed Acyclic Graph (DAG). Two classical approaches are often encountered when automaticaly determining an appropriate graphical structure from a database of cases,. The first one consists in the detection of (in)dependencies between the variables (Spirtes et al., 2001; Cheng et al., 2002). The second one uses a scoring metric (Chickering, 2002a). But neither the first nor the second are really satisfactory. The first one uses statistical tests which are not reliable enough when in presence of small datasets. If numerous variables are required, it is the computing time that highly increases. Even if score-based methods require relatively less computation, their disadvantage lies in that the searcher is often confronted with the presence of many local optima within the search space of candidate DAGs. Finally, in the case of the automatic determination of the appropriate graphical structure of a BN, it was shown that the search space is huge (Robinson, 1976) and that is a NP-hard problem (Chickering et al., 1994) for a scoring approach.

In this field of research, evolutionary methods such as Genetic Algorithms – GAs (De Jong, 2006) have already been used in various forms (Larrañaga et al., 1996; Muruzábal & Cotta, 2004; Wong et al., 1999; Wong et al., 2002; Van Dijk et al., 2003b; Acid & De Campos, 2003). Among these works, two lines of research are interesting. The first idea is to effectively reduce the search space using the notion of equivalence class (Pearl, 1988). In (Van Dijk et al., 2003b) for example the authors have tried to implement a genetic algorithm over the partial directed acyclic graph space in hope to benefit from the resulting non-redundancy, without noticeable effect. Our idea is to take advantage both from the (relative) simplicity of the DAG space in terms of manipulation and fitness calculation and the unicity of the equivalence classes' representations.

One major difficulty when tackling the problem of structure learning with scoring methods — evolutionary methods included — is to avoid the premature convergence of the population to a local optimum. When using a genetic algorithm, local optima avoidance is often ensured by preserving some genetic diversity. However, the latter often leads to slow convergence and difficulties in tuning the GA's parameters.

To overcome these problems, we designed a general genetic algorithm based upon dedicated operators: mutation, crossover but also a mutual information-driven repair

Source: Advances in Evolutionary Algorithms, Book edited by: Witold Kosiński, ISBN 978-953-7619-11-4, pp. 468, November 2008, I-Tech Education and Publishing, Vienna, Austria

operator which ensures the closeness of the previous. Various strategies were then tested in order to find a balance between speed of convergence and avoidance of local optima. We focus particularly onto two of these: a new adaptive scheme to the mutation rate on one hand and sequential niching techniques on the other.

The remaining of the chapter is structured as follows: In the second section we will define the problem, ended by a brief state of the art. In the third section, we will show how an evolutionary approach is well suited to this kind of problem. After briefly recalling the theory of genetic algorithms, we will describe the representation of a Bayesian network adapted to genetic algorithms and all the needed operators necessary to take in account the inherent constraints to Bayesian networks. In the fourth section the various strategies will then be developed: Adaptive scheme to the mutation rate on one hand and niching techniques on the other hand. The fifth section will describe the test protocol and the results obtained compared to other classical algorithms. A study of the behaviour of the used strategies will also be given. And finally, the sixth section will present future search in this domain.

2. Problem settings and related work

2.1 Settings

A probabilistic graphical model can represent a whole of conditional relations within a field $X = \{X_1, X_2,\ldots, X_n\}$ of random variables having each one their own field of definition. Bayesian networks belong to a specific branch of the family of the probabilistic graphical models and appear as a directed acryclic graph (DAG) symbolizing the various dependences existing between the variables represented. An example of such a model is given Fig. 1.

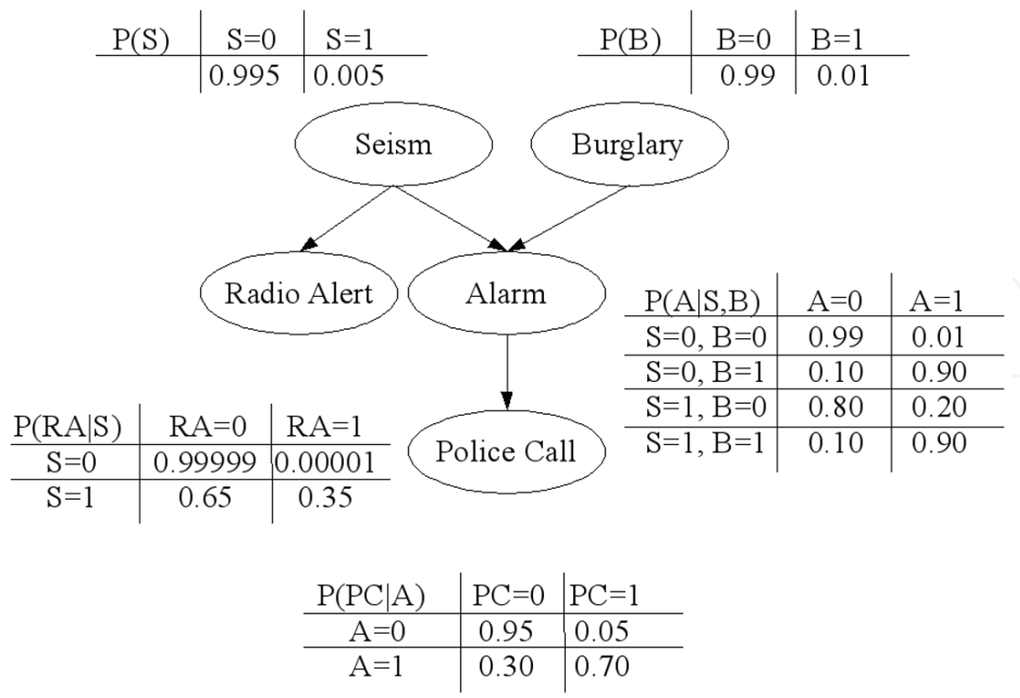

Fig. 1. Example of a Bayesian network.

A Bayesian network is denoted $B = \{G, \theta\}$. Here, $G = \{X, E\}$ is a directed acyclic graph whose set of vertices X represents a set of random variables and its set of arcs E represents the dependencies between these variables. The set of parameters θ holds the conditional probabilities for each vertice, depending on the values taken by its parents in G. The probability $\theta_i = \{P(X_i \mid Pa(X_i))\}$, where $Pa(X_i)$ are the parents of variable X_i in G. If X_i has no parents, then $Pa(X_i) = \emptyset$.
The main convenience of Bayesian networks is that, given the representation of conditional independences by its structure and the set θ of local conditional distributions, we can write the global joint probability distribution as:

$$P(X_1,...,X_n) = \prod_{k=1}^{n} P(X_k \mid Pa(X_k)) \quad (1)$$

2.2. Field of applications of Bayesian networks

Bayesian networks are encountered in various applications like filtering junk e-mail (Sahami et al., 1998), assistance for blind people (Lacey & MacNamara, 2000), meteorology (Cano et al., 2004), traffic accident reconstruction (Davis, 2003), image analysis for tactical computer-aided decision (Fennell & Wishner, 1998), market research (Jaronski et al., 2001), user assistance in sofware use (Horvitz et al. 1998), fraud detection (Ezawa & Schuermann, 1995), human-machine interaction enhancement (Allanach et al., 2004).
The growing interest, since the mid-nineties, that has been shown by the industry for Bayesian models is growing particularly through the widespread process of interaction between man and machine to accelerate decisions. Moreover, it should be emphasized their ability, in combination with Bayesian statistical methods (i.e. taking into account prior probability distribution model) to combine the knowledge derived from the observed domain with a prior knowledge of that domain. This knowledge, subjective, is frequently the product of the advice of a human expert on the subject. This property is valuable when it is known that in the practical application, data acquisition is not only costly in resources and in time, but, unfortunately, often leads to a small knowledge database.

2.3 Training the structure of a Bayesian network

Learning Bayesian network can be broken up into two phases. As a first step, the network structure is determined, either by an expert, either automatically from observations made over the studied domain (most often). Finally, the set of parameters θ is defined here too by an expert or by means of an algorithm.
The problem of learning structure can be compared to the exploration of the data, i.e. the extraction of knowledge (in our case, network topology) from a database (Krause, 1999). It is not always possible for experts to determine the structure of a Bayesian network. In some cases, the determination of the model can therefore be a problem to solve. Thus, in (Yu et al., 2002) learning the structure of a Bayesian network can be used to identify the most obvious relationships between different genetic regulators in order to guide subsequent experiments. The structure is then only a part of the solution to the problem but itself a solution.
Learning the structure of a Bayesian network may need to take into account the nature of the data provided for learning (or just the nature of the modelled domain): continuous variables— variables can take their values in a continuous space (Lauritzen & Wermuth,

1989; Lerner et al. 2001, Cobb & Shenoy, 2006) —, incomplete databases (Lauritzen, 1995; Heckerman, 1995). We assume in this work that the variables modelled take their values in a discrete set, they are fully observed, there is no latent variable i.e. there is no model in the field of non-observable variable that is the parent of two or more observed variables.

