2

Design of evolutionary methods applied to the learning of Bayesian network structures

Thierry BROUARD, Alain DELAPLACE, Muhammad Muzzamil LUQMAN, Hubert CARDOT and Jean-Yves RAMEL
University François Rabelais
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 automatically determining an appropriate graphical structure from a database of cases. The first one consists in the detection of (in)dependencies between the variables (Cheng et al., 2002; Spirtes et al., 2001). 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 (GA) (De Jong, 2006) have already been used in various forms (Acid & de Campos, 2003; Larrañaga et al., 1996; Muruzábal & Cotta, 2004; Van Dijk, Thierens & Van Der Gaag, 2003; Wong et al., 1999; 2002). 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, Thierens & Van Der Gaag, 2003) 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 parameters.

To overcome these problems, we designed a general genetic algorithm based upon dedicated operators: mutation, crossover but also a mutual information-driven repair 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 behavior of the used strategies will also be given. And finally, the sixth section will present an application of these algorithms in the field of graphic symbol recognition.

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.

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 vertices, depending on the values taken by its parents in G. The probability $k = \{P(X_k|Pa(X_k))\}$, where $Pa(X_k)$ are the parents of variable X_k in G. If X_k has no parents, then $Pa(X_k) = \emptyset$.
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:

$$P(X_1, \ldots, X_n) = \prod_{k=1}^{n} P(X_k|Pa(X_k)) \tag{1}$$

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

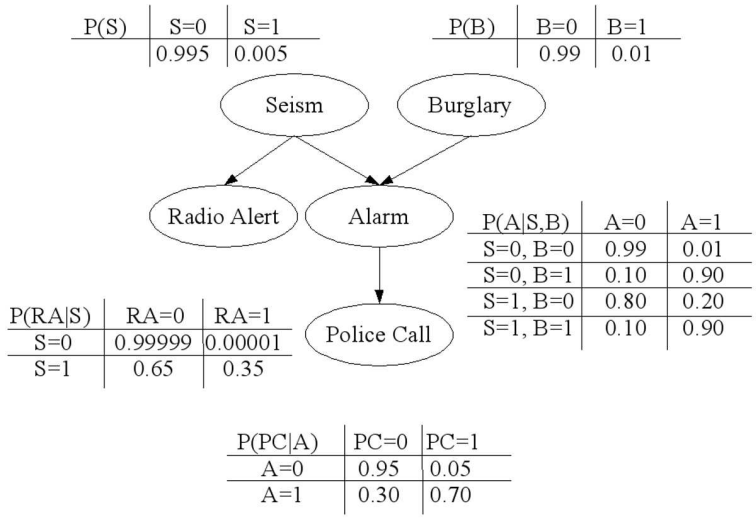


Fig. 1. Example of a Bayesian network.

software 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 resolve. 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 modeled domain): continuous variables – variables can take their values in a continuous space (Cobb & Shenoy, 2006; Lauritzen & Wermuth, 1989; Lerner et al., 2001) –, incomplete databases (Heckerman, 1995; Lauritzen, 1995). We assume in this work that the variables modeled 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 $S_{X_i,X_j} \in X/\{X_i, X_j\}$, the vertices X_i and X_j are connected by an arc in G if there is no S_{X_i,X_j} such as $(Xi \perp Xj|S_{X_i,X_j})$ 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 follow 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_{X_i,X_j} such as $(Xi \perp Xj|S_{X_i,X_j})$). 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

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

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, (denoted BD and corresponds in fact to the marginal likelihood we are trying to maximize through the determination of a structure G). In (Heckerman, 1995) 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 (Etxeberria et al., 1997; Larrañaga et al., 1996). 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 & Thierens, 2004; Van Dijk, Thierens & Van Der Gaag, 2003; Van Dijk, Van Der Gaag & Thierens, 2003) a similar method was proposed. The chromosome contains all the arcs of the network, and three alleles are defined: none, $X \rightarrow Y$ and $X \rightarrow Y$. 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; 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.

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 Algorithm. 1) 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.

Algorithm 1 Holland's canonical genetic algorithm (Holland, 1992)

/ 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 */*
repeat
  Select P_t 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 */*
until a definite criterion is met

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

1. 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.
2. 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:

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

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

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

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

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

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 Setting up the population

We choose to initialize the population of structures by the various trees (depending on the chosen root vertex) 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:

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

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):

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

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. 2. 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.

