
ELSEVIER International Journal of Approximate Reasoning 31 (2002) 291–311

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

Ant colony optimization for learning Bayesian networks

Luis M. de Campos [a,*], Juan M. Fernández-Luna [b], José A. Gámez [c], José M. Puerta [c]

[a] *Departamento de Ciencias de la Computación e I.A., E.T.S. de Ingenieria Informatica, Universidad de Granada, Avenida Andalucia 38, 18071 Granada, Spain*

[b] *Departamento de Informática, Universidad de Jaén, 23071 Jaén, Spain*

[c] *Departamento de Informática, Universidad de Castilla-La Mancha, 02071 Albacete, Spain*

Received 1 January 2002; accepted 1 July 2002

Abstract

One important approach to learning Bayesian networks (BNs) from data uses a scoring metric to evaluate the fitness of any given candidate network for the data base, and applies a search procedure to explore the set of candidate networks. The most usual search methods are greedy hill climbing, either deterministic or stochastic, although other techniques have also been used. In this paper we propose a new algorithm for learning BNs based on a recently introduced metaheuristic, which has been successfully applied to solve a variety of combinatorial optimization problems: ant colony optimization (ACO). We describe all the elements necessary to tackle our learning problem using this metaheuristic, and experimentally compare the performance of our ACO-based algorithm with other algorithms used in the literature. The experimental work is carried out using three different domains: ALARM, INSURANCE and BOBLO.

Keywords: Bayesian networks; Learning; Ant colony optimization

* Corresponding author. Tel.: +34-958-244019; fax: +34-958-243317.

E-mail addresses: lci@decsai.ugr.es (L.M. de Campos), jmfluna@ujaen.es (J.M. Fernández-Luna), jgamez@info-ab.uclm.es (J.A. Gámez), jpuerta@info-ab.uclm.es (J.M. Puerta).

0888-613X/02/$ - see front matter
PII: S0888-613X(02)00091-9

1. Introduction

Bayesian networks (BNs), also known as probabilistic belief networks or causal networks, are knowledge representation tools capable of efficiently manage the dependence/independence relationships among the random variables that compose the problem domain we wish to model. This representation has two components: (a) a graphical structure, or more precisely a directed acyclic graph (dag), and (b) a set of parameters, which together specify a joint probability distribution over the random variables [28,37]. In BNs, the graphical structure represents dependence and independence relationships. The parameters are a collection of conditional probability measures, which shape the relationships.

Once the BN has been specified, it constitutes an efficient device for the performance of inference tasks. However, there still remains the previous problem of building such a network. It is an important task, therefore, to develop automatic methods capable of learning the network directly from data, as an alternative or a complement to the method of eliciting conditional (in)dependence assertions from experts.

Nowadays, the problem of learning or estimating a BN from data is receiving increasing attention within the community of researchers into uncertainty in artificial intelligence. Algorithms for learning (the structure of) BNs have been studied, basically from two points of view: methods based on conditional independence tests [15,17,38,41] and methods based on a scoring metric optimization [13,26,30]. This classification is not exhaustive and/or strict, since there are also some algorithms that use a combination of these two methods [1,2,14,40]. In this paper we focus on learning methods based on a scoring metric.

Because learning BNs is, in general, a NP-hard problem [12] and exact methods become unfeasible, we have to solve the problem with heuristic methods. Most existing scoring-based learning algorithms apply standard heuristic search techniques, such as greedy hill climbing (HC), iterated local search (ILS), simulated annealing, etc. In this paper we propose a new scoring-based learning method that uses a recently introduced metaheuristic for combinatorial optimization: ant colony optimization (ACO) [23]. The algorithms based on ACO were initially used to solve specific problems: the ant system, for example, was successfully applied to the traveling salesman problem (TSP) [22], whose search space is the set of permutations of the nodes in a graph. Applications to shortest path problems in graphs were developed in order to study the behavior of these algorithms on simple problems, but they later gave rise to an optimization metaheuristic that can be applied to combinatorial optimization problems which may be represented in the form of a graph [20].

The paper is structured as follows: we begin in Section 2 with the preliminaries, where we briefly describe the concepts and methods related to BNs and ACO which we require for our discussion in the latter part of the paper. In Section 3 we develop an algorithm for learning BNs based on ACO. In Section 4 we present the experimental evaluation of our algorithm. Finally, Section 5 contains the concluding remarks.

2. Preliminaries

In this section we briefly review some basic concepts related to BNs and how to learn them, as well as other concepts related to ACO.

2.1. Learning Bayesian networks

A BN is a directed acyclic graph $G = (\boldsymbol{V}, E)$, where the set of nodes $\boldsymbol{V} = \{x_1, x_2, \ldots, x_n\}$ represents the system variables and E, a set of arcs, represents the direct dependence relationships among the variables. A set of parameters is also stored for each variable in $\boldsymbol{V}$, which are usually conditional probability distributions. For each variable $x_i \in \boldsymbol{V}$ we have a family of conditional distributions $P(x_i|Pa(x_i))$, where $Pa(x_i)$ represents the parent set of the variable x_i. From these conditional distributions we can recover the joint distribution over $\boldsymbol{V}$:

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|Pa(x_i)) \tag{1}$$

This expression represents a decomposition of the joint distribution. The dependence/independence relationships which make this decomposition possible are graphically encoded (through the d-separation criterion [37]) by means of the presence or absence of direct connections between pairs of variables.

The problem of learning a BN can be stated as follows: given a *training set* $D = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$ of instances of $\boldsymbol{V}$, find the BN that best matches D. The common approach to this problem is to introduce a scoring function, f, that evaluates each network with respect to the training data, and then to search for the best network according to this score. Different Bayesian and non-Bayesian scoring metrics can be used [2,10,13,26,30].