The methods used for learning the structure of a Bayesian network can be divided into two main groups:

1. Discovery of independence relationships: these methods consist in the testing procedures on allowing conditional independence to find a structure;
2. Exploration and evaluation: these methods use a score to evaluate the ability of the graph to recreate conditional independence within the model. A search algorithm will build a solution based on the value of the score and will make it evolve iteratively.

Without being exhaustive, belonging to the statistical test-based methods it should be noted first the algorithm PC, changing the algorithm SGS (Spirtes et al. 2001). In this approach, considering a graph $G\ (X, E, \theta)$, two vertices X_i and X_j from X and a subset of vertices $S_{Xi,Xj} \in X\ /\{X_i,X_j\}$, the vertices X_i and X_j are connected by an arc in G if there is no $S_{Xi,Xj}$ such as $(X_i \perp X_j \mid S_{Xi,Xj})$ where $\perp$ denotes the relation of conditional independence. Based on an undirected and fully connected graph, the detection of independence allows us to remove the corresponding arcs until the obtention the skeleton of the expected DAG. Then followed two distinct phases: i) detection and determination of the V-structures[1] of the graph and ii) orientation of the remaining arcs. The algorithm returns a directed graph belonging to the Markov's equivalence class of the sought model. The orientation of the arcs, except those of V-structures detected, does not necessarily correspond to the real causality of this model. In parallel to the algorithm PC, another algorithm, called IC (Inductive Causation) has been developed by the team of Judea Pearl (Pearl & Verma, 1991). This algorithm is similar to the algorithm PC, but starts with an empty structure and links couples of variables as soon as a conditional dependency is detected (in the sense that there is no identified subset conditioning $S_{Xi,Xj}$ such as $(X_i \perp X_j \mid S_{Xi,Xj})$. The common disadvantage to the two algorithms is the numerous tests required to detect conditional independences. Finally, the algorithm BNPC — Bayes Net Power Constructor — (Cheng et al., 2002) uses a quantitative analysis of mutual information between the variables in the studied field to build a structure G. Tests of conditional independence are equivalent to determine a threshold for mutual information (conditional or not) between couples of involved variables. In the latter case, a work (Chickering & Meek, 2003) comes to question the reliability of BNPC.

Many algorithms, by conducting casual research, are quite similar. These algorithms propose a gradual construction of the structure returned. However, we noticed some remaining shortcomings. In the presence of an insufficient number of cases describing the observed domain, the statistical tests of independence are not reliable enough. The number of tests to be independently carried out to cover all the variables is huge. An alternative is the use of a measure for evaluating the quality of a structure knowing the training database in combination with a heuristic exploring a space of options.

Scoring methods use a score to evaluate the consistency of the current structure with the probability distribution that generated the data. Thus, in (Cooper & Herskovits, 1992) a formulation was proposed, under certain conditions, to compute the Bayesian score,

[1] We call V-structure, or convergence, a triplet (x, y, z) such as y depends on x and z $(x \rightarrow y \leftarrow z)$.

(denoted BD and corresponds in fact to the marginal likelihood we are trying to maximize through the determination of a structure G). In (Heckerman et al. 1995a) a variant of Bayesian score based on an assumption of equivalency of likelihood is presented. BDe, the resulting score, has the advantage of preventing a particular configuration of a variable X_i and of its parents $Pa(X_i)$ from being regarded as impossible. A variant, BDeu, initializes the prior probability distributions of parameters according to a uniform law. In (Kayaalp & Cooper, 2002) authors have shown that under certain conditions, this algorithm was able to detect arcs corresponding to low-weighted conditional dependencies. AIC, the Akaike Information Criterion (Akaike, 1970) tries to avoid the learning problems related to likelihood alone. When penalizing the complexity of the structures evaluated, the AIC criterion focuses the simplest model being the most expressive of extracted knowledge from the base D. AIC is not consistent with the dimension of the model, with the result that other alternatives have emerged, for example CAIC - Consistent AIC - (Bozdogan, 1987). If the size of the database is very small, it is generally preferable to use AICC - Akaike Information Corrected Criterion - (Hurvich & Tsai, 1989). The MDL criterion (Rissanen, 1978; Suzuki, 1996) incorporates a penalizing scheme for the structures which are too complex. It takes into account the complexity of the model and the complexity of encoding data related to this model. Finally, the BIC criterion (Bayesian Information Criterion), proposed in (Schwartz, 1978), is similar to the AIC criterion. Properties such as equivalence, breakdown-ability of the score and consistency are introduced. Due to its tendency to return the simplest models (Bouckaert, 1994), BIC is a metric evaluation as widely used as the BDeu score.

To efficiently go through the huge space of solutions, algorithms use heuristics. We can found in the literature deterministic ones like K2 (Cooper & Herskovits, 1992), GES (Chickering, 2002b), KES (Nielsen et al., 2003) or stochastic ones like an application of Monte Carlo Markov Chains methods (Madigan & York, 1995) for example. We particularly notice evolutionary methods applied to the training of a Bayesian network structure. Initial work is presented in (Larrañaga et al., 1996; Etxeberria et al., 1997). In this work, the structure is build using a genetic algorithm and with or without the knowledge of a topologically correct order on the variables of the network. In (Larrañaga et al., 1996) an evolutionary algorithm is used to conduct research over all topologic orders and then the K2 algorithm is used to train the model. Cotta and Muruzábal (Cotta & Muruzábal, 2002) emphasize the use of phenotypic operators instead of genotypic ones. The first one takes into account the expression of the individual's allele while the latter uses a purely random selection. In (Wong et al., 1999), structures are learned using the MDL criterion. Their algorithm, named MDLEP, does not require a crossover operator but is based on a succession of mutation operators. An advanced version of MDLEP named HEP (Hybrid Evolutionary Programming) was proposed (Wong et al., 2002). Based on a hybrid technique, it limits the search space by determining in advance a network skeleton by conducting a series of low-order tests of independence: if X and Y are independent variables, the arcs $X \rightarrow Y$ and $X \leftarrow Y$ can not be added by the mutation operator. The algorithm forbids the creation of a cycle during and after the mutation. In (Van Dijk et al., 2003a, Van Dijk et al., 2003b, Van Dijk & Thierens, 2004) a similar method was proposed. The chromosome contains all the arcs of the network, and three alleles are defined: *none*, $X \rightarrow Y$ and $X \leftarrow Y$. The algorithm acts as Wong's one (Wong et al., 2002) but only recombination and repair are used to make the individuals evolve. The results presented in (Van Dijk & Thierens, 2004) are slightly better than these

obtained by HEP. A search, directly done in the equivalence graph space, is presented in (Muruzábal & Cotta, 2004, Muruzábal & Cotta, 2007). Another approach, where the algorithm works in the limited partially directed acyclic graph is reported in (Acid & De Campos, 2003). These are a special form of PDAG where many of these could fit the same equivalence class. Finally, approaches such as Estimation of Distribution Algorithms (EDA) are applied in (Mühlenbein & Paab, 1996). In (Blanco et al., 2003), the authors have implemented two approaches (UMDA and PBIL) to search structures over the PDAG space. These algorithms were applied to the distribution of arcs in the adjacency matrix of the expected structure. The results appear to support the approach PBIL. In (Romero et al., 2004), two approaches (UMDA and MIMIC) have been applied to the topological orders space. Individuals (i.e. topological orders candidates) are themselves evaluated with the Bayesian scoring.

2.5 Our contribution

For the training of the structure of a Bayesian network with a score function and without prior knowledge like the topology of the structure sought, one often use a greedy search algorithm over the space of structures or in the equivalence classes. But these methods have the disadvantage of being frequently trapped into a solution corresponding to a local optimum of the evaluation function. This is due to the presence of many local optima in space solutions. The smaller the training base is the numerous the optima are. The main reason for a premature convergence is that a greedy algorithm considers, at each moment, only one solution. The search stops if there is no better evaluated solution around a given point. The most widespread technique to avoid this is to use multiple initialization of the greedy algorithm, from very different initial structures and keep the best solution obtained. This technique has the disadvantage to dramatically increase the computing time but also offer no guarantee of obtaining x distinct solutions for x different initialization of the algorithm.

Evolutionary algorithms have two major advantages when processing a problem with many local optima. On the one hand, they allow us to maintain a population of solutions, i.e. several points in the space of solutions. With the maintenance and development of alternatives it becomes possible to reduce the chances of being trapped in a single locally optimum. On the other hand, stochastic behaviour of these methods through the mutation operator can amplify the robustness to local optima attraction (conditionally on the use of parameters and adapted operators) by allowing an exploration of the solutions area which is no longer limited to the immediate neighbourhood of individuals in the population.