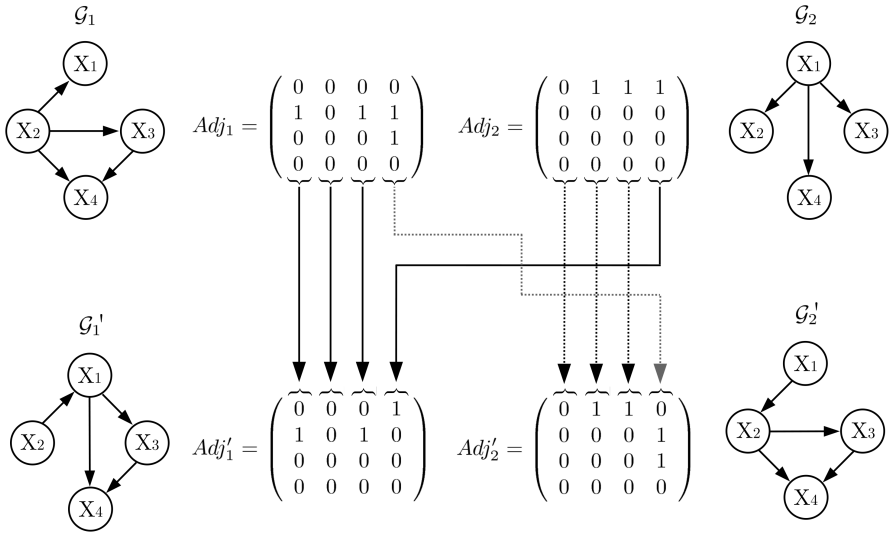


Fig. 2. The crossover operator and the transformation it performs over two DAGs

3.7 Mutation operator

Each node of one individual has a P_{mute} probability of losing or gaining one parent or to see one of its incoming arcs reverted (ie. 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.

4. Strategies

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.

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.

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 the following adaptive policy.
We propose to conduct the search over the space of solutions by taking into account information on the quality of later search. 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 op_{mute}. The mutation decays according to the choice of i, j and op_{mute}. We can simplify the density of probability by conditioning 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

$P(i|op_{mute}, j)$. 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 Algorithm 2.

Algorithm 2 The mutation operator scheme

```
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
```

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

$$\begin{cases} \sum_{op_{mute}} \delta_{op_{mute}}^{(i,j)} P(i|op_{mute}, j) = 1 \\ \delta_{op_{mute}}^{(i,j)} = \begin{cases} 1 \text{ if } op_{mute}(i,j) \text{ is allowed} \\ 0 \text{ else} \end{cases} \end{cases} \tag{7}$$

The diversity of the individuals lay down to compute $P(i|op_{mute}, j)$ 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 $1 \leq i, j \leq n$ and $i \neq j$ to control $P(i|op_{mute}, j)$. So we define:

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

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

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

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:

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

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.

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

The regular update $\zeta(i, j, op_{mute}(i, j))$ leads to standardize the $P(i|op_{mute}, j)$ 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 approach 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 evolutionary 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 neighborhood 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 neighborhood 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 an 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: too quickly consider an equivalence class as a local optimum slows exploring the search space by the genetic algorithm, which focuses on many local 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 of 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 finally 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 transfer $R_{mig} \times I_{size}$ 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, noted 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 are 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.
GS: This algorithm is initialized with a tree returned by the MWST method, where the root vertex 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)

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 GA-AM only
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.

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 hand. The latter is defined from 4 terms: (D) the total number of different arcs between two graphs G_1 and G_2, ($\oplus$) the number of arcs existing in G_1 but not in G_2, ($\ominus$) 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.

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.

5.5 Results for the ALARM network

This network contains more vertices than the INSURANCE one, but less low-valued arcs. The evaluation is averaged over 30 databases. 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.