A desirable and important property of a metric is its decomposability in the presence of full data, i.e., the scoring function can be decomposed in the following way:

$$f(G : D) = \sum_{i=1}^{n} f(x_i, Pa(x_i) : N_{x_i, Pa(x_i)}) \tag{2}$$

where $N_{x_i,Pa(x_i)}$ are the statistics of the variable x_i and $Pa(x_i)$ in D, i.e., the number of instances in D that match each possible instantiation of x_i and $Pa(x_i)$. The decomposition of the metric is very important for the learning task: a local search procedure that changes one arc at each move can efficiently evaluate the improvement obtained by this change, because it can reuse most of the computations made in previous stages (only the statistics corresponding to the variables whose parent sets have been modified need to be recomputed). One example is a greedy HC method that at each step performs the local change yielding the maximal gain, until it reaches a local maximum of the scoring function. The usual choices for local changes in the space of dags are arc addition, arc deletion and arc reversal [26]. As this procedure is trapped in the first local maximum it reaches, several methods for avoiding this situation have been used, such as stochastic HC (with random restart [26]), variable neighborhood search [18], genetic algorithms (GAs) [31,34], simulated annealing [11], Tabu search [8], etc.

We will now review the algorithm B and the K2 metric, because the former will be used in our ACO based learning algorithm and the latter will be the scoring function used in our experiments.

2.1.1. The algorithm B

Algorithm B [9] is a greedy construction heuristic. It starts with an empty dag (arc-less structure) and at each step it adds the arc with the maximum increase in the (decomposable) scoring metric f, but avoiding the inclusion of directed cycles in the graph. The algorithm stops when adding any valid arc does not increase the value of the metric. An outline of algorithm B is given in Fig. 1.

In this algorithm $A[i,j]$ is an adjacency matrix that stores the difference $f(x_i, Pa(x_i) \cup \{x_j\}) - f(x_i, Pa(x_i))$, i.e., the gain obtained by inserting the arc $x_j \rightarrow x_i$ in the graph. The arcs whose inclusion in the graph would generate a directed cycle are identified by assigning to $A[i,j]$ the value $-\infty$. At each step, after inserting a valid arc $x_j \rightarrow x_i$, the algorithm identifies the new forbidden arcs by searching for the ancestors and descendants of x_i. Then, as the value $f(x_i, Pa(x_i))$ has been modified, the algorithm recomputes the new values of $A[i,k]$ for any valid arc $x_k \rightarrow x_i$. The computational complexity for this update is $O(n^2)$.

2.1.2. The K2 metric

The K2 algorithm [13] is perhaps the best known of the algorithms for learning BNs, and has been the basis of much subsequent work. This algorithm uses a Bayesian scoring metric, which measures the joint probability of a BN G and a database D. The metric has adopted the name of the algorithm, so that it is referred to as the K2 metric, whose expression is:

1. *Initialization:*

(a) **for** $i = 1$ **to** n **do**: $Pa(x_i) = \emptyset$
(b) **for** $i = 1$ *and* $j = 1$ **to** n **do**:
if $(i \neq j)$ *then* $A[i,j] = f(x_i, x_j) - f(x_i, \emptyset)$

2. *Loop:*

(a) **repeat**
i. *Select two indexes* (i,j), *such that,* $(i,j) = \arg\max_{i',j'} A[i',j']$
ii. **if** $(A[i,j] > 0)$ **then** $Pa(x_i) = Pa(x_i) \cup \{x_j\}$
iii. $A[i,j] = -\infty$
iv. **for all** $x_a \in Ancestors(x_j) \cup \{x_j\}$ *and* $x_b \in Descendants(x_i) \cup \{x_i\}$ **do**: $A[a,b] = -\infty$
v. **for** $k = 1$ **to** n **do**:
if $(A[i,k] > -\infty)$ **then** $A[i,k] = f(x_i, Pa(x_i) \cup \{x_k\}) - f(x_i, Pa(x_i))$
until $\forall\, i,j\ (A[i,j] \leq 0$ *or* $A[i,j] = -\infty)$.

Fig. 1. Structure of algorithm B.

$$P(G,D) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

where r_i is the number of possible values of the variable x_i, q_i is the number of possible configurations (instantiations) for the variables in $Pa(x_i)$, N_{ijk} is the number of cases in D in which variable x_i has its kth value and $Pa(x_i)$ is instantiated to its jth value, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

Assuming an uniform prior for $P(G)$ and using $\log(P(G,D))$ instead of $P(G,D)$, we get a decomposable metric:

$$f_{\mathrm{K2}}(G:D) = \sum_{i=1}^{n} f_{\mathrm{K2}}(x_i, Pa(x_i) : N_{x_i, Pa(x_i)}) \tag{3}$$

$$f_{\mathrm{K2}}(x_i, Pa(x_i) : N_{x_i, Pa(x_i)}) = \sum_{j=1}^{q_i} \left(\log \left(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \right) + \sum_{k=1}^{r_i} \log(N_{ijk}!) \right) \tag{4}$$

2.2. Optimization based on ant colonies

Ant algorithms [20–23] are based on the cooperative behavior of real ant colonies, which are able to find the shortest path from a food source to their

nest. While walking, real ants deposit a chemical substance called *pheromone* on the ground. Ants can smell pheromone and, when choosing their way, they tend to choose, in a probabilistic way, paths marked by strong pheromone concentrations. In the absence of pheromone, ants choose randomly, but after a transitory period shortest paths will be more frequently visited and pheromone will accumulate faster on them, which in turn causes more ants to use these paths. This positive feedback effect means that all the ants will eventually use the shortest path. So, although a single ant is capable of building a solution (i.e., a path), the optimal solution comes about solely as a result of the cooperative behavior of the ant colony (which is based on a simple form of indirect communication through the pheromone, called *stigmergy*). Although the first ACO algorithm, called Ant System, was applied to solve the TSP problem, a large number of applications to other problems were proposed after the introduction of ant system. Recently, the ACO metaheuristic was proposed as a common framework for existing applications [20,21].

Each ant builds a possible solution to the problem by moving through a finite sequence of neighbor states (nodes). Moves are selected by applying a stochastic local search directed by the ant internal state, problem-specific local information and the shared information about the pheromone.

Pheromone is modeled by means of a matrix τ, where τ_{ij} contains the level of pheromone deposited in the arc from node i to node j. In the first ant systems, an ant k in node i will select the next node j to visit with probability:

$$p_k(i,j) = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in J_k(i)} [\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}} & \text{if } j \in J_k(i) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where η_{ij} represents heuristic information about the problem; $J_k(i)$ is the set of neighbor nodes of node i that have not yet been visited by the ant k; α and β are two parameters that determine the relative importance of the pheromone with respect to the heuristic information. For example, in the TSP, $\eta_{ij} = 1/d_{ij}$, d_{ij} being the distance between cities i and j.

A different transition rule (for complex models) [22] introduces another parameter, as a trade-off between *exploitation* and *exploration*. The next node j to visit is chosen as

$$j = \begin{cases} \arg\max_{u \in J_k(i)} \{[\tau_{iu}][\eta_{iu}]^{\beta}\} & \text{if } q \leqslant q_0 \\ J & \text{if } q > q_0 \end{cases} \tag{6}$$

where q is a random number uniformly distributed in $[0,1]$; q_0 is the parameter that determines the relative importance of exploitation versus exploration ($0 \leqslant q_0 < 1$); $J \in J_k(i)$ is a node randomly selected according to the probabilities in eq. (5), with $\alpha = 1$.

At each iteration of the algorithm each ant, using the previous transition rule, progressively builds a solution (path). The matrix of pheromone is updated in the following way:

- *Global* updating: Only the ant which constructed the best solution reinforces the level of pheromone in the arcs that are part of the best solution, S^+, obtained so far. This directs the search in the neighborhood of the best solution. The global updating rule is:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{7}$$

where

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{C(S^+)} & \text{if } \{ij\} \in S^+ \\ \tau_{ij} & \text{if } \{ij\} \notin S^+ \end{cases}$$

ρ ($0 < \rho \leqslant 1$) is a parameter that controls the pheromone evaporation (decay) and $C(S^+)$ is the cost associated the best solution. Note with that the expression above implies that only the arcs belonging to the current best solution are reinforced.

- *Local* updating: While building a solution, if an ant carries out the transition from node i to node j, then the pheromone level of the corresponding arc is changed in the following way:

$$\tau_{ij} \leftarrow (1-\psi)\tau_{ij} + \psi\tau_0 \tag{8}$$

where τ_0 is the initial pheromone level and $0 < \psi \leqslant 1$. Every time an ant uses an arc it looses some of its pheromone, making the arc less desirable. This rule favors the exploration of other arcs, thus avoiding premature convergence; without local updating all the ants would search in the neighborhood of the best solution found so far.

Another improvement included in the Ant colony system algorithm with respect to previous ant systems is the use of a local optimizer: some or all of the solutions obtained by the ants are locally optimized by using a local search method. This technique is particularly useful for many combinatorial optimization problems, where in practice best results are obtained when coupling ACO algorithms with local optimizers.

3. Learning Bayesian networks using ant colony optimization

In this section we develop a scoring-based learning algorithm for BNs that uses any given decomposable metric, the search method being based on ACO.

To apply the ACO metaheuristic to our problem, we have to define the following components:

- An appropriate representation of the problem, which allows the incremental construction of possible solutions, using a probabilistic transition rule to move from one state i to a neighboring state j.
- The heuristic information that will represent the problem-specific knowledge used by the search process to move from state i to state j, η_{ij}.
- The rule(s) to update the pheromone matrix τ.
- The probabilistic transition rule that uses the heuristic η and the pheromone τ.
- The local optimizer.

Now, let us define all these components for our learning problem:

- *Representation of the problem*: In our case, the representation of the problem is a graph where the states of the problem are dags with n nodes. Thus, a state G_h will be a graph with the nodes $x_i \in \boldsymbol{V}$ and exactly h arcs and no directed cycle. The ant incremental construction of the solution starts from the empty graph G_0 (arcs-less dag) and proceeds by adding an arc $x_j \rightarrow x_i$ to the current state G_h, i.e., $G_{h+1} = G_h \cup \{x_j \rightarrow x_i\}$. The final solution will be the state G_h in which the ant decides to stop the construction phase. Fig. 2 illustrates this process.
- *Heuristic information*: The selected heuristic is to include in the graph the arc producing the greatest increase in the selected decomposable metric f. Therefore, we define

$$\eta_{ij} = f(x_i, Pa(x_i) \cup \{x_j\}) - f(x_i, Pa(x_i)) \tag{9}$$

  Note that this is the way in which the algorithm B proceeds. Note also that this heuristic information is not static, i.e., it is not the same for all the ants (as happens, e.g., in the TSP, where the distance between cities is fixed). In our case the heuristic also depends on the ant's internal state, i.e., the current graph representing the partial solution of the problem, which determines the identity of the sets $Pa(x_i)$.
- *Pheromone updating rules*: The global and local updating rules considered are the same as previously described, Eqs. (7) and (8), where τ_{ij} is the level

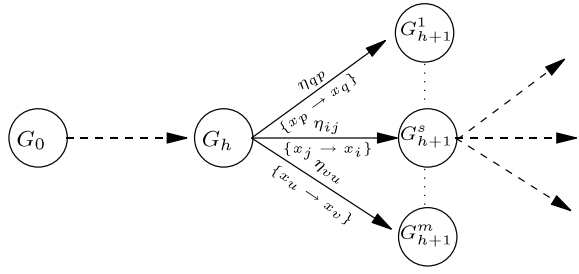

Fig. 2. Transition graph for Ant B.

of pheromone in the arc $x_j \rightarrow x_i$, G^+ is the best graph found so far and $\Delta\tau_{ij} = 1/|f(G^+ : D)|$, if $x_j \rightarrow x_i \in G^+$ and $\Delta\tau_{ij} = \tau_{ij}$ otherwise; the initial level of pheromone is $\tau_0 = 1/n|f(G_{\mathrm{K2SN}} : D)|$, where n is the number of variables and G_{K2SN} is the network obtained by the K2SN heuristic [18]. Observe that we are using the absolute value, $|\cdot|$, in the expressions for $\Delta\tau_{ij}$ and τ_0. The reason is that the values of $f(\cdot)$ are negative, because we always use the logarithmic version of the metric.

- *Probabilistic transition rule*: The next arc to be included in the current graph, G, by an ant is selected in a way similar to that used by algorithm B, but using a stochastic decision rule (instead of a deterministic rule) that also takes into account the pheromone deposited at each arc (thus obtaining expressions similar to Eqs. (6) and (5)):

$$\text{Select } x_l \rightarrow x_r \text{ such that } r,\ l = \begin{cases} \arg\max_{i,j \in F_G} \left\{ [\tau_{ij}][\eta_{ij}]^{\beta} \right\} & \text{if } q \leqslant q_0 \\ I,\ J & \text{if } q > q_0 \end{cases} \tag{10}$$

where I, J are two nodes randomly selected according to the following probabilities:

1. *Initialization:*

(a) **for** $i = 1$ **to** n **do**: $Pa(x_i) = \emptyset$
(b) **for** $i = 1$ *and* $j = 1$ **to** n **do**:
if $(i \neq j)$ *then* $\eta_{ij} = f(x_i, x_j) - f(x_i, \emptyset)$

2. *Loop:*

(a) **repeat**
i. *Select two indexes* i *and* j *by using expressions* (10) *and* (11)
ii. **if** $(\eta_{ij} > 0)$ **then** $Pa(x_i) = Pa(x_i) \cup \{x_j\}$
iii. $\eta_{ij} = -\infty$
iv. **for all** $x_a \in Ancestors(x_j) \cup \{x_j\}$ *and* $x_b \in Descendants(x_i) \cup \{x_i\}$ **do**: $\eta_{ab} = -\infty$.
v. **for** $k = 1$ **to** n **do**:
if $(\eta_{ik} > -\infty)$ **then** $\eta_{ik} = f(x_i, Pa(x_i) \cup \{x_k\}) - f(x_i, Pa(x_i))$
vi. $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$
until $\forall\ i, j\ (\eta_{ij} \leq 0$ *or* $\eta_{ij} = -\infty)$.

Fig. 3. Structure of Ant B.

$$p_k(i,j) = \begin{cases} \frac{[\tau_{ij}][\eta_{ij}]^{\beta}}{\sum_{u,v \in F_G} [\tau_{uv}]^{\alpha} [\eta_{uv}]^{\beta}} & \text{if } i,j \in F_G \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The set F_G contains all the arcs which are still candidates for insertion in G (i.e., they do not belong to G, their inclusion in G does not create a directed cycle and $\eta_{ij} > 0$).

- *Local optimizer*: In this case we use a HC algorithm with the standard operators of arc addition, arc deletion and arc reversal (HCST) [26]. This algorithm is a greedy best-improvement where, at each step, the best move according to the metric and operators used is selected. The complexity for these moves is $O(n^2)$ (n is the number of variables). We should note that if we use a decomposable metric, a large number of computations can be reused from the previous stages of the algorithm. We should also note that the transition operators chosen for HCST contain the one chosen for Ant B.

1. *Initialization:*

- **(a)** *Obtain* G_{K2SN}
- **(b)** $\tau_0 = \frac{1}{n \cdot |f(G_{K2SN}:D)|}$
- **(c)** **for all** *arc* (i,j) **do**: $\tau_{ij} = \tau_0$
- **(d)** $G^+ = G_{K2SN}$
- **(e)** *Set* t_{step} */* number of iterations for doing local search */*

2. *Loop:* */** t_{max} *iterations,* m *ants */*

- **(a)** **For** $t = 1$ **to** t_{max} **do**:
  - *i.* **for** $k = 1$ **to** m **do**:
    - *A.* $G_k = Ant - B()$ */* see Figure 3 */*
    - *B.* **If** $(t \ mod \ t_{step} = 0)$ *then* $G_k = HillClimbing(G_k)$
  - *ii.* $G_b = \arg\max_{k:1..m} f(G_k : D)$
  - *iii.* **If** $f(G_b : D) \geq f(G^+ : D)$ **then** $G^+ = G_b$
  - *iv. Perform global pheromone update, eq. (7), using* $f(G^+ : D)$

3. *Local optimization:*

- **(a)** **for** $k = 1$ **to** m **do**: $G_k^o = HillClimbing(G_k)$
- **(b)** $G_b = \arg\max_{k:1..m} f(G_k^o : D)$
- **(c)** **If** $f(G_b : D) \geq f(G^+ : D)$ **then** $G^+ = G_b$

4. *Return* G^+

Fig. 4. Description of the ACO-B learning algorithm.

Therefore, once an ant has obtained a solution, then by deleting or reversing an arc, the HCST algorithm can escape from an eventual local optima reached by the ant.

The local optimizer has been used in the very last iteration or every 10 iterations, where the result obtained by each ant is the starting point of the corresponding local search.

Fig. 3 shows the steps followed by an ant in our system to build a solution. Fig. 4 displays the overall process.

4. Experimental evaluation

4.1. Databases and algorithms

In order to test the behavior of the method proposed in the paper, three problems (domains) have been selected: ALARM [5], INSURANCE [6] and BOBLO [39]. The ALARM network has 37 nodes and 46 arcs and is used for diagnosis in a medical domain. It has been considered to be a benchmark for evaluating learning algorithms. All the experiments with ALARM have been carried out on the first 3000 cases of the ALARM database (which contains 20,000 cases, generated by probabilistic logic sampling [27]). The INSURANCE network, which contains 27 variables and 52 arcs, is a network for evaluating car insurance risks. The BOBLO network is a system which helps in the verification of the parentage of Jersey cattle through blood type identification, and contains 23 variables and 24 arcs. The experiments with INSURANCE use three databases containing 10,000 cases each and those for BOBLO use a database containing 5000 cases, all of them also generated by probabilistic logic sampling.

We have carried out an empirical comparison of the proposed learning algorithm (ACO-B) with two variants (ACO-B1, where the local optimizer is fired in the last iteration and ACO-B2, where the local optimizer is fired every 10 iterations and also in the last one) and another three algorithms using different optimization techniques: HC, ILS, which couples a HC until a local optimum is reached with a random transformation of that local optimum to be used as the starting point for the next HC phase, and finally estimation of distribution algorithms (EDAs). Below we give a brief description of EDAs. In all the cases the scoring metric used to guide the search is the K2 metric.

4.1.1. Estimation of distribution algorithms

EDAs [32] are a recent paradigm of evolutionary computation. EDAs are populational-based evolutionary algorithms, but differ from the well-known GAs [35] in the way the next population (P_{t+1}) is generated. The key point in an

evolutionary algorithm is the evolution of a population of chromosomes/individuals along time. Each chromosome in the population is a potential solution. In our case, in a n-dimensional domain, a chromosome will be a bidimensional array C of order $n \times n$, such that,

$$C[i,j] = \begin{cases} 1 & \text{if } x_i \to x_j \text{ is in the graph} \\ 0 & \text{otherwise} \end{cases}$$

Of course, we must be careful to avoid directed cycles in the chromosomes.

In EDAs there are neither crossover nor mutation. Instead of these operators, a probabilistic model is induced from some of the individuals in population P_t, and then P_{t+1} is obtained by sampling this probabilistic model. In this work we only consider the most simple version of EDAs, in which no dependence relation is considered among the variables (positions in the chromosome). Therefore, the joint probability distribution can be factorized as the product of marginal probabilities.

Two algorithms will be considered: UMDA [36] and PBIL [4]. The main difference between these two algorithms is that in UMDA the new probabilistic model replaces the old one, while in PBIL the new model is used to refine the old one by means of a parameter α.

We have used the following strategy in order to avoid the generation of directed cycles in the individuals sampled from the learned model: If $P_{i,j}(0)$ and $P_{i,j}(1)$ are the estimated probabilities of an arc $x_i \to x_j$ being absent or present, respectively, we replace them by $P_{i,j}(0) = 1.0$ and $P_{i,j}(1) = 0.0$, when the introduction of this arc induces a directed cycle in the graph. Notice that, if sampling is carried out by visiting the positions of the chromosome in a lexicographical way, i.e., $(1,1), (1,2), \ldots, (n-1,n), (n,n)$, then the positions (arcs) sampled as 1 processed earlier have a greater probability of actually having value 1 in the sampled configuration than the positions considered later. For this reason, we manage this enumeration as a circular list, and each time sampling is carried out a new starting point is randomly generated.

The whole process of obtaining P_{t+1} from P_t is as follows: (1) Select the best $populationSize/2$ individuals from P_t; (2) learn a probabilistic model by using the selected individuals as training data; (3) if UMDA replace the current model by the new one, else (PBIL) refine the current model by using the new one; (4) sample a new population P_{aux} from the probabilistic model; and (5) get P_{t+1} as the *populationSize* best individuals contained in $P_t \cup P_{\text{aux}}$.

4.2. Parameter settings

- The ACO-B algorithms have been used with the following parameters in all the cases: $\rho = \psi = 0.4$, $\beta = 2.0$, $q_0 = 0.8$, $m = 10$ ants and $t_{\max} = 100$ iterations. These parameters have not been fitted by preliminary experimentation, and are similar to the ones used for other problems [22,25].

- The HC (HCST) search uses the already mentioned operators of addition, deletion and reversal of arcs. The initialization of the search is carried out by using an empty network. This is the same local optimizer that the ACO-B algorithm uses.
- The ILS algorithm uses a previously fixed number of random local perturbation of each local optimum reached by the HCST algorithm (avoiding directed cycles). This number has been fixed at 125, [1] together with an upper bound for the number of parents of each variable in the perturbed graph equal to 8. [2] The maximum number of iterations in this case has been fixed at 15.
- For EDAs we have used the same parameters as in [7]: the initial population is generated randomly; population size is calculated as $10n$, n being the number of variables in the domain. In PBIL, $\alpha = 0.5$. In UMDA and PBIL the best half individuals contained in the population are used to induce the new probabilistic model (UMDA) or to refine the current probabilistic model (PBIL). The algorithms stop before carrying out the maximum number of generations (600) if the sum of the fitness in P_{t+1} is the same as in P_t. We have also used the HCST algorithm with the best individual found in the last iteration as the starting point.

4.3. Performance measures

To evaluate the quality of the different algorithms, we have calculated several performance measures, some measuring the quality of the results and others measuring the complexity of the algorithms:

- Measures to evaluate the quality of the learned networks:
  - The value of the **K2** metric (log version), Eq. (4).
  - The **KL** value, defined as follows:

$$KL(G:D) = \sum_{i=1, Pa(x_i)\neq\emptyset}^{n} \mathrm{Dep}(x_i, Pa(x_i)) \tag{12}$$

where $\mathrm{Dep}(\cdot,\cdot)$ is the measure of mutual information. Note that $KL(G:D)$ is a decreasing monotonic transformation of the Kullback distance [29] between the probability distribution associated with the database and the probability distribution associated with the network G [15,30]. We use this transformation because it can be calculated very

[1] A similar value has been used in other works [18,26].

[2] Without imposing such a limit, the algorithm becomes extremely slow because of the great complexity of the statistics to be computed after the random perturbations.

efficiently, whereas the computation of the Kullback distance has an exponential complexity. The interpretation of $KL(G : D)$ is: the higher this parameter the better the network.
  - Structural differences between the learned and the original network: the number of arcs added (**A**), deleted (**D**) and inverted [3] (**I**), compared with the original network.
- Measures to evaluate the complexity of the algorithms:
  - The number of the iterations, **It**, where the best individual was found. The subsequent iterations do not improve the results.
  - **TEst** represents the total number of statistics N_{ijk} evaluated during the learning process.
  - The value TEst is not necessarily equal to the number of statistics truly computed *from the data*, [4] since we can use hashing techniques to avoid the necessity of recomputing previously calculated values, thus reducing substantially this number. Therefore, we also show the number of *different* statistics used, **DEst**.
  - As the complexity of the computation of the statistics grows exponentially with the number of variables involved, we also show the average number of variables appearing in the different statistics evaluated, **NVars**. For comparative purposes, a raw estimation of the running time employed by an algorithm [5] is DEst $\times 2^{\text{NVars}}$.

In addition to the previously described measures, we have also considered, for the ACO-B and the EDAs algorithms, the value of the K2 metric attained *before* using the local optimizer, **K2noHC**.

4.4. Experimental results and analysis

The results of our experimentation are displayed in Tables 1–3. In these tables $\mu \pm \sigma$ indicates the mean and the standard deviation over the executions carried out. We have carried out 10 executions of each algorithm and for each domain considered. The value inside $(\cdot)$ is the best result found along the experimentation by using the corresponding algorithm. We should note that the parameters shown in tables for the best BN are those corresponding to the best K2 value found. Notice that HCST is a deterministic algorithm, so only one execution was carried out.

[3] Only if they give rise to non-equivalent structures [38].

[4] Note that this is usually the most costly process for scoring-based learning algorithms.

[5] For example, for the ALARM domain, a typical execution in our implementation takes about 1:56 min for HCST and about 1:05 h for ACO-B2, i.e., ACO-B2 is around 34 times slower than HCST. The approximation in this case yields a rate of 36.

Table 1
Results for ALARM

ALARM	$\mu \pm \sigma$ (best)					HCST (best)
	ILS	ACO-B1	ACO-B2	UMDA	PBIL	
K2	(−) −14,413.20 ± 6.64 (−14,403.28)	(−) −14,406.06 ± 4.54 (−14,401.29)	−14,401.83 ± 0.72 (−14,401.29)	(−) −14,431.87 ± 12.61 (−14,420.17)	(−) −14,431.22 ± 27.10 (−14,413.01)	(−) −14,425.62
KL	9.230 ± 0.006 (9.230)	9.231 ± 0.003 (9.231)	9.231 ± 0.001 (9.231)	9.230 ± 0.005 (9.230)	9.229 ± 0.006 (9.232)	9.220
A	5.80 ± 2.25 (2)	3.30 ± 1.79 (2)	2.10 ± 0.70 (2)	7.80 ± 0.87 (7)	6.90 ± 2.07 (5)	6
D	2.10 ± 0.99 (1)	1.10 ± 0.30 (1)	1.00 ± 0.00 (1)	2.10 ± 1.04 (2)	1.60 ± 1.02 (1)	4
I	2.30 ± 2.67 (0)	1.70 ± 1.90 (0)	0.00 ± 0.00 (0)	7.50 ± 1.91 (8)	6.10 ± 1.76 (5)	3
It	7.80 ± 5.18	87.10 ± 20.68	49.20 ± 16.35	469.50 ± 50.04	458.50 ± 100.82	1
K2noHC	–	−14,407.32 ± 5.01	−14,401.83 ± 0.72	−14,501.85 ± 40.59	−14,475.13 ± 43.49	–
DEst	44,381.60 ± 1116.15	(+) 43,349.60 ± 953.87	44,693.20 ± 1208.56	(−) 275,313.80 ± 12,030.43	(−) 389,112.00 ± 12,124.67	(+) 3375
TEst	31.16E05 ± 97.90E03	64.76E05 ± 11.67E04	75.48E05 ± 10.78E04	71.49E05 ± 10.22E05	69.76E05 ± 10.89E05	154,637
NVars	(*−) 5.40 ± 0.06	4.43 ± 0.02	4.44 ± 0.02	(*−) 5.03 ± 0.03	(*−) 5.14 ± 0.04	(*+) 2.99

Table 2
Results for INSURANCE

INSURANCE	$\mu \pm \sigma$ (best)					HCST (Best)
	ILS	ACO-B1	ACO-B2	UMDA	PBIL	
K2	(−) −57,918.90 ± 36.95 (−57,872.20)	−57,807.81 ± 36.68 (−57,763.07)	−57,808.49 ± 36.90 (−57,763.07)	(−) −57,984.19 ± 130.66 (−57,806.78)	(−) −57,940.79 ± 54.83 (−57,834.59)	(−) −57,998.10
KL	8.435 ± 0.036 (8.400)	8.450 ± 0.036 (8.487)	8.450 ± 0.033 (8.487)	8.436 ± 0.038 (8.409)	8.440 ± 0.033 (8.412)	8.423
A	8.78 ± 3.11 (7)	3.60 ± 1.99 (3)	3.87 ± 2.49 (3)	8.07 ± 3.44 (4)	7.60 ± 2.59 (3)	10.33
D	11.44 ± 2.19 (9)	8.50 ± 1.34 (8)	8.57 ± 1.45 (8)	10.67 ± 2.30 (8)	10.83 ± 2.00 (8)	11.67
I	6.44 ± 3.54 (3)	2.43 ± 2.09 (2)	3.13 ± 3.46 (2)	5.43 ± 3.51 (1)	6.37 ± 2.58 (1)	7.67
It	8.78 ± 4.94	80.20 ± 21.57	48.73 ± 24.63	378.93 ± 84.89	444.27 ± 65.92	1
K2noHC	–	−57,811.04 ± 37.27	−57,808.61 ± 37.21	−58,056.38 ± 135.71	−58,003.18 ± 72.60	–
DEst	(+) 27,610.56 ± 841.26	28,738.77 ± 882.93	(−) 29,498.63 ± 993.64	(−) 87,575.23 ± 6157.88	(−) 126,897.63 ± 5331.41	(+) 2050.33
TEst	14.11E05 ± 33.14E03	38.19E05 ± 33.44E03	43.48E05 ± 66.13E03	29.95E05 ± 58.03E04	34.77E05 ± 52.79E04	7.66E+04
NVars	(*−) 6.06 ± 0.10	4.35 ± 0.02	(*−) 4.38 ± 0.03	(*−) 4.80 ± 0.04	(*−) 4.88 ± 0.02	(*+) 3.09

Table 3
Results for BOBLO

BOBLO	$\mu \pm \sigma$ (best)					HCST (Best)
	ILS	ACO-B1	ACO-B2	UMDA	PBIL	
K2	(−) −11,903.12 ± 14.91 (−11,883.13)	−11,875.89 ± 2.12 (−11,873.68)	−11,876.27 ± 1.37 (−11,873.68)	(−) −11,900.11 ± 6.95 (−11,882.00)	(−) −11,898.92 ± 8.71 (−11,881.03)	(−) −11,927.00
KL	7.488 ± 0.005 (7.492)	7.492 ± 0.000 (7.492)	7.492 ± 0.001 (7.492)	7.489 ± 0.001 (7.491)	7.489 ± 0.001 (7.491)	7.476
A	16.40 ± 2.99 (13)	13.20 ± 1.72 (15)	12.40 ± 1.43 (15)	8.80 ± 1.17 (9)	8.60 ± 1.11 (10)	17
D	1.70 ± 1.42 (0)	0.40 ± 0.49 (1)	0.40 ± 0.66 (1)	1.70 ± 0.78 (1)	1.60 ± 0.80 (0)	3
I	6.40 ± 0.97 (6)	6.80 ± 0.60 (7)	6.70 ± 0.90 (7)	2.30 ± 0.90 (2)	2.20 ± 0.75 (3)	6
It	8.30 ± 4.97	48.50 ± 28.42	18.40 ± 14.04	327.30 ± 81.12	368.20 ± 73.83	1
K2noHC	–	−11,875.89 ± 2.12	−11,876.27 ± 1.37	−11,900.56 ± 7.13	−11,900.03 ± 9.15	–
DEst	(−) 21,854.10 ± 514.50	12,019.00 ± 378.71	(−) 12,647.40 ± 413.92	(−) 42,012.50 ± 2417.38	(−) 62,965.00 ± 2443.03	(+) 1255
TEst	98.11E04 ± 25.44E03	25.32E05 ± 38.72E03	28.16E05 ± 80.36E03	18.09E05 ± 39.96E04	20.83E05 ± 36.53E04	37,285
NVars	(*−) 6.27 ± 0.09	4.25 ± 0.04	(*−) 4.30 ± 0.04	(*−) 4.59 ± 0.03	(*−) 4.73 ± 0.02	(*+) 2.90

As a reference for the goodness of the results we can consider the K2 values for the *true* graphical structures, which are −14,412.69 for ALARM, −58,120.95 for INSURANCE and −11,907.09 for BOBLO. In order to obtain signification comparisons between the algorithms, a statistical analysis has been carried out: We have chosen the Mann–Whitney test for comparison between the stochastic algorithms and a *t*-test for comparison between the deterministic HCST and the stochastic algorithms. In all the cases a 5% significance level has been used. First, we have compared the two ACO-B algorithms and then we have chosen the best as the reference for comparisons. When significant differences are found, they are shown in the tables by a (+) if a positive difference is found and (−) if a negative difference is found for the case of K2 and DEst values, and by a (*+) or (*−) for the estimation of CPU time (although it appears in the rows corresponding to NVars).

From the analysis of these data we can draw the following conclusions:

- The best result found for ALARM has a K2 value of −14,401.29 and was found by the ACO-B algorithms. The same occurs for INSURANCE and BOBLO, where the best network found has a K2 value of −57,763.07 and −11,873.68 respectively. Notice that, in all cases, these results are considerably better than the reference values.
- With respect to the accuracy of the algorithms, it is clear that ACO-B algorithms improve on the results obtained by the rest of algorithms. This conclusion is valid for the K2 and KL values, and also for structural differences ($A + D + I$), except for the BOBLO domain, where ACO-B tends to include more arcs than EDA.
- From the results, it seems that the local search step carried out by applying HCST to the solutions obtained by the ants between iterations and in the last generation, does not significantly improve the quality of the networks. A different scheme of local optimization may work better. ACO-B2 only significantly improves ACO-B1 in the ALARM domain. In the remaining of domains both ACO algorithms obtain similar results.
- With respect to the efficiency of the algorithms, it is clear that the fastest is HCST. Of the evolutionary algorithms, ACO is the fastest. This affirmation can be clearly deduced from the number of different statistics computed (DEst) and from the number of variables involved in these statistics (NVars).
- As a general conclusion, we think that the use of heuristic knowledge by the ants when they are constructing a new sequence, helps to guide the search. For this reason, the results are of high quality and improve on the ones obtained by other methods. Furthermore, the use of this heuristic knowledge causes relatively small changes in the network being constructed with respect to those previously obtained, and so a great deal of computations can be reused, increasing the efficiency of the algorithm as well.

Following up this idea, we believe that a more *informed* initialization of the population in UMDA and PBIL, can help the process to be focused on promising regions of the search space.

5. Concluding remarks

In this paper a new scoring-based algorithm for learning BNs has been studied. The novelty of our method lies in the use of the ACO metaheuristic to guide the search process. This allows the algorithm to exploit heuristic knowledge about the problem, together with a simple but efficient form of cooperation between independent agents. This may be particularly important for large problems, because we can use distributed computation to obtain good solutions without increasing the computational cost. The experimental results are encouraging: Our ACO-B algorithms clearly outperform all the other algorithms based on different search methods, which we have used in this paper for comparative purposes.

For future research, our aim is to look more closely at the use of ACO for learning BNs, by refining the proposed algorithm (parameter fitting, other forms of local optimization, use of different types of ants or transition rules, etc.). More empirical studies are also required to definitively establish the validity of our approach. On the other hand, several authors [3,16,19,24,33] have successfully used the variable ordering space for learning BNs. It would also be interesting, therefore, to apply ACO to search in the space of orderings.

Acknowledgements

This work has been supported by the Spanish Ministerio de Ciencia y Tecnología (MCyT), under projects TIC2001-2973-C05-01 and TIC2001-2973-C05-05.

References

[1] S. Acid, L.M. de Campos, Benedict: an algorithm for learning probabilistic Bayesian networks, in: Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU'96), 1996, pp. 979–984.

[2] S. Acid, L.M. de Campos, A hybrid methodology for learning Bayesian networks: benedict, International Journal of Approximate Reasoning 27 (3) (2001) 235–262.

[3] S. Acid, L.M. de Campos, J. Huete, The search of causal orderings: a short cut for learning Bayesian networks, in: European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'2001), LNAI 2143, 2001, pp. 216–227.

[4] S. Baluja, Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning, Technical Report CMU-CS-94-163, Computer Science Department, Carnegie Mellon University, 1994.
[5] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, G.F. Cooper, The case study with two probabilistic inference techniques for Bayesian networks, in: Proceedings of the Second European Conference on Artificial Intelligence in Medicine, Springer-Verlag, 1989, pp. 247–256.
[6] J. Binder, D. Koller, S. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, Machine Learning 29 (2) (1997) 213–244.
[7] R. Blanco, I. Inza, P. Larrañaga, Learning Bayesian networks in the space of structures by Estimation of Distribution Algorithms, International Journal of Intelligent Systems, in press.
[8] R.R. Bouckaert, Bayesian belief networks: from construction to inference, Ph.D. thesis, University of Utrecht, 1995.
[9] W. Buntine, Theory refinement of Bayesian networks, in: Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1991, pp. 52–60.
[10] W. Buntine, A guide to the literature on learning probabilistic networks from data, IEEE Transactions on Knowledge and Data Engineering 8 (1996) 195–210.
[11] D.M. Chickering, D. Geiger, D. Heckerman, Learning Bayesian networks: search methods and experimental results, in: Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics, 1995, pp. 112–128.
[12] D.M. Chickering, D. Geiger, D. Heckerman, Learning Bayesian networks is NP-complete, in: D. Fisher, H. Lenz (Eds.), Learning from Data: Artificial Intelligence and Statistics V, Springer-Verlag, 1996, pp. 121–130.
[13] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9 (4) (1992) 309–348.
[14] D. Dash, M. Druzdel, A hybrid anytime algorithm for the construction of causal models from sparse data, in: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1999, pp. 142–149.
[15] L.M. de Campos, Independency relationships and learning algorithms for singly connected networks, Journal of Experimental and Theoretical Artificial Intelligence 10 (4) (1998) 511–549.
[16] L.M. de Campos, J.F. Huete, Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing, in: Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00), 2000, pp. 333–340.
[17] L.M. de Campos, J.F. Huete, A new approach for learning Bayesian networks using independence criteria, International Journal of Approximate Reasoning 24 (2000) 11–37.
[18] L.M. de Campos, J.M. Puerta, Stochastic local and distributed search algorithms for learning Bayesian networks, in: III International Symposium on Adaptive Systems (ISAS): Evolutionary Computation and Probabilistic Graphical Model, 2001, pp. 109–115.
[19] L.M. de Campos, J.M. Puerta, Stochastic local search algorithms for learning Bayesian networks: searching in the space of orderings, in: European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'2001), LNAI 2143, Springer, 2001, pp. 228–239.
[20] M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, 1999, pp. 11–33.
[21] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artificial Life 5 (2) (1999) 137–172.
[22] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1997) 53–66.
[23] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man and Cybernetics, Part B 26 (1996) 29–41.

[24] N. Friedman, D. Koller, Being Bayesian about network structure, in: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 2000, pp. 201–210.
[25] J.A. Gámez, J.M. Puerta, Searching the best elimination sequence in Bayesian networks by using ant-colony optimization, Pattern Recognition Letters 23 (1–3) (2002) 261–277.
[26] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Machine Learning 20 (1995) 197–244.
[27] M. Henrion, Propagating uncertainty by logic sampling in Bayes networks, in: J. Lemmer, L.N. Kanal (Eds.), Uncertainty in Artificial Intelligence, vol. 2, North Holland, Amsterdam, 1988, pp. 149–164.
[28] F.V. Jensen, Bayesian Networks and Decision Graphs, Springer-Verlag, 2001.
[29] S. Kullback, Information Theory and Statistics, Dover, New York, 1968.
[30] W. Lam, F. Bacchus, Learning Bayesian networks. An approach based on the MDL principle, Computational Intelligence 10 (4) (1994) 269–293.
[31] P. Larrañaga, Aprendizaje estructural y descomposición de redes Bayesianas vía algoritmos genéticos, Ph.D. thesis, University of Basque Country, 1995 (in Spanish).
[32] P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation, Kluwer Academic Press, 2001.
[33] P. Larrañaga, C. Kuijpers, R. Murga, Learning Bayesian network structures by searching for the best ordering with genetic algorithms, IEEE Transactions on System, Man and Cybernetics 26 (4) (1996) 487–493.
[34] P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, C. Kuijpers, Structure learning of Bayesian networks by genetic algorithms: a performance analysis of control parameters, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (9) (1996) 912–926.
[35] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1996.
[36] H. Mühlenbein, The equation for response to selection and its use for prediction, Evolutionary Computation 5 (1998) 303–346.
[37] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, 1988.
[38] J. Pearl, T.S. Verma, Equivalence and synthesis of causal models, in: Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence, 1990, pp. 220–227.
[39] L.K. Rasmussen, Bayesian Network for blood typing and parentage verification of cattle, Ph.D. thesis, Research Centre Foulum, Denmark, 1995.
[40] M. Singh, M. Valtorta, Construction of Bayesian network structures from data: a brief survey and an efficient algorithm, International Journal of Approximate Reasoning 12 (1995) 111–131.
[41] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction, and Search, Lecture Notes in Statistics 81, Springer-Verlag, 1993.