3. Genetic algorithm design

Genetic algorithms are a family of computational models inspired by Darwin's theory of Evolution. Genetic algorithms encode potential solutions to a problem in a chromosome-like data structure, exploring and exploiting the search space using dedicated operators. Their actual form is mainly issued from the work of J.Holland (Holland, 1992) in which we can find the general scheme of a genetic algorithm (see Fig. 2) called *canonical GA*. Throughout the years, different strategies and operators have been developed in order to perform an efficient search over the considered space of individuals: selection, mutation and crossing operators, etc.

/* Initialization*/
$t \leftarrow 0$;
Randomly and uniformly generate an initial population P_0 of λ individuals and evaluate them using a fitness function *f*
/* Evolution */
Select P_t individuals for the reproduction
Build new individuals by application of the crossing operator on the beforehand selected individuals
Apply a mutation operator to the new individuals: individuals obtained are affected to the new population P_{t+1}
/* Evaluation */
Evaluate the individuals of P_{t+1} using *f*
$t \leftarrow t + 1$
/* Stop */
If a definite criterion is met then stop else start again the evolution phase

Fig. 2. Holland's canonical genetic algorithm (Holland, 1992)

Applied to the search for Bayesian networks structures, genetic algorithm pose two problems:

- The constraint on the absence of circuits in the structures creates a strong link between the different genes — and alleles — of a person, regardless of the chosen representation. Ideally, operators should reflect this property;
- Often, a heuristic searching over the space of solutions (genetic algorithm, greedy algorithm and so on.) finds itself trapped in a local optimum. This makes it difficult to find a balance between a technique able to avoid this problem, with the risk of overlooking many quality solutions, and a more careful exploration with a good chance to compute only a locally-optimal solution.

If the first item involves essentially the design of a thoughtful and evolutionary approach to the problem, the second point characterizes an issue relating to the multimodal optimization. For this kind of problem, there is a particular methodology: the niching.
We now proceed to a description of a genetic algorithm adapted to find a good structure for a Bayesian network.

3.1 Representation

As our search is performed over the space of directed acyclic graphs, each invidual is represented by an adjacency matrix. Denoting with N the number of variables in the domain, an individual is thus described by an $N \times N$ binary matrix Adj_{ij} where one of its coefficients a_{ij} is equal to 1 if an oriented arc going from X_i to X_j in G exists.
Whereas the traditional genetic algorithm considers chromosomes defined by a binary alphabet, we chose to model the Bayesian network structure by a chain of N genes (where N is the number of variables in the network). Each gene represents one row of the adjacency matrix, that's to say each gene corresponds to the set of parents of one variable. Although this non-binary encoding is unusual in the domain of structure learning, it is not an uncommon practice among genetic algorithms. In fact, this approach turns out to be especially practical for the manipulation and evaluation of candidate solutions.

3.2 Fitness function

We chose to use the Bayesian Information Criterion (BIC) score as the fitness function for our algorithm:

$$S_{BIC}(B,D) = \log\left(L(D|B,\theta^{MAP})\right) - \frac{1}{2}Dim(B)\times\log(N) \quad (2)$$

where D represents the training data, θ^{MAP} the MAP-estimated parameters, and $Dim()$ is the dimension function defined by Eq. 3:

$$Dim(B) = \sum_{i=1}^{n}(r_i - 1)\times \prod_{X_k \in Pa(X_i)} r_k \quad (3)$$

where r_i is the number of possible values for X_i. The fitness function $f(individual)$ can be written as in Eq. 4:

$$f(individual) = \sum_{k=1}^{n} f_k(X_k, Pa(X_k)) \quad (4)$$

where f_k is the local BIC score computed over the family of variable X_k.

The genetic algorithm takes advantage of the breakdown of the evaluation function and evaluates new individuals from their inception, through crossing, mutation or repair. The impact of any change on local an individual's genome shall be immediately passed on to the phenotype of it through the computing of the local score. The direct consequence is that the evaluation phase of the generated population took actually place for each individual, depending on the changes made, as a result of changes endured by him.

3.3 Seting up the population

We choose to initialize the population of structures by the various trees (depending on the chosen root vertice) returned by the MWST algorithm. Although these n trees are Markov-equivalent, the initialization can generate individuals with relevant characteristics. Moreover, since early generations, the combined action of the crossover and the mutation operators provides various and good quality individuals in order to significantly improve the convergence time. We use the undirected tree returned by the algorithm: each individual of the population is initialized by a tree directed from a randomly-chosen root. This mechanism introduces some diversity in the population.

3.4 Selection of the individuals

We use a rank selection where each one of the λ individuals in the population is selected with a probability equal to:

$$P_{select}(individual) = 2\times\frac{\lambda + 1 - rank(individual)}{\lambda\times(\lambda+1)} \quad (5)$$

This strategy allows promote individuals which best suit the problem while leaving the weakest one the opportunity to participate to the evolution process. If the major drawback of this method is to require a systematic classification of individuals in advance, the cost is negligible. Other common strategies have been evaluated without success: the roulette

wheel (prematured convergence), the tournament (the selection pressure remained too strong) and the fitness scaling (Forrest, 1985; Kreinovich et al., 1993). The latter aims to allow in the first instance to prevent the phenomenon of predominance of "super individuals" in the early generations while ensuring when the population converges, that the mid-quality individuals did not hamper the reproduction of the best ones.

3.5 Repair operator

In order to preserve the closeness of our operators over the space of directed acyclic graphs, we need to design a repair operator to convert those invalid graphs (typically, cyclic directed graphs) into valid directed acyclic graphs. When one cycle is detected within a graph, the operator suppresses the one arc in the cycle bearing the weakest mutual information. The mutual information between two variables is defined as in (Chow & Liu, 1968):

$$W(X_A, X_B) = \sum_{x_a, x_b} \frac{N_{ab}}{N} \log\left(\frac{N_{ab} N}{N_a N_b}\right) \tag{6}$$

Where the mutual information $W(X_A,X_B)$ between two variables X_A and X_B is calculated according to the number of times N_{ab} that $X_A=a$ and $X_B=b$, N_a the number of times $X_A=a$ and so on. The mutual information is computed once for a given database. It may happen that an individual has several circuits, as a result of a mutation that generated and/or inverted several arcs. In this case, the repair is iteratively performed, starting with deleting the shortest circuit until the entire circuit has been deleted.

3.6 Crossover operator

A first attempt was to create a one-point crossover operator. At least, the operator used has been developed from the model of (Vekaria & Clack, 1998). This operator is used to generate two individuals with the particularity of defining the crossing point as a function of the quality of the individual. The form taken by the criterion (BIC and, in general, by any decomposable score) makes it possible to assign a local score to the set $\{X_i, Pa(X_i)\}$. Using these different local scores we can therefore choose to generate an individual which received the best elements of his ancestors. This operation is shown Fig. 3.
This generation can be performed only if a DAG is produced (the operator is closed). In our experiments, P_{cross}, the probability that an individual is crossed with another is set to 0.8.

3.7 Mutation operator

Each node of one individual has a P_{mute} probability to either lose or gain one parent or to see one of its incoming arcs reverted (i.e. reversing the relationship with one parent).

3.8 Other parameters

The five best individuals from the previous population are automatically transferred to the next one. The rest of the population at $t+1$ is composed of the $S-5$ best children where S is the size of the population.
Now, after describing our basic GA, we will present how it can be improved by i) a specific adaptive mutation scheme and ii) an exploration strategy: the niching.

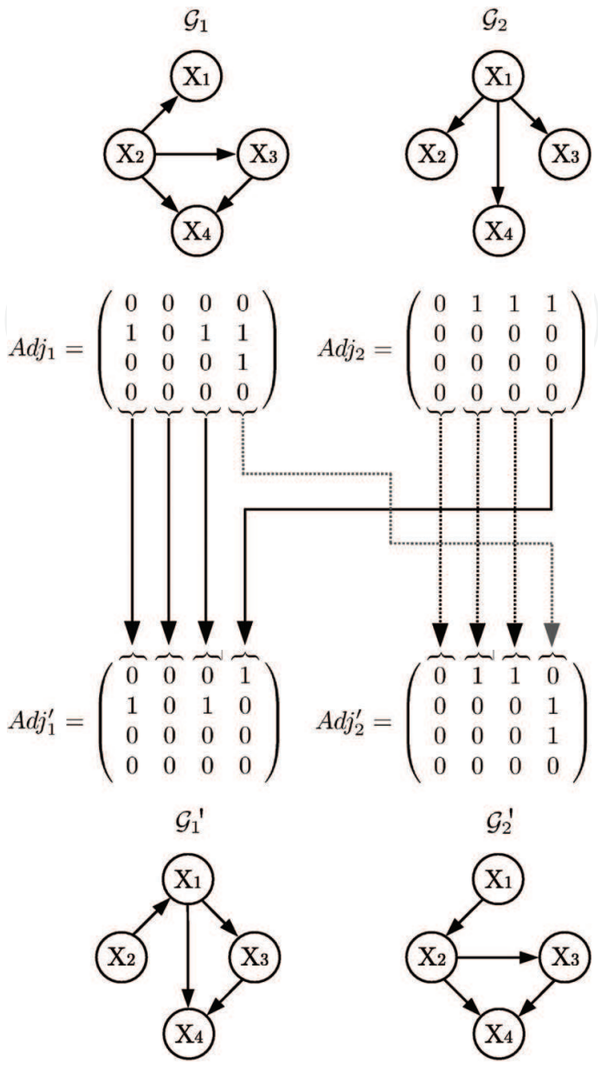

Fig. 3. The crossover operator and the transformation it performs over two DAGs.