5.6 Behavior of the GAs

Now look at some measures in order to evaluate the behavior 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 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 4. 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

	Insurance			
	250	**500**	**1000**	**2000**
GA	-32135 ± 290	-31200 ± 333	-29584 ± 359	-28841 ± 89†
GA-SN	-31917 ± 286	-31099 ± 282	-29766 ± 492	**-28681**± 156
GA-AM	**-31826**± 270	-31076 ± 151	-29635 ± 261	-28688 ± 165
GA-HN	-31958 ± 246	**-31075**± 255	**-29428**± 290	-28715 ± 164
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†
Original	-28353			
$\mathcal{G}_0$	-45614			

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

	Insurance							
	250				**500**			
	D	$\oplus$	*Inv*	$\ominus$	**D**	$\oplus$	*Inv*	$\ominus$
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-AM	37,5	4,3	6,6	26,6	33,9	3,2	7,7	23
GA-HN	38,1	3,5	7,5	27,1	**33,3**	3	7,3	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	$\oplus$	*Inv*	$\ominus$	**D**	$\oplus$	*Inv*	$\ominus$
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-AM	31,4	4	8	19,4	27	4,3	8,4	14,3
GA-HN	**29,3**	3,6	6,5	19,2	26,6	3,6	8,6	14,4
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.

makes it competitive in terms of computing time if we could detect this event.

	Insurance Net.			
	250	**500**	**1000**	**2000**
GA	364 ± 319	454 ± 295	425 ± 249	555 ± 278
GA-SN	704 ± 295	605 ± 321	694 ± 258	723 ± 234
GA-AM	398 ± 326	414 ± 277	526 ± 320	501 ± 281
GA-HN	82 ± 59	106 ± 77	166 ± 84	116 ± 27

Table 4. Mean of the necessary number of iterations to find the best structure (INSURANCE).

The averaged computing time of each algorithm is given Table 5 (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.

	ALARM Net.			
	250	**500**	**1000**	**2000**
GA	3593 ± 47	3659 ± 41	3871 ± 53	4088 ± 180
GA-SN	3843 ± 58	3877 ± 44	4051 ± 59	4332 ± 78
GA-AM	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	$4055 \pm 3,4$	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 5. Averaged computing times (in seconds) and standard deviations (ALARM).

6. Application

Graphics recognition deals with graphic entities in document images and is a subfield of document image analysis. These graphic entities could correspond to symbols, mathematical formulas, musical scores, silhouettes, logos etc., depending on the application domain. Documents from electronics, engineering, music, architecture and various other fields use domain-dependent graphic notations which are based on particular alphabets of symbols. These industries have a rich heritage of hand-drawn documents and because of high demands of application domains, overtime symbol recognition is becoming core goal of automatic image analysis and understanding systems. The method proposed in (Luqman et al., 2009) is a hybrid of structural and statistical pattern recognition approaches where the representational power of structural approaches is exploited and the computational efficiency of statistical classifiers is employed.

In our knowledge there are only a few methods which use Bayesian networks for graphic symbol recognition. Recently Barrat et al. (Barrat et al., 2007) have used the naive Bayes classifier in a *pure* statistical manner for graphic symbol recognition. Their system uses three

shape descriptors: Generic Fourier Descriptor, Zernike descriptor & R-Signature 1D, and applies dimensionality reduction for extracting the most relevant and discriminating features to formulate a feature vector. This reduces the length of their feature vector and eventually the number of variables (nodes) in Bayesian network. The naive Bayes classifier is a powerful Bayesian classifier but it assumes a strong independence relationship among attributes given the class variable. We believe that the power of Bayesian networks is not fully explored; as instead of using predefined dependency relationships, if we find dependencies between all variable pairs from underlying data we can obtain a more powerful Bayesian network classifier. This will also help to ignore irrelevant variables and exploit the variables that are interesting for discriminating symbols in underlying symbol set.

Our method is an original adaptation of Bayesian network learning for the problem of graphic symbol recognition. For symbol representation, we use a structural signature. The signature is computed from the attributed relational graph (ARG) of symbol and is composed of geometric & topologic characteristics of the structure of symbol. We use (overlapping) fuzzy intervals for computing noise sensitive features in signature. This increases the ability of our signature to resist against irregularities (Mitra & Pal, 2005) that may be introduced in the shape of symbol by deformations & degradations. For symbol recognition, we employ a Bayesian network. This network is learned from underlying training data by using the GA-HN algorithm. A query symbol is classified by using Bayesian probabilistic inference (on encoded joint probability distribution). We have selected the features in signature very carefully to best suit them to linear graphic symbols and to restrict their number to minimum; as Bayesian network algorithms are known to perform better for a smaller number of nodes. Our structural signature makes the proposed system robust & independent of application domains and it could be used for all types of 2D linear graphic symbols.