4. Strategies

The many parameters of a GA are usually fixed by the user and, unfortunately, usually lead to sub-optimal choices. As the amount of tests required to evaluate all the conceivable sets of parameters will be eventually exponential, a natural approach consists in letting the different parameters evolve along with the algorithm. (Eiben et al., 1999) defines a terminology for self-adaptiveness which can be resumed as follows:

- Deterministic Parameter Control: the parameters are modified by a deterministic rule;
- Adaptive Parameter Control: consists in modifying the parameters using feedback from the search;
- Self-adaptive Parameter Control: parameters are encoded in the individuals and evolve along.

We now present three techniques. The first one, an adaptive parameter control, aims at managing the mutation rate. The second one, an evolutionary method tries to avoid local

optima using a penalizing scheme. Finaly, the third one, another evolutionary method, makes many populations evolve granting sometimes a few individuals to go from one population to another.

4.1 Self-adaptive scheme of the mutation rate

As for the mutation rate, the usual approach consists in starting with a high mutation rate and reducing it as the population converges. Indeed, as the population clusters near one optimum, high mutation rates tend to be degrading. In this case, a self-adaptive strategy would naturally decrease the mutation rate of individuals so that they would be more likely to undergo the minor changes required to reach the optimum.
However, applying this kind of policy can do more harm than good. When there are many local optima, as in our case, we can be confronted with the *bowl effect* described in (Glickman & Sycara, 2000). That is: when the population is clustered around a local optimum and the mutation rate is too low to allow at least one individual to escape this local optimum, a strictly decrementing adaptive policy will only trap the population around this optimum.
Other strategies have been proposed which allow the individual mutation rates to either increase or decrease, such as in (Thierens, 2002). There, the mutation step of one individual induces three differently rated mutations: greater, equal and smaller than the individual's actual rate. The issued individual and its mutation rate are chosen accordingly to the qualitative results of the three mutations. Unfortunately, as the mutation process is the most costly operation in our algorithm, we obviously cannot choose such a strategy. Therefore, we designed two adaptive policies.
The first one is given Fig. 4:

At each mutation process, given one individual I, its fitness value $f(I)$ and its mutation rate $P_{m,\omega} < 1$, $\gamma > 1$:

1. Mutate individual I according to its mutation rate P_m: $(I,P_m) \rightarrow (I')$
2. If $f(I') > f(I)$: allocate mutation rate $\omega \times P_m$ to individual I' and $\gamma \times P_m$ to individual I,
3. If $f(I') \leq f(I)$: allocate mutation rate $\gamma \times P_m$ to individual I' and $\omega \times P_m$ to individual I

Fig. 4. Basic adaptive mutation rate scheme.

This principle is based on the fact that, during an evolution-based process, the less fit individuals have the best chances to produce new, fitter individuals. Our scheme is based on the idea of maximizing the mutation rate of less fit individuals while reducing the mutation rate of the fitter. However, in order to control the computational complexity of the algorithm as well as to leave to the best individuals the possibility to explore their neighbourhood, we define a maximum threshold $Mute_{max}$ and a minimum threshold $Mute_{min}$ for the mutation rate of all individuals. Since we also apply an elitist strategy, we added a deterministic rule in order to control the mutation rate of the best individuals: At the end of each iteration multipliy the mutation rates of the best D individuals by ω where D is the degree of our elitist policy.
An improvement of this approach is now proposed. Indeed, the computed probability concerns all the possible mutation operations. But, perharps some could be benefits, others none. So we propose to conduct the search over the space of solutions by taking into account information on the quality of later searchs. Our goal is to define a probability distribution

which drives the choice of the mutation operation. This distribution should reflect the performance of the mutation operations being applied over the individuals during the previous iterations of the search.
Let us define $P(i,j,op_{mute})$ the probability that the coefficient a_{ij} of the adjacency matrix is modified by the mutation operation opmute. The mutation decays according to the choice of *i*, *j* and *opmute*. We can simplify the density of probability by conditionning a subset of $\{i,j,op_{mute}\}$ by its complementary; this latter being activated according to a static distribution of probability. After studying all the possible combination, we have chosen to design a process to control $P(i \mid op_{mute},j)$. This one influences the choice of the source vertex knowing the destination vertex and for a given mutation operation. So the mutation operator can be rewritten such as shown by Fig. 5.

```
for j = 1 to n do
    if Pa(X_j) mute with a probability P_mute then
        choose a mutation operation among these allowed on Pa(X_j)
        apply op_mute(i, j) with the probability P(i | op_mute ,j)
    end if
end for
```

Fig. 5. The mutation operator scheme

Assuming that the selection probability of $Pa(X_j)$ is uniformly distributed and equals a given P_{mute}, Eq. 7 must be verified:

$$\begin{cases} \sum_{op_{mute}} \delta_{op_{mute}}^{(i,j)} P(i \mid op_{mute}, j) = 1 \\ \delta_{op_{mute}}^{(i,j)} = \begin{cases} 1 \text{ if } op_{mute}(i,j) \text{ is allowed} \\ O \text{ else} \end{cases} \end{cases} \tag{7}$$

The diversity of the individuals lay down to compute $P(i \mid op_{mute},j)$ for each allowed op_{mute} and for each individual X_j. We introduce a set of coefficients denoted $\zeta(i,j,op_{mute}(i,j))$ where $1 \le i,j \le n$ and $i \ne j$ to control $P(i \mid op_{mute},j)$. So we define:

$$P(i \mid op_{mute}, j) = \frac{\zeta(i,j,op_{mute}(i,j))}{\sum \delta_{op_{mute}}^{(i,j)} \zeta(i,j,op_{mute}(i,j))} \tag{8}$$

During the initialisation and without any prior knowledge, $\zeta(i,j,op_{mute}(i,j))$ follows an uniform distribution:

$$\zeta(i,j,op_{mute}(i,j)) = \frac{1}{n-1} \quad \begin{cases} \forall 1 \le i,j \le n \\ \forall\, op_{mute} \end{cases} \tag{9}$$

Finally, to avoid the predominance of a given op_{mute} (probability set to 1) and a total lack of a given op_{mute} (probability set to 0) we add a constraint given by Eq.10:

$$0.01 \le \zeta(i,j,op_{mute}(i,j)) \le 0.9 \quad \begin{cases} \forall 1 \le i,j \le n \\ \forall\, op_{mute} \end{cases} \tag{10}$$

Now, to modify $\zeta(i,j,op_{mute}(i,j))$ we must take in account the quality of the mutations and either their frequencies. After each evolution phase, the $\zeta(i,j,op_{mute}(i,j))$ associated to the op_{mute}

applied at least one time are reestimated. This compute is made according to a parameter γ which quantifies the modification range of $\zeta(i,j,op_{mute}(i,j))$ and depends on ω which is computed as the number of successful applications of op_{mute} minus the number of detrimental ones in the current population. Eq.11 gives the computation. In this relation, if we set γ=0 the algorithm acts as the basic genetic algorithm previoulsy defined.

$$\zeta(i,j,op_{mute}(i,j)) \leftarrow \begin{cases} \min(\zeta(i,j,op_{mute}(i,j)) \times (1-\gamma)^{\omega}, 0.9) \text{ if } \omega > 0 \\ \max(\zeta(i,j,op_{mute}(i,j)) \times (1-\gamma)^{\omega}, 0.01) \text{ else} \end{cases} \quad (11)$$

The regular update $\zeta(i,j,op_{mute}(i,j))$ leads to standardize the $P(i \mid op_{mute},j)$ values and avoids a prematured convergence of the algorithm as seen in (Glickman & Sycara, 2000) in which the mutation probability is strictly decreasing. Our approach is different from an EDA one: we drive the evolution by influencing the mutation operator when an EDA makes the best individuals features probability distribution evolve until then generated.

4.2 Niching

Niching methods appear to be a valuable choice for learning the structure of a Bayesian network because they are well-adapted to multi-modal optimization problem. Two kind of niching techniques could be encountered: spatial ones and temporal ones. They all have in common the definition of a distance which is used to define the niches. In (Mahfoud, 1995), it seemed to be expressed a global consensus about performance: spatial approch gives better results than temporal one. But the latter is easier to implement because it consists in the addition of a penalizing scheme to a given evolutionnary method.

4.2.1 Sequential niching

So we propose two algorithms. The first one is apparented to a sequential niching. It makes a similar trend to that of a classic genetic algorithm (iterated cycles evaluation, selection, crossover, mutation and replacement of individuals) except for the fact that a list of optima is maintained. Individuals matching these optima see their fitness deteriorated to discourage any inspection and maintenance of these individuals in the future.

The local optima, in the context of our method, correspond to the equivalence classes in the meaning of Markov. When at least one equivalence class has been labelled as corresponding to an optimum value of the fitness, the various individuals in the population belonging to this optimum saw the value of their fitness deteriorated to discourage any further use of these parts of the space of solutions. The determination of whether or not an individual belongs to a class of equivalence of the list occurs during the evaluation phase, after generation by crossover and mutation of the new population. The graph equivalent of each new individual is then calculated and compared with those contained in the list of optima. If a match is determined, then the individual sees his fitness penalized and set to at an arbitrary value (very low, lower than the score of the empty structure).

The equivalence classes identified by the list are determined during the course of the algorithm: if, after a predetermined number of iterations Ite_{opt}, there is no improvement of the fitness of the best individual, the algorithm retrieves the graph equivalent of the equivalence class of it and adds it to the list.

It is important to note here that the local optima are not formally banned in the population. The registered optima may well reappear in our population due to a crossover. The

evaluation of these equivalence classes began, in fact until the end of a period of change; an optimum previously memorized may well reappear at the end of the crossover operation and the individual concerned undergo mutation allowing to explore the neighbourhood of the optimum.

The authors of (Beasley et al., 1993) carry out an evolutionary process reset after each determination of an optimum. Our algorithm continues the evolution considering the updated list of these optima. However, by allowing the people to move in the neighbourhood of the detected optima, we seek to preserve the various building blocks hitherto found, as well as reducing the number of evaluations required by multiple launches of the algorithm.

At the meeting of a stopping criterion, the genetic algorithm completes its execution thus returning the list of previously determined optima. The stopping criterion of the algorithm can also be viewed in different ways, for example:

- After a fixed number of local optima detected;
- After a fixed number of iterations (generations).

We opt for the second option. Choosing a fixed number of local optima may, in fact, appear to be a much more arbitrary choice as the number of iterations. Depending on the problem under consideration and/or data learning, the number of local optima in which the evolutionary process may vary. The algorithm returns a directed acyclic graph corresponding to the instantiation of the graph equivalent attached to the highest score in the list of optima.

An important parameter of the algorithm is, at first glance, the threshold beyond which an individual is identified as qu'optimum of the evaluation function. It is necessary to define a value of this parameter, which we call Ite_{opt} that is:

- Neither too small: take it too hasty a class of equity as a local optimum hamper space exploration research of the genetic algorithm, and it amalga over too many optima;
- Nor too high: loss of the benefit of the method staying too long in the same point in space research: the local optima actually impede the progress of the research.

Experience has taught us that Ite_{opt} value of between 15 and 25 iterations can get good results. The value of the required parameter Ite_{opt} seems to be fairly stable as it allows both to stay a short time around the same optimum while allowing solutions to converge around it. The value of the penalty imposed on equivalence classes is arbitrary. The only constraint is that the value is lowered when assessing the optimum detected is lower than the worst possible structure, for example: -10^{15}.

4.2.2 Sequential and spatial niching combined

The second algorithm uses the same approach as for the sequential niching combined with a technique used in parallels GAs to split the population. We use an island model approach for our distributed algorithm. This model is inspired from a model used in genetic of populations (Wright, 1964). In this model, the population is distributed to k islands. Each island can exchange individuals with others avoiding the uniformization of the genome of the individuals. The goals of all of this is to preserve (or to introduce) genetic diversity.

Some additional parameters are required to control this second algorithm. First, we denote I_{mig} the migration interval, i.e. the number if iteration of the GA between two migration phases. Then, we use R_{mig} the migration rate: the rate of individuals selected for a migration.

N_{isl} is the number of islands and finaly I_{size} represents the number of individuals in each island.
In order to remember the local optima encountered by the populations, we follow the next process:

- The population of each island evolves during I_{mig} iterations and then transfert $R_{mig} \times I_{size}$ individuals
- Local optima detected in a given island are registered in a shared list. Then they can be known by all the islands.

5. Evaluation and discussion

From an experimental point of view, the training of the structure of a Bayesian network consists in:

- to have an input database containing examples of instantiation of the variables
- to determine the conditional relationship between the variables of the model
  - Either from statistical tests performed on several subsets of variables;
  - Either from measurements of a match between a given solution and the training database
- to compare the learned structures to determine the respective qualities of the different algorithms used

5.1 Tested methods

So that we can compare with existing methods, we used some of the most-used learning methods: the K2 algorithm, the greedy algorithm applied to the structures space, denoted GS; the greedy algorithm applied to the graph equivalent space, noted GES; the MWST algorithm, the PC algorithm. These methods are compared to our four evolutionary algorithms learning: the simple genetic algorithm (GA); genetic algorithm combined with a strategy of sequential niching (GA-SN); the hybrid sequential-spatial genetic approach (GA-HN); the genetic algorithm with the dynamic adaptive mutation scheme GA-AM.

5.2 The Bayesian networks used

We apply the various algorithms in search of some common structures like: Insurance (Binder et al., 1997) consisting of 27 variables and 52 arcs; ALARM (Beinlich et al. 1989) consisting of 37 variables and 46 arcs. We use each of these networks to summarize:

- Four training data sets for each network, each one containing a number of databases of the same size (250, 500, 1000 & 2000 samples);
- A single and large database (20000 or 30000 samples) for each network. This one is supposed to be sufficiently representative of the conditional dependencies of the network it comes from.

All these data sets is obtained by logic probabilistic sampling (Henrion, 1988): the value of vertices with no predecessors is randomly set, according to the probability distributions of the guenine network, and then the remaining variables are sampled following the same principle, taking into account the values of the parent vertices. We use several training databases for a given network and for a given number of cases, in order to reduce any bias due to sampling error. Indeed, in the case of small databases, it is possible (and it is common) that the extracted statistics are not exactly the conditional dependencies in the

guenine network. After training with small databases, the BIC score of the returned structures by the different methods are computed from the large database mentioned earlier, in order to assess qualitative measures.

5.3 Experiments

GAs: The parameters of the evolutionary algorithms are given in Table 1.

Parameter	**Value**	**Remarks**
Population size	150	
Mutation probability	$1/n$	
Crossover probability	0.8	
Recombination scheme	elitist	The best solution is never lost
Stop criterion	1000 iter.	
Initialisation		See footnote[2]
Ite_{opt}	20	For GA-SN only
γ	0.5	For D1-GA & GA-AM
I_{mig}	20	For GA-HN only
R_{mig}	0.1	For GA-HN only
N_{isl}	30	For GA-HN only
I_{size}	30	For GA-HN only

Table 1. Parameters used for the evolutionary algorithms.

GS: This algorihtm is initialized with a tree returned by the MWST method, where the root vertice is randomly chosen.
GES: This algorithm is initialized with the empty structure.
MWST: it is initialized with a root node randomly selected (it had no effect on the score of the structure obtained).
K2: This algorithm requires a topological order on the vertices of the graph. We used for this purpose two types of initialization:

- The topological order of a tree returned by the MWST algorithm (method K2-T);
- A topological order random (method K2-R).

For each instance of K2-R — i.e. for each training database considered — we are proceeding with $5 \times n$ random initialization for choosing only those returning the best BIC score.
Some of these values (crossover, mutation probability) are coming from some habits of the domain (Bäck, 1993) but especially from experiments too. The choice of the iteration number is therefore sufficient to monitor and interpret the performance of the method considered while avoiding a number of assessments distorting the comparison of results with greedy methods.
We evaluate the quality of the solutions with two criteria: the BIC score from one hand, and a graphic distance measuring the number of differences between two graphs on the other

[2] The populations of the evolutionary methods are all initialized like GS. We make sure, however, that each vertice will be selected at least once as root.

hand. The latter is defined from 4 terms: (D) the total number of different arcs between two graphs G_1 and G_2, (+) the number of arcs existing in G_1 but not in G_2, (-) the number of arcs existing in G_2 but not in G_1 and (inv) the number of arcs inverted in G_1 comparing to G_2. These terms are important because, when considering two graphs of the same equivalence class, some arcs could be inverted. This implies that the corresponding arcs are not oriented in the corresponding PDAG. The consequence is that G_1 and G_2 have the same BIC score but not the same graphic distance. To compare the results with we also give the score of the empty structure G_0 and the score of the reference network G_R.

5.4 Results for the INSURANCE network

Results are given Table 2 & Table 3. The evaluation is averaged over 30 databases. Table 2 shows the means and the standard deviations of the BIC scores. For a better seeing, values are all divided by 10. Values labelled by † are significantly different from the best mean score (Mann-Whitney's test).