After representing the symbols in learning set by ARG and describing them by structural signatures, we proceed to learning of a Bayesian network. The signatures are first discretized. We discretize each feature variable (of signature) separately and independently of others. The class labels are chosen intelligently in order to avoid the need of any discretization for them. The discretization of *number of nodes* and *number of arcs* achieves a comparison of similarity of symbols (instead of strict comparison of exact feature values). This discretization step also ensures that the features in signature of query symbol will look for symbols whose number of nodes and arcs lie in same intervals as that of the query symbol.

The Bayesian network is learned in two steps. First we learn the structure of the network. Despite the training algorithms are evolutionary one, they have provided stable results (for a given dataset multiple invocations always returned identical network structures). Each feature in signature becomes a node of network. The goal of structure learning stage is to find the best network structure from underlying data which contains all possible dependency relationships between all variable pairs. The structure of the learned network depicts the dependency relationships between different features in signature. Fig.3 shows one of the learned structures from our experiments. The second step is learning of parameters of network; which are conditional probability distributions $Pr(node_i|parents_i)$ associated to nodes of the network and which quantify the dependency relationships between nodes. The network parameters are obtained by maximum likelihood estimation (MLE); which is a robust parameter estimation technique and assigns the most likely parameter values to best

describe a given distribution of data. We avoid null probabilities by using Dirichlet priors with MLE. The learned Bayesian network encodes joint probability distribution of the symbol signatures.

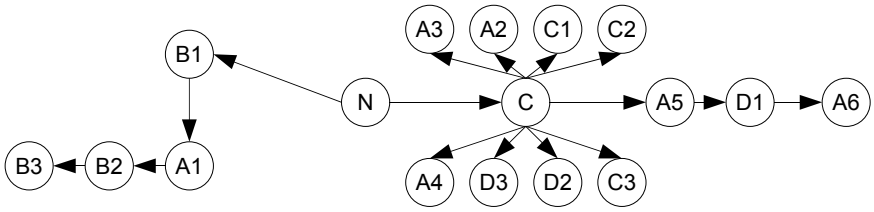


Fig. 3. Example of a Bayesian network : C = class, N = number of nodes, A1 = number of connections, A2 = number of L-junctions, A3 = number of T-junctions, A4 = number of intersections, A5 = number of parallel connections, A6 = number of successive connections, B1 (resp. B2 and B3) = number of nodes with low (resp. medium and high) density of connections, C1 (resp. C2 and C3) = number of small-length (resp. medium-length and full-length) primitives, D1 = number of small-angle (resp. medium-angle and full-angle) connections.

The conditional independence property of Bayesian networks helps us to ignore irrelevant features in structural signature for an underlying symbol set. This property states that a node is conditionally independent of its non-descendants given its immediate parents (Charniak, 1991). Conditional independence of a node in Bayesian network is fully exploited during probabilistic inference and thus helps us to ignore irrelevant features for an underlying symbol set while computing posterior probabilities for different symbol classes.

For recognizing a query symbol we use Bayesian probabilistic inference on the encoded joint probability distribution. This is achieved by using junction tree inference engine which is the most popular exact inference engine for Bayesian probabilistic inference. The inference engine propagates the evidence (signature of query symbol) in network and computes posterior probability for each symbol class. Equation 12 gives Bayes rule for our system. It states that posterior probability or probability of a symbol class c_i given a query signature *evidence* e is computed from likelihood (probability of e given c_i), prior probability of c_i and marginal likelihood (prior probability of e). The marginal likelihood $Pr(e)$ is to normalize the posterior probability; it ensures that the probabilities fall between 0 and 1.

$$Pr(c_i|e) = \frac{Pr(e, c_i)}{Pr(e)} = \frac{Pr(e|c_i) \times Pr(c_i)}{Pr(e)} \tag{12}$$

where,

$$\begin{cases} e = f_1, f_2, f_3, ..., f_{16} \\ Pr(e) = Pr(e, c_i) = \sum_{i=1}^{k} Pr(e|c_i) \times Pr(c_i) \end{cases} \tag{13}$$

The posterior probabilities are computed for all k symbol classes and the query symbol is then assigned to class which maximizes the posterior probability i.e. which has highest posterior probability for the given query symbol.