The results in Table 2 give an advantage to evolutionary methods. While it is impossible to distinguish clearly the performance of the different evolutionary methods, it is interesting to note that these latter generally outperform algorithms like GES and GS. Only the algorithm GS has such good results as the evolutionary methods on small databases (250 and 500). We can notice too, according to a Mann-Whitney's test that, for large datasets, GA-SN & GA-AM returns a structure close to the original one. Standard deviations are not very large for the GAs, showing a relative stability of the algorithms and so, a good avoidance of local optima.

	250	**500**	**1000**	**2000**
GA	−32135 ± 290	−31200 ± 333	**−29584** ± 359	−28841 ± 89†
GA-SN	−31917 ± 286	−31099 ± 282	−29766 ± 492	**−28681**±156
GA-HN	−31958±246	**−31075** ± 255	**−29428** ± 290	−28715 ± 164
GA-AM	**−31826**±270	−31076 ± 151	−29635 ± 261	−28688 ± 165
GS	−32227 ± 397	−31217 ± 314	−29789 ± 225†	−28865 ± 151†
GES	−33572 ± 247†	−31952 ± 273†	−30448 ± 836†	−29255 ± 634†
K2-T	−32334 ± 489†	−31772 ± 339†	−30322 ± 337†	−29248 ± 163†
K2-R	−33002 ± 489†	−31858 ± 395†	−29866 ± 281†	−29320 ± 245†
MWST	−34045 ± 141†	−33791 ± 519†	−33744 ± 296†	−33717 ± 254†
G_R	−28353			
G_0	−45614			

Table 2. Means and standard deviations of the BIC scores (INSURANCE).

Table 3 shows the mean structural differences between the original network and these delivered by some learning algorithms. There, we can see that evolutionary methods, particularly GA-SN, return the structures which are the closest to the original one. This network was chosen because it contains numerous low-valued conditional probabilities. These are difficult to find using small databases. So even if the BIC score is rather close to the original one, graphical distances reveals some differences. First, we can see that D is

rather high (the original network G_R is made with only 52 arcs, compared to D which minimum is 24.4) even if the BIC score is very close (resp. -28353 compared to -28681). Second, as expected, *D* decreases when the size of the learning database grows, mainly because of the (-) term. Third, GAs obtains the closest models to the original in 11 cases over 16; the 5 others are provided by GES.

	250				**500**			
	D	+	inv	-	D	+	inv	-
GA	39.6	4.4	7.2	28	34	3.1	7.6	23.3
GA-SN	**37**	3.5	7.1	26.4	35.1	3.7	7.4	24
GA-HN	38.1	3.5	7.5	27.1	**33.3**	3	7.3	23
GA-AM	37.5	4.3	6.6	26.6	33.9	3.2	7.7	23
GS	42.1	4.6	9.4	28.1	37.7	4.5	9.4	23.8
GES	39.5	3.7	7.1	28.7	35.1	3	7.1	25
K2-T	42.7	5.1	8.4	29.2	40.8	5.4	8.8	26.6
K2-R	42.4	4.8	7.2	30.4	41.8	6.5	8.8	26.6
MWST	41.7	4	7.7	30	41.3	3.5	8.3	29.5
	1000				**2000**			
	D	+	inv	-	D	+	inv	-
GA	39.6	4.4	7.2	28	27.8	4.7	8	15.1
GA-SN	30.8	3.8	7.4	19.6	**24,4**	3.4	6.7	14.3
GA-HN	**29.3**	3.6	6.5	19.2	26.6	3.6	8.6	14.4
GA-AM	31.4	4	8	19.4	27	4.3	8.4	14.3
GS	35.9	5.1	10	20.8	31.9	5.2	11.4	15.3
GES	32.4	4.1	8.1	20.2	27.5	4	8.4	15.1
K2-T	38.7	5.9	11	21.8	34.6	7.3	10.9	16.4
K2-R	39.6	8.3	8.3	23	36.1	8.5	8.5	9.1
MWST	37.7	1.7	8.3	27.7	36.3	1.2	7.9	27.2

Table 3. Mean structural differences between the original INSURANCE network and the best solutions founded by some algorithms

5.5 Results for the ALARM network

The results are shown Table 4 & Table 5. This network contains more vertices than the INSURANCE one, but less low-valued arcs. The evaluation is averaged over 30 databases. Table 4 shows that evolutionary algorithms obtain the best scores. But while GES provides less qualitative solutions accordingly to the BIC score, these solutions are closest to the original one if we consider the graphical distance. Here, a strategy consisting in gradually building a solution seems to produce better structures than an evolutionary search. In this case, a GA has a huge space (3×10^{237} when applying the Robinson's formula) into which one it enumerates solutions. If we increases the size of the population the results are better than these provided by GES.

	250	**500**	**1000**	**2000**
GA	−36239 ± 335	−34815 ± 317	−33839 ± 159	−33722 ± 204†
GA-SN	−**36094**±297	−34863 ± 346	−33865 ± 203	−33640 ± 196†
GA-HN	−36144 ± 326	−34864 ± 337	**−33723 ± 251**	**−33496 ± 170**
GA-AM	−36104 ± 316	−**34791**±340	−33942 ± 198†	−33722 ± 204†
GS	−36301 ± 309†	−35049 ± 380†	−33839 ± 109†	−33638 ± 964†
GES	−36124 ± 315	−34834 ± 288	−33801 ± 562†	−33593 ± 692†
K2-T	−36615 ± 308†	−35637 ± 328†	−34427 ± 200†	−34045 ± 818†
K2-R	−37173 ± 435†	−35756 ± 264†	−34579 ± 305†	−34128 ± 173†
MWST	−37531 ± 185†	−37294 ± 737†	−37218 ± 425†	−37207 ± 366†
G_R	−33097			
G_0	−63113			

Table 4. Means and standard deviations of the BIC scores (ALARM).

	250				**500**			
	D	+	inv	-	D	+	inv	-
GA	34.2	4.8	13.9	15.5	25.7	4.5	10.2	11
GA-SN	33.1	4.6	13.5	15	25.6	4.2	10.6	10.8
GA-HN	33.6	4.6	13.8	15.2	25.1	3.7	10.7	10.7
GA-AM	33	4.6	13.4	15	26.2	4	11.5	10.7
GS	33.7	5	12.6	16.1	30.2	5	13.5	11.7
GES	**32.5**	4.5	12.7	15.3	**23.3**	3.8	8	11.5
K2-T	34.5	5.1	13.1	16.3	35.1	7.2	15.2	12.7
K2-R	36.5	6.6	10.2	19.6	35	8.7	11.3	11.5
MWST	38.5	6.9	14.7	16.9	36.5	4.7	17.1	14.7
	1000				**2000**			
	D	+	inv	-	D	+	inv	-
GA	19.7	3.7	9	6.9	23	5.3	11.8	5.9
GA-SN	22	4.5	10.4	7.1	20.1	4.1	10.2	5.8
GA-HN	**18.3**	3.3	10.1	4.9	18.9	3.6	9	6.3
GA-AM	27	6.4	13.1	7.4	29	7.4	16	6.3
GS	27.8	6.2	14.5	7.1	25.4	6.2	13.6	5.6
GES	20.2	4.3	8.5	7.3	**17.3**	3.5	8.2	5.6
K2-T	35.4	10.4	15.7	9.3	36.9	12.3	17.4	7.2
K2-R	37.1	11.4	15.1	10.6	40.2	14.6	16.1	9.5
MWST	35.1	4.4	16.3	14.4	34.1	14	16.1	14

Table 5. Mean structural differences between the original ALARM network and the best solutions founded by some algorithms

5.5 Behaviour of the GAs

Now look at some measures in order to evaluate the behaviour of our genetic algorithms.
A repair operator was designed to avoid individuals having a cycle. Statistics computed during the tests show that the rate of individuals repaired does not seem to depend neither on the algorithm used nor and on the size of the training set. It seems to be directly related to the complexity of the network. Thus, this rate is about 15% for the INSURANCE network and about 7% for the ALARM network.
The mean number of iterations before the GA found the best solution returned for the INSURANCE network is given Table 6. The data obtained for the ALARM network are the same order of magnitude. We note here that GA-HN quickly gets the best solution. This makes it competitive in terms of computing time if we could detect this event.

	250	500	1000	2000
GA	364	454	425	555
GA-AM	704	605	694	723
GA-SN	398	414	526	501
GA-HN	82	106	166	116

Table 6. Mean of the necessary number of iterations to find the best structure (INSURANCE).

The averaged computing time of each algorithm is given Table 7 (for the ALARM network). We note here that GA-HN is only three times slower than GES. We note too that these computing times are rather stable when the size of the database increases.

	250	500	1000	2000
GA	3593 ± 47	3659 ± 41	3871 ± 53	4088 ± 180
GA-AM	3843 ± 58	3877 ± 44	4051 ± 59	4332 ± 78
GA-SN	3875 ± 32	4005 ± 43	4481 ± 46	4834 ± 52
GA-HN	9118 ± 269	9179 ± 285	9026 ± 236	9214 ± 244
GS	9040 ± 1866	9503 ± 1555	12283 ± 1403	16216 ± 2192
GES	3112 ± 321	2762 ± 166	4055 ± 3.4	5759 ± 420
K2-T	733 ± 9	855 ± 25	1011 ± 14	1184 ± 8
K2-R	3734 ± 61	4368 ± 152	5019 ± 67	5982 ± 43
MWST	10 ± 1	10 ± 2	11 ± 1	12 ± 1

Table 7. Averaged computing times (in seconds) and standard deviations (ALARM).

An example of the evolution of the fitness of the population is given Fig. 6. The curves for GA, GA-SN and GA-AM are very similar. The curve associated with GA-HN increases through levels, a consequence of spatial niching policy who promptly exchange some

individuals between islands. Although the average quality is progressing more slowly, it is revealed fairly quickly, however, better than in other genetic algorithms. Although the curve corresponding to the algorithm GA seems well placed, Tables 4 and 5 learn us a bit more. First, the score is not considered equivalent: the algorithm GA-HN have the best one. Second, the graphical distance of GA-HN is the lowest. Although GA-SN seems more remote, the results presented in Tables 4 and 5 show that the BIC score obtained by GA-SN is closer to the optimal, and the editing distance of GA-SN is better than the GA one.

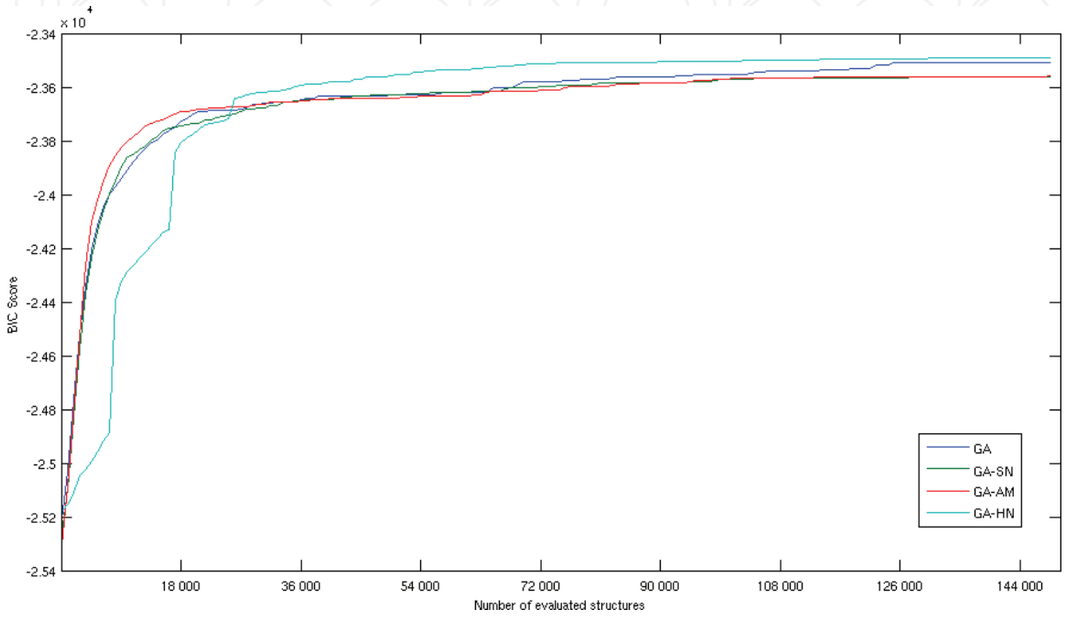

Fig. 6. Evolution of the individuals' fitness (ALARM, 2000 training samples).

6. Future search

We will continue the development of the hybrid niching technique. The first step is the distribution over a cluster of computers. Then we plan to develop new strategies implying a global behaviour like in GA-HN and a dynamic mutation scheme like this one used in GA-AM. The next goal will be the definition of a stopping criterion based on population's statistics to make our algorithm competitive in term of computing time.

7. Conclusion

We have presented three methods for learning the structure of a Bayesian network. The first one consists in the control of the probability distribution of mutation in the genetic algorithm. The second one is to incorporate a scheme penalty in the genetic algorithm so that it avoids certain areas of space research. The third method is to search through several competing populations and to allow timely exchange among these populations. We have shown experimentally that different algorithms behaved satisfactorily, in particular that they were proving to be successful on large databases. We also examined the behaviour of proposed algorithms. Niching strategies are interesting, especially using the spatial one, which focuses quickly on the best solutions.

8. Acknowledgements

This work was realized using Matlab and two toolboxes dedicated to Bayesian networks manipulation: Bayesian Net Toolbox from K. P. Murphy (Murphy, 2001) and Structure Learning Package (SLP) from P. Leray & O. François (Francois & Leray, 2004).

9. References

Acid, S. & De Campos, L.M. (2003). Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, Vol. 18, 05/03, 445-490, 11076-9757

Akaike, H. (1970). Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, Vol. 22, No. 1, 12/70, 203–217, 0020-3157

Allanach, J.; Tu, H.; Singh, S.; Pattipati, K. & Willett, P. (2004). Detecting, tracking and counteracting terrorist networks via hidden markov models, *Proceedings of IEEE Aerospace Conference*, pp. 3257, 0-7803-8155-6, 03/2004

Bäck, T. (1993). Optimal mutation rates in genetic search, *Proceedings of International Conference on Genetic Algorithms*, pp. 2-8, 1-55860-299-2, San Mateo (CA), Morgan Kaufmann

Beasley, D.; Bull, D.R.; & Martin, R.R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, Vol. 1, No. 2, 101–125, 1063-6560

Beinlich, I.A.; Suermondt, H.J.; Chavez, R.M. & Cooper, G.F. (1989). The alarm monitoring system : A case study with two probabilistic inference techniques for belief networks, *Proceedings of European Conference on Artificial Intelligence in Medicine*, pp. 247–256, London, Springer-Verlag, Berlin

Binder, J.; Koller, D.; Russell, S.J. & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, Vol. 29, No. 2-3, 11/97, 213–244, 0885-6125

Blanco, R. ; Inza, I. ; & Larrañaga, P. (2003). Learning bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, Vol. 18, No. 2, 205–220, 0884-8173

Bouckaert, R. (1994). Properties of bayesian belief network learning algorithms, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 102–10, Morgan Kaufmann, San Francisco (CA)

Bozdogan, H. (1987). Model selection and Akaike's information criteria (AIC): The general theory and its analytical extentions. *Psychometrika*, Vol. 52, 354–370, 0033-3123

Cano, R.; Sordo, C.; & Gutiérrez, J. (2004). Applications of Bayesian Networks in Meteorology, In *Advances in Bayesian Networks*, Gámez J.A, Moral S. & Salmerón A. (Eds.), 309-327, Springer, 3540208763

Cheng, J.; Bell, D.A. & Liu, W. (2002). Learning belief networks from data: An information theory based approach. *Artificial Intelligence*, Vol. 137, No. 1-2, 43–90

Chickering, D.M. (2002b). Learning equivalence classes of Bayesian network structures. *Journal of Machine Learning Research*, Vol. 2, 03/02, 445–498, 1532-4435

Chickering, D.M. & Meek, C. (2003). Monotone DAG faithfulness : A bad assumption. Technical Report MSR-TR-2003-16, Microsoft Research

Chickering, D.M. (2002a). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, Vol. 3, 03/03, 507–554, 1532-4435

Chickering, D.M.; Geiger, D. & Heckerman, D. (1994). Learning Bayesian networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research

Chow, C. & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, Vol. 14, No. 3, 05/68, 462-467, 0018-9448

Cobb, B.R. & Shenoy, P.P. (2006). Inference in hybrid bayesian networks with mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, Vol. 41, No. 3, 04/06, 257–286, 0888-613X

Cooper, G. & Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, Vol. 9, No. 4, 10/92, 309–347, 0885-6125

Cotta, C. & Muruzábal, J. (2002). Towards a more efficient evolutionary induction of bayesian networks, *Proceedings of Parallel Problem Solving from Nature*, pp. 730-739, Granada, 09/2002

Davis, G.A. (2003) Bayesian reconstruction of traffic accidents, *Law, Probability and Risk*, Vol. 2, No. 2, 69-89, 1470-8396

De Jong, K. (2006). *Evolutionary Computation: A Unified Approach*, MIT Press, 0262041944

Eiben, A.E.; Hinterding, R. & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, 124-141, 1089-778X

Etxeberria, R.; Larrañaga, P. & Picaza, J.M. (1997). Analysis of the behaviour of genetic algorithms when learning bayesian network structure from data. *Pattern Recognition Letters*, Vol. 18, No. 11-13, 11/97, 1269–1273, 0167-8655

Ezawa, K.J. & Schuermann, T. (1995) Fraud/Uncollectible Debt Detection Using a Bayesian Network Based Learning System: A Rare Binary Outcome with Mixed Data Structures, *Proceedings of Uncertainty in Artificial Intelligence*, pp.157-16, Morgan Kaufmann, San Francisco (CA)

Fennell, M.T. & Wishner, R.P. (1998). Battlefield awareness via synergistic SAR and MTI exploitation. *IEEE Aerospace and Electronic Systems Magazine*, Vol. 13, No. 2, 39-43, 0885-8985

Forrest, S. (1985). Documentation for prisoner's dilemma and norms programs that use the genetic algorithm, Technical Report, Univ. of Michigan.