6.1 Symbols with vectorial and binary noise

The organization of four international symbol recognition contests over last decade (Aksoy et al., 2000; Dosch & Valveny, 2005; Valveny & Dosch, 2003; Valveny et al., 2007), has provided our community an important test bed for evaluation of methods over a standard dataset. These contests were organized to evaluate and test the symbol recognition methods for their scalability and robustness against binary degradation and vectorial deformations. The contests were run on pre-segmented linear symbols from architectural and electronic drawings, as these symbols are representative of a wide range of shapes (Valveny & Dosch, 2003). GREC2005 (Dosch & Valveny, 2005) & GREC2007 (Valveny et al., 2007) databases are composed of the same set of models, whereas GREC2003 (Valveny & Dosch, 2003) database is a subset of GREC2005.

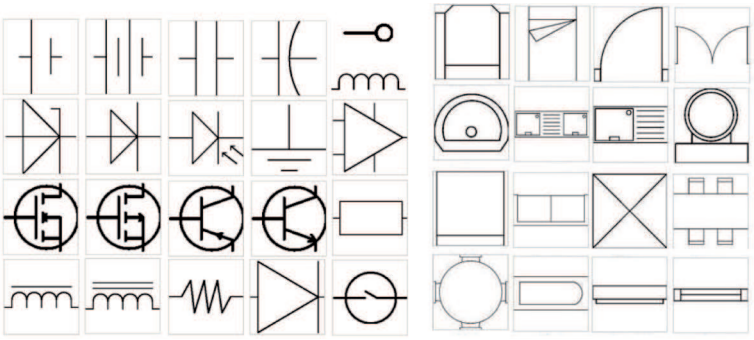


Fig. 4. Model symbols from electronic drawings and from floor plans.

We experimented with synthetically generated 2D symbols of models collected from database of GREC2005. In order to get a true picture of the performance of our proposed method on this database, we have experimented with 20, 50, 75, 100, 125 & 150 symbol classes. We generated our own learning & test sets (based on deformations & degradations of GREC2005) for our experiments. For each class the perfect symbol (the model) along with its 36 rotated and 12 scaled examples was used for learning; as the features have already been shown invariant to scaling & rotation and because of the fact that generally Bayesian network learning algorithms perform better on datasets with large number of examples. The system has been tested for its scalability on clean symbols (rotated & scaled), various levels of vectorial deformations and for binary degradations of GREC symbol recognition contest. Each test dataset was composed of 10 query symbols for each class.

Number of classes (models)	20	50	75	100	125	150
Clean symbols (rotated & scaled)	100%	100%	100%	100%	100%	99%
Hand-drawn deform. Level-1	99%	96%	93%	92%	90%	89%
Hand-drawn deform. Level-2	98%	95%	92%	90%	89%	87%
Hand-drawn deform. Level-3	95%	77%	73%	70%	69%	67%
Binary degrade	98%	96%	93%	92%	89%	89%

Table 6. Results of symbol recognition experiments.

Table 6 summarizes the experimental results. A 100% recognition rate for clean symbols illustrates the invariance of our method to rotation & scaling. Our method outperforms all GREC participants (available results from GREC2003 and GREC2005 competitions) in scalability tests and is comparable to contest participants for low levels of deformation & degradations. The recognition rates decrease with level of deformation and drop drastically for high binary degradations. This is an expected behavior and is a result of the irregularities produced in symbol signature; which is a direct outcome of the noise sensitivity of vectorization step, as also pointed out by (Llados et al., 2002). We used only clean symbols for learning and (thus) the recognition rates truely illustrate the robustness of our system against vectorial and binary noise.

6.2 Symbols with contextual noise

A second set of experimentation was performed on a synthetically generated corpus, of symbols cropped from complete documents (Delalandre et al., 2007). These experiments focused on evaluating the robustness of the proposed system against context noise i.e. the structural noise introduced in symbols when they are cropped from documents. We believe that this type of noise gets very important when we are dealing with symbols in context in complete documents and to the best of our knowledge; no results have yet been published for this type of noise. We have performed these experiments on two subsets of symbols: consisting of 16 models from floor plans and 21 models from electronic diagrams. The models are derived from GREC2005 database and are given in Fig.4. For each class the perfect symbol (model), along with its 36 rotated and 12 scaled examples was used for learning. The examples of models, for learning, were generated using ImageMagick and the test sets were generated synthetically (Delalandre et al., 2007) with different levels of context-noise in order to simulate the cropping of symbols from documents. Test symbols were randomly rotated & scaled and multiple query symbols were included for each class. The test datasets are available at (Delalandre, 2009).

Dataset	Noise	1-TOP	3-TOP
Floor plans	Level 1	84%	95%
Floor plans	Level 2	79%	90%
Floor plans	Level 3	76%	87%
Electronic diagrams	Level 1	69%	89%
Electronic diagrams	Level 2	66%	88%
Electronic diagrams	Level 3	61%	85%

Table 7. Results of symbol recognition experiments for context noise.*1*-TOP stands for the right class in given in first position and *3*-TOP stands for the right class in belonging to the first 3 answers.

Table 7 summarizes the results of experiments for context noise. We have not used any sophisticated de-noising or pretreatment and our method derives its ability to resist against context noise, directly from underlying vectorization technique, the fuzzy approach used for computing structural signature and the capabilities of Bayesian networks to cope with uncertainties. The models for electronic diagrams contain symbols consisting of complex arrangement of lines & arcs, which affects the features in structural signature as the employed vectorization

technique is not able to cope with arcs & circles; as is depicted by the recognition rates for these symbols. But keeping in view the fact that we have used only clean symbols for learning and noisy symbols for testing, we believe that the results show the ability of our signature to exploit the sufficient structural details of symbols and it could be used to discriminate and recognize symbols with context noise.

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 behavior 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, *Journ. of Art. Int. Res.* **18**: 445–490.

Akaike, H. (1970). Statistical predictor identification, *Ann. Inst. Stat. Math.* **22**(1): 203–217.

Aksoy, S., Ye, M., Schauf, M., Song, M., Wang, Y., Haralick, R., Parker, J., Pivovarov, J., Royko, D., Sun, C. & Farnebačk, G. (2000). Algorithm performance contest, *Proc. of ICPR*, pp. 4870–4876.

Allanach, J., Tu, H., Singh, S., Pattipati, K. & Willett, P. (2004). Detecting, tracking and counteracting terrorist networks via hidden markov models, *Proc. of IEEE Aero. Conf.*

Bäck, T. (1993). Optimal mutation rates in genetic search, *Proc. of Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo (CA), pp. 2–8.

Barrat, S., Tabbone, S. & Nourrissier, P. (2007). A bayesian classifier for symbol recognition, *Proc. of GREC*.

Beasley, D., Bull, D. R. & Martin, R. R. (1993). A sequential niche technique for multimodal function optimization, *Evolutionary Computation* **1**(2): 101–125.

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, *Proc. of Eur. Conf. Art. Int. in Med.*, Springer Verlag, Berlin, pp. 247–256.

Binder, J., Koller, D., Russell, S. J. & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables, *Machine Learning* **29**(2-3): 213–244.

Blanco, R., Inza, I. & Larrañaga, P. (2003). Learning bayesian networks in the space of structures by estimation of distribution algorithms., *Int. Jour. of Int. Syst.* **18**(2): 205–220.

Bouckaert, R. (1994). Properties of bayesian belief network learning algorithms, *Proc. of Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, pp. 102–110.

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

Charniak, E. (1991). Bayesian networks without tears, *AI Magazine* **12**(4): 50–63.

Cheng, J., Bell, D. A. & Liu, W. (2002). Learning belief networks from data: An information theory based approach, *Artificial Intelligence* **1-2**: 43–90.

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

Chickering, D. M. (2002b). Learning equivalence classes of bayesian-network structures, *J. of Mach. Learn. Res.* **2**: 445–498.

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

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

Chow, C. & Liu, C. (1968). Approximating discrete probability distributions with dependence trees, *IEEE Trans. on Information Theory* **14(3)**(3): 462–467.

Cobb, B. R. & Shenoy, P. P. (2006). Inference in hybrid bayesian networks with mixtures of truncated exponentials, *International Jounal of Approximate Reasoning* **41**(3): 257–286.

Cooper, G. & Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data, *Machine Learning* **9**: 309–347.

Cotta, C. & Muruzábal, J. (2002). Towards a more efficient evolutionary induction of bayesian networks., *Proc. of PPSN VII, Granada, Spain, September 7-11*, pp. 730–739.

Davis, G. (2003). Bayesian reconstruction of traffic accidents, *Law, Prob. and Risk* **2**(2): 69–89.

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

Delalandre, M. (2009). http://mathieu.delalandre.free.fr/projects/sesyd/queries.html.

Delalandre, M., Pridmore, T., Valveny, E., Locteau, H. & Trupin, E. (2007). Building synthetic graphical documents for performance evaluation, *in* W. Liu, J. Llados & J. Ogier (eds), *Lecture Notes in Computer Science*, Vol. 5046, Springer, pp. 288–298.

Dosch, P. & Valveny, E. (2005). Report on the second symbol recognition contest, *Proc. of GREC*.

Eiben, A. E., Hinterding, R. & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms, *IEEE Trans. on Evolutionary Computation* **3**(2): 124–141.

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* **18**(11-13): 1269–1273.

Ezawa, K. & Schuermann, T. (1995). Fraud/uncollectible debt detection using a bayesian network based learning system: A rare binary outcome with mixed data structures, *Proc. of Uncertainty in Artificial Intelligence*, 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* **13**(2): 39–43.

Forrest, S. (1985). Documentation for prisoners dilemma and norms programs that use the genetic algorithm. University of Michigan, Ann Arbor, MI.

Francois, O. & Leray, P. (2004). BNT structure learning package: Documentation and experiments, *Technical report*, Laboratoire PSI.
URL: *http://bnt.insa-rouen.fr/programmes/BNT_StructureLearning_v1.3.pdf*

Glickman, M. & Sycara, K. (2000). Reasons for premature convergence of self-adapting mutation rates, *Proc. of Evolutionary Computation*, Vol. 1, pp. 62–69.

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

Henrion, M. (1988). Propagation of uncertainty by probabilistic logic sampling in bayes networks, *Proc. of Uncertainty in Artificial Intelligence*, Vol. 2, Morgan Kaufmann, San Francisco (CA), pp. 149–164.

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

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, *Proc. of Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco (CA).

Hurvich, C. M. & Tsai, C.-L. (1989). Regression and time series model selection in small samples, *Biometrika* **76**(2): 297–307.

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

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

Krause, P. J. (1999). Learning probabilistic networks, *Know. Eng. Rev. Arc.* **13**(4): 321–351.

Kreinovich, V., Quintana, C. & Fuentes, O. (1993). Genetic algorithms: What fitness scaling is optimal?, *Cybernetics and Systems* **24**(1): 9–26.

Lacey, G. & MacNamara, S. (2000). Context-aware shared control of a robot mobility aid for the elderly blind, *Int. Journal of Robotic Research* **19**(11): 1054–1065.

Larrañaga, 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 Trans. on PAMI* **18**(9): 912–926.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data., *Computational Statistics & Data Analysis* **19**(2): 191–201.

Lauritzen, S. L. & Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative, *Annals of Statistics* **17**(1): 31–57.

Lerner, U., Segal, E. & Koller, D. (2001). Exact inference in networks with discrete children of continuous parents, *Proc. of UAI*, Morgan Kaufmann, San Francisco, CA, pp. 319–332.

Llados, J., Valveny, E., Sańchez, G. & Marti, E. (2002). Symbol recognition: Current advances and perspectives, *Lecture Notes in Computer Science*, Vol. 2390, Springer, pp. 104–128.

Luqman, M. M., Brouard, T. & Ramel, J.-Y. (2009). Graphic symbol recognition using graph based signature and bayesian network classifier, *International Conference on Document Analysis and Recognition*, IEEE Comp. Soc., Los Alamitos, CA, USA, pp. 1325–1329.

Madigan, D. & York, J. (1995). Bayesian graphical models for discrete data, *Int. Stat. Rev.* **63**(2): 215–232.

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

Mitra, S. & Pal, S. (2005). Fuzzy sets in pattern recognition and machine intelligence, *Fuzzy Sets and Systems* **156**(3): 381–386.

Mühlenbein, H. & PaaB, G. (1996). From recombination of genes to the estimation of distributions, *Proc. of PPSN*, Vol. 1411, pp. 178–187.

Murphy, K. (2001). The bayes net toolbox for matlab, *Comp. Sci. and Stat.* **33**: 331–350.

Muruzábal, J. & Cotta, C. (2004). A primer on the evolution of equivalence classes of bayesian-network structures, *Proc. of PPSN*, Birmingham, UK, pp. 612–621.

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

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

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 1st edn, Morgan Kaufmann, San Francisco (CA).

Pearl, J. & Verma, T. S. (1991). A theory of inferred causation, *in* J. F. Allen, R. Fikes & E. Sandewall (eds), *Proc. of Princ. of Know. Repr. and Reas.*, Morgan Kaufmann, San Mateo, California, pp. 441–452.

Rissanen, J. (1978). Modelling by shortest data description., *Automatica* **14**: 465–471.

Robinson, R. (1976). Counting unlabeled acyclic digraphs, *Proc. of Combinatorial Mathematics V*, Royal Melbourne Institute of Technology, Am. Math. Soc., Australia, pp. 28–43.

Romero, T., Larrañaga, P. & Sierra, B. (2004). Learning bayesian networks in the space of orderings with estimation of distribution algorithms, *Int. Jour. of Pat. Rec. and Art. Int.* **18**(4): 607–625.

Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail., *Proc. of the AAAI Work. on Text Categorization*, Madison, WI, pp. 55–62.

Schwartz, G. (1978). Estimating the dimensions of a model, *The Ann. of Stat.* **6**(2): 461–464.

Spirtes, P., Glymour, C. & Scheines, R. (2001). *Causation, Prediction and Search*, 2nd edn, MIT Press.

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

Thierens, D. (2002). Adaptive mutation rate control schemes in genetic algorithms, *Technical Report UU-CS-2002-056*, Inst. of Information and Computing Sciences, Utrecht Univ.

Valveny, E. & Dosch, P. (2003). Symbol recognition contest: A synthesis, *Proc. of GREC*, pp. 368–386.

Valveny, E., Dosch, P., Fornes, A. & Escalera, S. (2007). Report on the third contest on symbol recognition, *Proc. of GREC*, pp. 321–328.

Van Dijk, S. & Thierens, D. (2004). On the use of a non-redundant encoding for learning bayesian networks from data with a GA, *Proc. of PPSN*, pp. 141–150.

Van Dijk, S., Thierens, D. & Van Der Gaag, L. (2003). Building a ga from design principles for learning bayesian networks, *Proc. of Genetic and Evol. Comp. Conf.*, pp. 886–897.

Van Dijk, S., Van Der Gaag, L. C. & Thierens, D. (2003). A skeleton-based approach to learning bayesian networks from data., *Proc. of Princ. and Prac. of Knowl. Disc. in Databases*, Cavtat-Dubrovnik, Croatia, pp. 132–143.

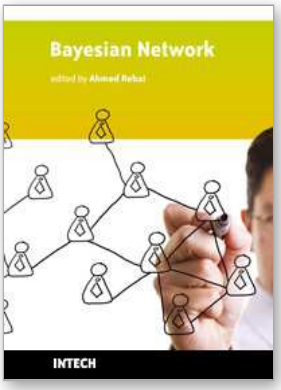
Vekaria, K. & Clack, C. (1998). Selective crossover in genetic algorithms: An empirical study, *Proc. of PPSN, Amsterdam, The Netherlands, September 27-30, 1998*, pp. 438–447.

Wong, M., Lam, W. & Leung, K. S. (1999). Using evolutionary programming and minimum description length principle for data mining of bayesian networks, *IEEE Trans. on PAMI* **21**(2): 174–178.

Wong, M., Lee, S. Y. & Leung, K. S. (2002). A hybrid data mining approach to discover bayesian networks using evolutionary programming., *Proc. of the Genetic and Evol. Comp. COnf.*, pp. 214–222.

Wright, S. (1964). Stochastic processes in evolution, *in* J. Gurland (ed.), *Stochastic models in medecine and biology*, University of Wisconsin Press, Madison, WI, pp. 199–241.

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, *Prof. of Int. Conf. on Systems Biology (ICSB02)*.



Bayesian Network

Edited by Ahmed Rebai

ISBN 978-953-307-124-4
Hard cover, 432 pages
Publisher Sciyo
Published online 18, August, 2010
Published in print edition August, 2010

Bayesian networks are a very general and powerful tool that can be used for a large number of problems involving uncertainty: reasoning, learning, planning and perception. They provide a language that supports efficient algorithms for the automatic construction of expert systems in several different contexts. The range of applications of Bayesian networks currently extends over almost all fields including engineering, biology and medicine, information and communication technologies and finance. This book is a collection of original contributions to the methodology and applications of Bayesian networks. It contains recent developments in the field and illustrates, on a sample of applications, the power of Bayesian networks in dealing the modeling of complex systems. Readers that are not familiar with this tool, but have some technical background, will find in this book all necessary theoretical and practical information on how to use and implement Bayesian networks in their own work. There is no doubt that this book constitutes a valuable resource for engineers, researchers, students and all those who are interested in discovering and experiencing the potential of this major tool of the century.

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