Francois, O. & Leray, P. (2004). BNT structure learning package: documentation and experiments, Technical Report, Univ. of Rouen (France).

Glickman, M. & Sycara, K. (2000). Reasons for premature convergence of self-adapting mutation rates, *Proceedings of Evolutionary Computation*, pp. 62 – 69, 07/2000

Heckerman, D. (1995a). A tutorial on learning bayesian networks, Technical Report MSR-TR-95-06, Microsoft Research

Heckerman, D.; Mamdani, A. & Wellman, M.P. (1995b). Real world applications of bayesian networks. *Communications of the ACM*, Vol. 38, No. 3, 03/95, 24-30, 0001-0782

Henrion, M. (1988). Propagation of uncertainty by probabilistic logic sampling in bayes networks. *Proceedings of Uncertainty in Artificial Intelligence*, pp. 149–164, Morgan Kaufmann, San Francisco (CA)

Holland, J.H. (1992). *Adaptation in natural and artificial systems*, The MIT Press, 0262581116

Horvitz, E.; Breese, J.; Heckerman, D.; Hovel, D. & Rommelse, K. (1998) The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users, *Proceedings of Uncertainty in Artificial Intelligence*, 07/1998, Morgan Kaufmann, San Francisco (CA).

Hurvich, C.M. & Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, Vol. 76, No. 2, 297-307, 0006-3444

Jaronski, W.; Bloemer, J.; Vanhoof, K. & Wets, G. (2001). Use of bayesian belief networks to help understand online audience, Proceedings of ECML/PKDD, 09/2001, Freiburg, Germany.

Kayaalp, M. & Cooper, G.F. (2002). A bayesian network scoring metric that is based on globally uniform parameter priors, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 251-258, Morgan Kaufmann, San Francisco (CA)

Krause, P.J. (1999). Learning probabilistic networks. *The Knowledge Engineering Review*, Vol. 13, No. 4, 321–351, 0269-8889

Kreinovich, V., Quintana, C. & Fuentes, O. (1993). Genetic algorithms: What fitness scaling is optimal ? *Cybernetics and Systems*, Vol. 24, No. 1, 9–26, 0196-9722

Lacey, G. & MacNamara, S. (2000). Context-aware shared control of a robot mobility aid for the elderly blind. *International Journal of Robotic Research*, Vol. 19, No. 11, 1054-1065, 0278-3649

Larranãga, P.; Poza, M.; Yurramendi, Y.; Murga, R. & Kuijpers. C. (1996). Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions PAMI*, Vol. 18, No. 9, 912–926, 0162-8828

Lauritzen, S.L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, Vol. 19, No. 2, 191 201, 0167-9473

Lauritzen, S.L. & Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, Vol. 17, No. 1, 31-57, 0090-5364

Lerner, U. ; Segal, E. & Koller, D. (2001). Exact inference in networks with discrete children of continuous parents, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 319-32, Morgan Kaufmann, San Francisco (CA)

Madigan, D. & York, J. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, Vol. 63, No. 2, 215–232, 03067734

Mahfoud, S.W. (1995). Niching methods for genetic algorithms. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA. IlliGAL Report 95001

Mühlenbein, H. & Paab, G. (1996). From recombination of genes to the estimation of distributions, Proceedings of Parallel Solving from Nature, pp. 178–187.

Murphy, K. (2001). The Bayes net toolbox for matlab. *Computing Science and Statistics*, Vol. 33, 331-350

Muruzábal, J. & Cotta, C. (2007). A study on the evolution of bayesian network graph structures. *Studies in Fuzziness and Soft Computing*, Vol. 213, 193-214, 1434-9922.

Muruzábal, J. & Cotta, C. (2004). A primer on the evolution of equivalence classes of Bayesian network structures, *Proceedings of Parallel Problem Solving from Nature*, pp. 612–621, Birmingham, 09/2004

Nielsen, J.D.; Kocka, T. & Peña, J.M. (2003). On local optima in learning bayesian networks, *Proceedings of Uncertainty in Artificial Intelligence*, pp. 435–442, Acapulco, Morgan Kaufmann, San Francisco (CA)

Pearl, J. & Verma, T.S. (1991). A theory of inferred causation, *In: Principles of Knowledge Representation and Reasoning*, Allen J.F., Fikes R.& Sandewall, E. (Eds.), pp. 441-452, Morgan Kaufmann, San Francisco (CA)

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 0934613737, San Francisco (CA)

Rissanen, J. (1978). Modelling by shortest data description. *Automatica*, Vol. 14, 465–471, 0005-1098

Robinson, R. (1976). Counting unlabeled acyclic digraphs, *Proceedings of Combinatorial Mathematics*, pp. 28–43

Romero, T. ; Larrañaga, P. & Sierra, B. (2004). Learning bayesian networks in the space of orderings with estimation of distribution algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 18, No. 4, 607–625, 0218-0014

Sahami, M.; Dumais, S.; Heckerman, D. & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail, *Proceedings of the AAAI-98 Workshop on Text Categorization*, pp.55-62, Madison (WI), 07/1998, AAAI Press.

Schwartz, G. (1978). Estimating the dimensions of a model. *The Annals of Statistics*, Vol. 6, No. 2, 461–464, 0090-5364

Spirtes, P.; Glymour, C. & Scheines, R. (2001). Causation, Prediction and Search, The MIT Press, 0262194406,

Suzuki, J. (1996). Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the B&B technique, *Proceedings of International Conference on Machine Learning*, pp. 462-470

Thierens, D. (2002). Adaptive mutation rate control schemes in genetic algorithms, Technical Report UU-CS-2002-056, Utrecht University

Van Dijk, S. & Thierens, D. (2004). On the use of a non-redundant encoding for learning bayesian networks from data with a GA, *Proceedings of Parallel Problem Solving from Nature*, pp. 141–150, Birmingham, 09/2004

Van Dijk, S., Thierens, D., & Van Der Gaag, L.C. (2003a). Building a GA from design principles for learning bayesian networks, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 886–897.

Van Dijk, S., Van Der Gaag, L.C. & Thierens, D. (2003b). A skeleton-based approach to learning bayesian networks from data, *Proceedings of Principles and Practice of Knowledge Discovery in Databases*, pp. 132–143, Cavtat-Dubrovnik, 09/2003

Vekaria, K. & Clack, C. (1998). Selective crossover in genetic algorithms: An empirical study, *Proceedings of Parallel Problem Solving from Nature*, pp. 438-447, Amsterdam, 09/1998

Wong, M., Lee, S.Y. & Leung, K.S. (2002). A hybrid data-mining approach to discover bayesian networks using evolutionary programming, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 214-222.

Wong, M.L., Lam,W., & Leung, K.S. (1999). Using evolutionary programming and minimum description length principle for data mining of bayesian networks. *IEEE Transactions PAMI*, Vol. 21, No. 2, 174-178, 0162-8828

Wright, S. (1964). Stochastic processes in evolution, In *Stochastic models in medecine and biology*, Gurland, J. (Ed.), 199-241, Univ. of Wisconsin Press, Madison

Yu, J.; Smith, V.A.; Wang, P.P.; Hartemink, A.J. & Jarvis., E.D. (2002). Using bayesian network inference algorithms to recover molecular genetic regulatory networks, *Proceedings of International Conference on Systems Biology*

Advances in Evolutionary Algorithms

Edited by Xiong Zhihui

ISBN 978-953-7619-11-4
Hard cover, 284 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

With the recent trends towards massive data sets and significant computational power, combined with evolutionary algorithmic advances evolutionary computation is becoming much more relevant to practice. Aim of the book is to present recent improvements, innovative ideas and concepts in a part of a huge EA field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Thierry Brouard, Alain Delaplace and Hubert Cardot (2008). Evolutionary Methods for Learning Bayesian Network Structures, Advances in Evolutionary Algorithms, Xiong Zhihui (Ed.), ISBN: 978-953-7619-11-4, InTech, Available from: http://www.intechopen.com/books/advances_in_evolutionary_algorithms/evolutionary_methods_for_learning_bayesian_network_structures

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen