International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 18, No. 4 (2004) 607–625
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

LEARNING BAYESIAN NETWORKS IN THE SPACE OF ORDERINGS WITH ESTIMATION OF DISTRIBUTION ALGORITHMS

TXOMIN ROMERO*, PEDRO LARRAÑAGA† and BASILIO SIERRA‡

Department of Computer Science and Artificial Intelligence,
University of the Basque Country, Spain
**scsroast@si.ehu.es*
†ccplamup@si.ehu.es
‡ccpsiarb@si.ehu.es

The search for the optimal ordering of a set of variables in order to solve a computational problem is a difficulty that can appear in several circumstances. One of these situations is the automatic learning of a network structure, for example, a Bayesian Network structure (BN) starting from a dataset. Searching in the space of structures is often unmanageable, especially if the number of variables is high. Popular heuristic approaches, like Cooper and Herskovits's K2 algorithm, depend on a given ordering of variables. Estimation of Distribution Algorithms (EDAs) are a new paradigm for Evolutionary Computation that have been used as a search engine in the BN structure learning problem. In this paper, we will use two different EDAs to obtain not the best structure, but the optimal ordering of variables for the K2 algorithm: UMDA and MIMIC, both of them in discrete and continuous domains. We will also check whether the individual representation and its relation to the corresponding ordering play important roles, and whether MIMIC outperforms the results of UMDA.

Keywords: Bayesian networks; estimation of distribution algorithms; space of orderings; learning from data; experimental results.

1. Introduction

Let D be an observed data set containing a sample of size N obtained from a domain $\mathbf{X} = (X_1, \ldots, X_n)$ with n variables. Rather than obtaining the complete Probabilistic Graphical Model (PGM) to model the problem domain composed of the random variables based on the data set (e.g. the network structure and the conditional probabilities of a BN), obtaining the structure S is often sufficient for our purposes. For example, if we only want to use the PGM to discover conditional dependencies and independencies among the variables that appear in a set of cases, we do not need to determine the conditional probability assignment of the dependencies. This work only focuses on obtaining a network structure.

The main problem is that the space of structures grows exponentially with the number of nodes.[31] Network structure learning is, generally, an NP-hard problem.[9]

One of the assumptions that has become a standard to reduce the search space is to only consider the structures that, given an ordering $<_o$, fulfil $X_i \in \pi_j \Rightarrow i <_o j$ being X_i a variable of the structure and π_j the set of parents of X_j. This can be dangerous to some extent, as the set of structures obtained is quite sensitive to the search procedure and in certain domains we do not have information on the prior likelihood of a given structure. A subsequent search in the space of orderings plays down this handicap.

The advantage of working with orderings rather than structures appears in several works. Not only is the convergence of the solution faster and its sharpness smaller,[15] but the space itself is also much smaller ($n!$ rather than $2^{O(n^2)}$, where n is the number of nodes). Consequently, we do not usually have to artificially reduce the space (e.g. limiting the number of parents or rejecting very complex models) in the cases when we would have to if we only used the search in the space of structures. If the number of nodes is too large, practical problems arise again. Thus, we will have to use a certain number of heuristic algorithms both in one problem and in the other. Therefore, we have to select one paradigm to search in the space of structures with the ordering restriction on the parents (*Structure Search Paradigm*, SSP) and, on the other hand, another paradigm to search in the space of orderings (*Ordering Search Paradigm*, OSP). In this work, we will use a greedy hill-climbing method as the SSP (the K2 algorithm[10]) and the *Univariate Marginal Distribution Algorithm* (UMDA)[25] and *Mutual Information Maximization for Input Clustering* (MIMIC)[12] as the search strategies in the space of orderings.

The rest of the paper is organized as follows: in Sec. 2, we introduce Bayesian Networks. In Sec. 3, we review the early work on searching in the space of orderings. In Sec. 4, the Estimation of Distribution Algorithm (EDA) paradigm is presented. In Sec. 5, we describe the individual representation, introduce the networks that we have worked with and also explain why we chose the EDA parameters used in the experiments. In Sec. 6, we present the results of the experiments and in Sec. 7, the concluding remarks.

2. Bayesian Networks

Let $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ be an n-dimensional random variable, where x_i is an instantiation of X_i. A *Probabilistic Graphical Model* $(S, \theta_\mathbf{s})$ is a graphic structure $S = (\mathbf{X}, A)$ and a set of local parameters $\theta_\mathbf{s}$. $\mathbf{X}$, a set of nodes, represents the system variables and A, a set of arcs, the conditional dependence/independence relationships among the variables of the structure. A *Bayesian Network* (BN) is a PGM where the graphic structure is a directed acyclic graph (DAG), the X_i are random discrete variables (called *nodes*) and the set of parameters $\theta_\mathbf{s} = (\theta_{ijk})$, where k goes from 1 to r_i, j from 1 to q_i and i from 1 to n, represent the local probability distributions over $\mathbf{X}$, i.e. θ_{ijk} is the conditional probability of X_i being in its kth value given that the set of its parent variables is in its jth value. Finally, r_i is the number of different values of the ith variable and q_i

represents the number of possible instantiations for the set of parents of the ith variable.

The BN paradigm can be used as a model to make inferences in domains with intrinsic uncertainty. Although a human expert is able to construct the model, this method can be too subjective and time consuming. A substitute for this kind of construction is the automatic learning of the model from a given data set.

The factorization of the joint distribution that a BN represents permits an efficient reasoning inside the model. This reasoning, i.e. the propagation of evidence through the model, can be carried out using several algorithms. The most popular algorithm for the retrieval of beliefs was proposed by Lauritzen and Spiegelhalter in 1988.[23] It consists of a manipulation of the BN that makes it easier and faster to obtain the obtaining probabilities of the network nodes. Introductions and classic textbooks about BN are available in Refs. 8, 18 and 28.

3. Previous Work in Structure Learning in the Space of Orderings

We can find several solutions in the early work on learning structures from data. Most of the works try to learn structure searching only in the space of DAGs or equivalence classes, based on conditional independence tests or on scoring metric optimization (the score+search approaches). Another approach is searching in the space of orderings: we search in the space of structures with a given fixed ordering of the variables that determines the search itself, and we make an ulterior search in the space of orderings. We are interested in these methods. In this section, some of the works that use the last method are reviewed.

In Ref. 7, Bouckaert proposes the K2 structure learning algorithm and, as OSP, an algorithm that manipulates a given ordering $<_o$ of the variables with operations similar to arc reversal. Only the operations that generate a resulting DAG that represents the same independency model present in the structure before the application are performed, making the set of independences bigger at each step.

In Ref. 21, Larrañaga *et al.* used Genetic Algorithms[24] as OSP, where they learn the DAG using the K2 structure learning algorithm of Cooper and Herskovits.[10] The score of the DAG (and of the order that the K2 algorithm takes into account to obtain it) is the K2 metric. The GA used is based on the principles of GENITOR, developed by Whitley.[33] It combines several genetic operators (crossover and mutations) to examine the convergence of the GA and its fitness.

For Friedman and Koller,[15] the first goal is to compute $P(S|D) \propto P(D|S)P(S)$ in order to search in the space of structures. $P(D|S)$, the *marginal likelihood* of the data given the structure, is obtained integrating all possible BN parameters. To reduce the complexity, the authors restricted the space of structures limiting the number of parents of a node with a constant k and a given ordering $<_o$ to restrict the available parents of a node. They also used several computational tricks, such as reducing the possible parents of a node and the possible families to the highest-scoring ones by means of a precomputation. With these restrictions they could estimate $P(S|D, <_o)$. To search in the space of the orderings, the authors constructed a homogeneous Markov chain with the state space being the $n!$ possible orderings of the variables. In this case, the construction of the Markov chain using a standard Metropolis algorithm sampling needed to guarantee that the invariant distribution over the states is $P(<_o |D)$.

Acid *et al.*[2] used the BENEDICT-*step* algorithm as an SSP. This algorithm is based on a hybrid methodology that combines independence tests and scoring metrics. The scoring metric measures the discrepancies between the d-separation statements of a candidate network S and the one that appears in D using the Kullback–Leibler cross entropy. However, if the number of nodes of S is large, computing problems to calculate the d-separation statements arise, as their number and complexity grow exponentially. Thus, given a candidate network S, they only take into account the conditional independences for every two nonadjacent single variables X_i and X_j, given the set of minimum size, $D_s(X_i, X_j)$, that d-separates both of them.[1] These d-separation statements are representative of all the d-separations that appear in S. The search method that uses the metric is a simple greedy algorithm that takes into account, as the measure of global discrepancy, the sum of the local discrepancies and adds one arc only if the addition makes the global discrepancy lower. This addition needs to respect a given order. As the OSP, they use the Simulated Annealing algorithm.[19]

In Ref. 13, de Campos and Puerta regarded the K2 metric as the scoring function to evaluate the quality of a structure. To obtain the structure, they used a simple hill-climbing search of the best parent set of each node with operators of arc addition and arc removal. As OSP, the authors used the Variable Neighborhood Search[16] based on orderings. First, they selected a set of neighborhood orderings N_k (with $k = 1, \ldots, k\max$), where $N_1(<_o)$ is the neighborhood of the ordering $<_o$ that results from the interchange of any two positions and, in general:

$$<''_o \in N_k(<_o) \Leftrightarrow <''_o \in N_1(<'_o) \wedge <'_o \in N_{k-1}(<_o) \qquad (1)$$

and then, they searched in the space of these neighborhood orderings (at first, in N_k, and if there is no improvement, in N_{k+1}): after selecting a random ordering from N_k, they used a hill-climbing search in the space of the orderings with an operator of interchange of two positions. To speed up the search process they established a number r that limits the distance between the variables to be interchanged in an ordering.

The search in the space of orderings proposed by Puerta[30] combines the K2SN algorithm with the Ant Colonies.[14] K2SN is a modification of the K2 algorithm that does not need a previous ordering, because K2SN itself determines the ordering. At each step, K2SN uses the K2 algorithm to discover the node, and the set of parents that maximizes the K2 metric, adding the node to the network and, therefore, setting its position. In an ant colony, the election of a node is not only based on the K2 metric, but on a pheromone quantity too. The more pheromone quantity you have in a particular edge, the more you obtain this edge election probability. Each

Table 1. Summary of works on structure learning in the space of orderings.

Authors	SSP + OSP
Bouckaert[7]	K2 + Independences preservation
Larrañaga *et al.*[21]	K2 + Genetic algorithms
Friedman and Koller[15]	Space restriction + MCMC
Acid *et al.*[2]	BENEDICT-step + Simulated annealing
de Campos and Puerta[13]	Hill-climbing + VNS based on orderings
Puerta[30]	Hill-climbing + K2SN and Ant colonies

time the edge is selected by an ant, in this case an execution of the Ant + K2SN algorithm, its pheromone quantity increases. For ulterior local search in the space of each ordering proposed by the ant colony, the author proposes a hill-climbing algorithm described in Ref. 13.

In Table 1, we can see a summary of this section.

4. Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDA)[20,22,26] are a new paradigm for Evolutionary Computation. As GAs, EDAs are population-based stochastic searches that replace the crossover and mutation operators by learning the probability distribution of the best individuals of each generation and its posterior simulation. Thus, we can capture all the relationships among the variables in an explicit way. To the best of our knowledge, EDAs have been used as a search engine in the BN structure learning problem in Refs. 5, 6 and 29.

4.1. *General form of an EDA algorithm*

Before showing the pseudocode of the EDA approach,[22] we will establish some definitions:

- $z = (z_1, \ldots, z_m)$ will denote the instantiation of the m-dimensional variable $\mathbf{Z} = (Z_1, \ldots, Z_m)$ called individual of m genes.
- D_l will denote the population of M individuals in the lth generation.
- D_l^s will denote the population that makes up the set of N individuals selected from D_l.
- $\rho_l(\mathbf{z}) = \rho_l(\mathbf{z}|D_{l-1}^s)$ will denote the joint generalized probability density function (or the mass probability if each Z_i is a random discrete variable) of $\mathbf{z}$ individual at the lth generation, given D_{l-1}^s.

The general form of the EDA algorithm proposed in this work for searching in the space of BN orderings can be seen in Fig. 1.

The main problem in this approach is the estimation of the probability distribution $\rho_l(\mathbf{z})$, even though there are other factors that can be important, like the

```
EDA
    $D_0 \leftarrow$ Generate $M$ individuals randomly (the initial population)
    $l \leftarrow 1$
    Repeat
        For each individual $i_i \in i_1, i_2, \ldots, i_M$ do
            Obtain the ordering $o_i$ associated to $i_i$
            Learn the network using the K2 algorithm and the ordering $o_i$
            Calculate the score $s_i$ of the network/ordering/individual using the K2 metric
            If $s_i$ is the best score found then $b_i \leftarrow i_i$ End if
        End for
        $D^s_{l-1} \leftarrow$ Select $N \leq M$ individuals from $D_{l-1}$ using a fixed selection
        method
        $\rho_l(\boldsymbol{z}) \leftarrow$ Estimate the probability distribution of $\boldsymbol{z}$ given $D^s_{l-1}$
        $D_l \leftarrow$ Sample $M$ individuals (the new population from $\rho_l(\boldsymbol{z})$
        $l \leftarrow l + 1$
    Until a stopping criterion is verified
    Return $b_i$
```

Fig. 1. Main scheme of the EDA approach.

initialization of the initial generation, the selection of the individuals used to estimate the probability distribution, etc. To estimate $\rho_l(\mathbf{z})$ and then sample the new generation, EDAs construct and use a BN if the variables that the individuals are made of are discrete, and a Gaussian Network (GN)[32] if they are continuous. A GN is a PGM where S is a DAG, the random variables of $\mathbf{Z}$ are continuous and the set of parameters $\theta_{\mathbf{i}} = (m_i, \mathbf{b}_i, v_i)$ determines the local density function (the linear-regression model):

$$f(z_i|\pi_i^S, \theta_i) = \mathcal{N}\left(z_i; m_i + \sum_{Z_j \in \pi_i^S} b_{ji}(z_j - m_j), v_i\right) \tag{2}$$

where $\mathcal{N}(z; \mu, \sigma^2)$ is a univariate normal distribution with means μ and variance σ^2. $\mathbf{b}_i = (b_{1i}, \ldots, b_{(i-1)i})^t$ is a column vector where $b_{ji} = 0$ if there is no arc from Z_j to Z_i in S. It reflects the intensity of the dependences of every variable. m_i is Z_i's unconditional mean and v_i is the conditional variance given π_i^S.

4.2. *The probability distribution estimation*

Several EDAs have been proposed using both kind of variables (discrete and continuous). In this work, we will use two examples of EDAs: *Univariate Marginal Distribution Algorithm* (UMDA) and the *Mutual Information Maximization for Input Clustering* (MIMIC). Both will be used in discrete and continuous domains.

We will present both algorithms in the discrete domain. Their adaptation to continuous domains — denoted by UMDA$_c$ and MIMIC$_c$ — can be consulted in Ref. 22.

The UMDA algorithm, introduced by Mühlenbein,[25] assumes independence between the variables (the genes of the individuals), i.e. the model used to estimate $p_l(\mathbf{z})$ is the simplest, using the marginal frequencies to get the probability distribution:

$$p_l(\mathbf{z}) = p(\mathbf{z}|D^s_{l-1}) = \prod_{i=1}^{m} p_l(z_i) \tag{3}$$

and

$$p_l(z_i) = \frac{\sum_{j=1}^{N} \delta_j(Z_i = z_i|D^s_{l-1})}{N} \tag{4}$$

where $\delta_j(Z_i = z_i|D^s_{l-1})$ is 1 if in the jth case D^s_{l-1}, Z_i is equal to z_i and 0 otherwise.

MIMIC, proposed by De Bonet *et al.*,[12] takes dependencies between pairs of variables into account. The main idea is to search for the best permutation among the variables, in every generation, in order to obtain the closest probability distribution $p^\pi_l(\mathbf{z})$ to the empirical distribution of D^s_l, using the Kullback–Leibler distance:

$$p^\pi_l(\mathbf{z}) = p_l(z_{i_1}|z_{i_2}) \cdot p_l(z_{i_2}|z_{i_3}) \cdots p_l(z_{i_{m-1}}|z_{i_m}) \cdot p_l(x_{i_m}) \tag{5}$$

where $\pi = (i_1, i_2, \ldots, i_m)$ is a permutation of the indexes $1, 2, \ldots, m$. De Bonet *et al.* obtained one approximation of π in each generation using a straightforward greedy algorithm through the $m!$ possible permutations.

4.3. *Sampling the new generation*

In order to obtain a new population of individuals, it is enough to create a database where the probabilistic relationships among the variables are underlying.

In this work, we will use the *Probabilistic Logic Sampling* (PLS) method proposed by Henrion.[17] With PLS, the instance of a variable is generated after all its parents have already been sampled, using the distribution $p(z_i|\pi_i)$. Thus, variables are in ancestral ordering before the simulation. In the case of GNs, normal distribution simulation is carried out by adding values coming from uniform distributions.

5. Searching in the Space of Orderings with EDAs

Next, we describe the adaptation of the previous ideas to the search in the space of orderings with EDAs.

5.1. *Score to evaluate each ordering*

As noted earlier, in this work we use the K2 algorithm proposed by Cooper and Herskovits[10] and its K2 metric as the scoring function to evaluate the quality of an ordering:

$$P(S|D) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{6}$$

where q_i is the number of possible different instances of π_i in D, N_{ijk} the number of cases in D in which X_i is instanced as kth possible value and π_i has its jth instantiation, r_i the number of possible values of X_i and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. This is only true if the cases occur independently, if the density of the parameters, given a structure, is uniform and if we are working with a complete data set, i.e. without missing values. From (6), the scoring function associated with variable X_i is:

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{7}$$

The K2 algorithm is a greedy algorithm that searches for every node the set of parents X_i that maximizes the $g(i, \pi_i)$ function, adding a parent at each step until the addition does not improve the score $g(i, \pi_i)$.

5.2. *Individual representation*

The evolution of the EDAs provides us with different individuals of m genes $(z_1, z_2, \ldots, z_m)$ that have to be univocally associated to a specific ordering of n variables. With continuous EDAs, the representation of the individuals is direct, i.e. the genes of the individual only have to be ordered and each gene instantiation is associated to its respective number of nodes to obtain a valid ordering, i.e. in this case $m = n$. For example, from the continuous individual $(0.5, 1.6, 0.2, 0.1)$, we obtain the ordering $(3 - 4 - 2 - 1)$. It must be taken into account that the redundancy of this representation is high due to the fact that different continuous individuals can generate the same ordering.

In the event of discrete domains, the individual representation is not so simple. If we have four variables and 4! possible permutations or orderings, we cannot use an individual of four genes whose variables can take four instantiations at the most, because we can obtain, for example, the instantiation $(2 - 3 - 4 - 4)$, which is not a valid ordering.

Even though this difficulty can be overcome by penalizing the individuals that do not verify the restrictions, or by even adapting the simulation process to sample valid individuals,[4] both can have a negative influence on the behavior. We have developed a solution that avoids the disadvantages of prior aproximations and that makes the individual-ordering association bijective.

We can determine a particular ordering from the $n!$ possible permutations with the factor decomposition of $n!$ For example, if we have four variables, the possible 4! orderings can be generated in a systematic form, shown in Table 2. The decomposition of 4! is $4 \cdot 3 \cdot 2 \cdot 1$. If we represent the individual with three variables Z_1, Z_2, Z_3 with $r_1 = 4$, $r_2 = 3$ and $r_3 = 2$, it is easy to obtain a particular order of the systematic list from an individual.

5.3. *Networks used in the experiments*

In the experiments we use the *Asia* and *Alarm* belief networks. The Asia network (see Fig. 2) was created by Lauritzen and Spiegelhalter[23] for example, purposes only.

Table 2. Possible orderings of four nodes.

1234	2134	3214	4231
1243	2143	3241	4213
1324	2314	3124	4321
1342	2341	3142	4312
1432	2431	3412	4132
1423	2413	3421	4123

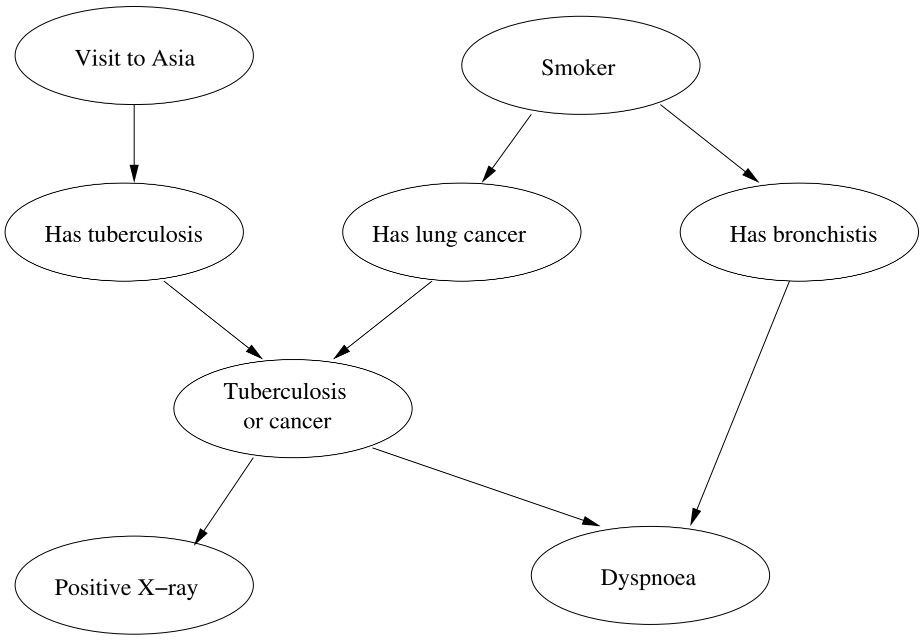


Fig. 2. The Asia network.

It can be seen as a network aimed at diagnosing patients. It has eight variables, each one with two possible states, and eight arcs. The database of 3,000 cases used in the experiments was generated with Netica from Norsys Software Corp.[11]

The Alarm network (see Fig. 3) is a classical benchmark for the evaluation of learning algorithms. It stands for *A Logical Alarm Reduction Mechanism.* Its goal is to be a medical diagnostic alarm message system for patient monitoring. It was first described by Beinlich *et al.*[3] It has 37 variables (with 2, 3 and, at most, 4 states) and 46 arcs. The Alarm database can be obtained from the UCI repository of machine learning databases.[27] From this database, we only used the first 3,000 cases.

5.4. *Choices about the parameters of EDAs*

The initial generation, in our work, is generated randomly.

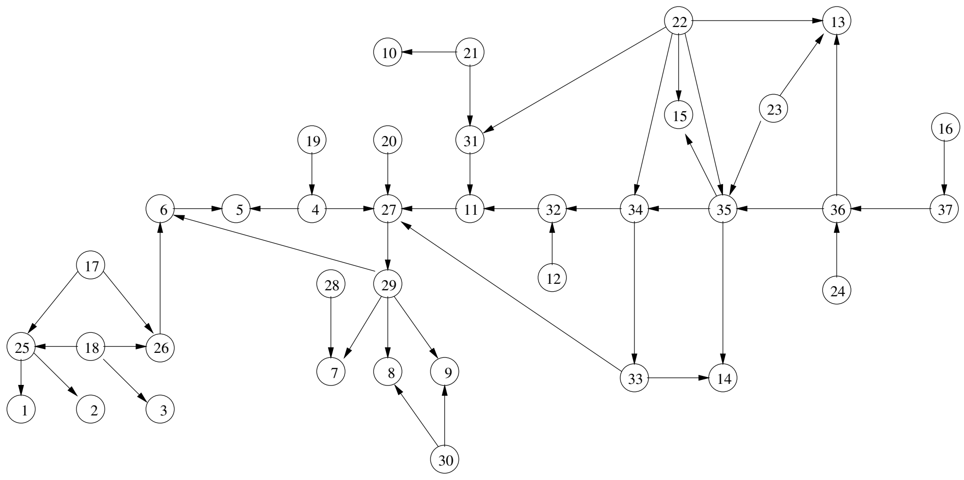


Fig. 3. The Alarm network.

In the experiments, we always use a population size of $M = 100$ individuals for Asia and $M = 500$ individuals for Alarm. In order to select the individuals used to estimate the probability distribution, we select the first $N = \frac{M}{2}$ best individuals from the actual population, i.e. the individuals with the best score (taking into account the K2 metric). We only keep the best individual from one generation to the next, sampling $M - 1$ individuals at each step or generation. To keep a high percentage of individuals among generations worsens the results, as EDAs cannot evolve.

Finally, the stopping method is not related to a convergence criterion. Normally, the convergence criterion, whatever the method, does not always guarantee the termination of the algorithm. In our work, we create a sample of the best individual of the current generation evaluated by each M different orderings. We stop when more than $M \cdot 50$ different orderings are evaluated, in order to compare the results with a random experiment, where we generate $M \cdot 50$ different orderings (i.e. we generate 50 samples in each experiment). With Asia, the repetition of the orderings in the last generations is numerous; therefore, if we visit M consecutive orderings that have been evaluated in old generations, we make a sample and sum M to the counter of different individuals evaluated. With Alarm, the algorithm cannot stop when exactly $M \cdot 50$ individuals have been evaluated, as the current generation has to be finished.

5.5. *Information collected*

In each sample of the best individual of the actual generation, we collect the following information:

- The value of the K2 metric (log). This is the metric to be maximized by the EDAs.
- The Hamming Distance (HAD) between the original network structure (Asia or Alarm) and the DAG obtained by the K2 Algorithm.
- The Skeleton Distance (SKD) takes into account the undirected graph that results from the DAG generated by the K2 algorithm and the one from the original network. It counts 1 for each arc that appears in one network and not in the other.
- The Mixed Distance (MID) counts 1 for each directed arc that appears in one network and not in the other, except for the arcs that appear in the other network but with an opposite direction, in which case it counts 0.5.

Note that the best evaluation (the value of the K2 metric) does not necessarily imply the best value for the structural distances for a given network.

6. Experimental Results

The experiments have been carried out ten times. In Figs. 4 and 5, the histogram of the distribution of the K2 metric for the 8! possible permutations of the nodes for the Asia network appears, as well as for 20,000 random orderings using the K2 search algorithm. In Fig. 6, the histogram for 20,000 random orderings for the

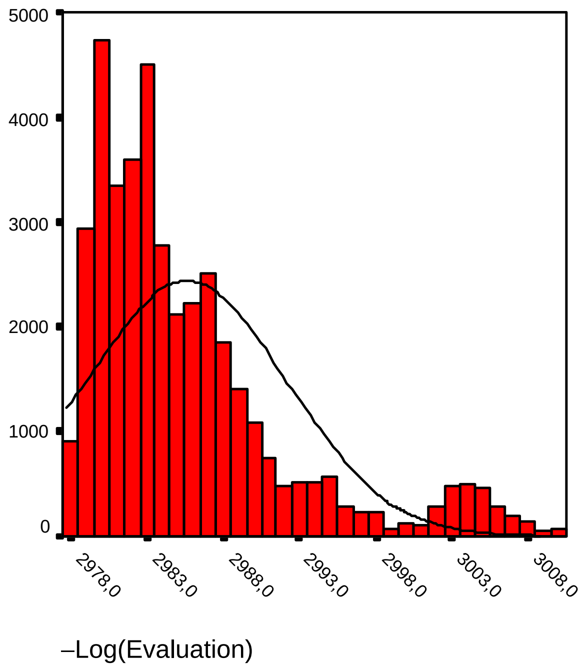


Fig. 4. Distribution of the K2 metric for the 8! possible orderings for the Asia network.

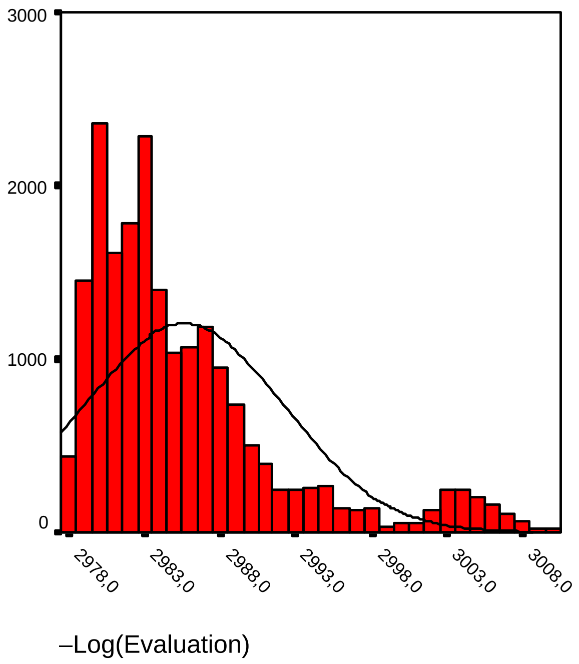


Fig. 5. Distribution of the K2 metric for 20,000 random orderings for the Asia network.

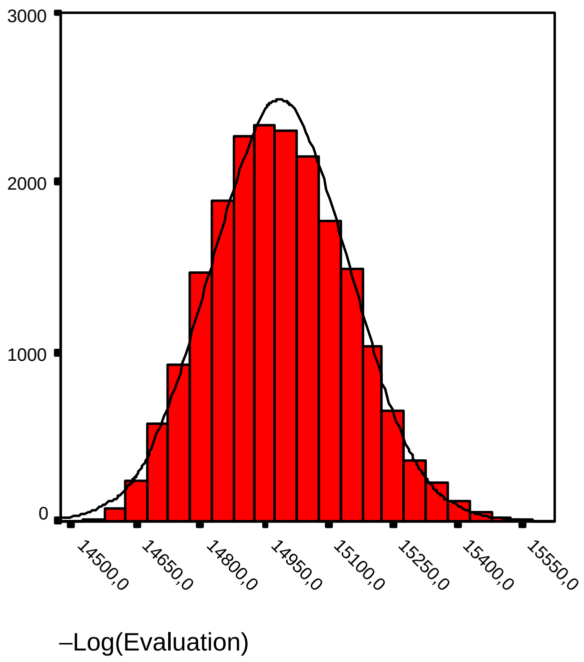


Fig. 6. Distribution of the K2 metric for 20,000 random orderings for the Alarm network.

Alarm network appears. Whereas the problem with Asia is easy because most of the orderings have a good score, the one for Alarm seems to be more difficult.

Experiments — not presented in this paper — with a version of the K2 algorithm that adds at each step not only one but the two parents that most improve the K2 metric, make the histogram even worse, moving the majority of the orderings to worse areas of the metric. This is because the algorithm determines the generated network too much (for example, it can have only an even number of parents for each node), which proves that the SSP is very important in structure learning. Note that, in certain domains (for example, the space of the structures for Asia or Alarm network), the highest scoring model is orders of magnitude more likely than any other, when the amount of data is large in relation to its size. However, in our case, searching in the space of the orderings, there are many K2-structures that can explain the data set well.

In Table 3, we can see the average best results (and its variance) for the Asia network and Continuous EDA (the results for Discrete EDA and the random experiment are the same), and in Tables 4–6 for Alarm. We can see that, with Asia, the K2 algorithm is able to obtain networks with a better score than Asia itself with all the methods, whereas the one for the Alarm network is the best scored structure. Seeing how the problem with Asia is easier, all the methods can reach the best ordering (even the random experiment).

The best average evaluation is obtained with MIMIC. MIMIC seems to outperform UMDA in discrete domains, whereas in continuous domains the behavior is almost the same.

In Table 7, we can see the evaluation of the best individual found for each method using Alarm network.

Table 3. Average results of ten executions for Asia and Continuous EDA.

	MIMIC_c	UMDA_c
Evaluation	-2977.52 ± 0.00	-2977.52 ± 0.00
HAD	2.00 ± 0.00	2.00 ± 0.00
SKD	1.00 ± 0.00	1.00 ± 0.00
MID	2.50 ± 0.00	2.50 ± 0.00
%Asia (-2979.16)	100.06 ± 0.00	100.06 ± 0.00

Table 4. Average results of ten executions for Alarm and Continuous EDA.

	MIMIC_c	UMDA_c
Evaluation	-14463.80 ± 213.23	-14463.13 ± 74.91
HAD	10.00 ± 7.40	10.30 ± 4.61
SKD	2.90 ± 0.49	2.60 ± 0.44
MID	13.15 ± 12.15	13.25 ± 5.51
%Alarm (-14412.69)	99.65 ± 0.01	99.65 ± 0.00

Table 5. Average results for ten executions for Alarm and Discrete EDA.

	MIMIC	UMDA
Evaluation	-14423.96 ± 27.41	-14453.54 ± 69.45
HAD	4.20 ± 0.76	11.40 ± 5.44
SKD	1.40 ± 0.44	2.60 ± 0.44
MID	5.50 ± 1.15	15.40 ± 10.29
%Alarm (−14412.69)	99.92 ± 0.00	99.72 ± 0.00

Table 6. Average results of ten executions for Alarm and random experiment.

	Random Experiment
Evaluation	-14522.30 ± 140.66
HAD	16.70 ± 9.21
SKD	3.40 ± 0.64
MID	22.25 ± 6.81
%Alarm (−14412.69)	99.25 ± 0.01

Table 7. Best scores found for the Alarm network.

MIMIC$_c$	−14437.23
UMDA$_c$	−14453.35
MIMIC	−14417.86
UMDA	−14435.87
Random Experiment	−14499.73

In Fig. 7, we can see the evolution over the 50 samples of the average evaluation of ten experiments for UMDA, MIMIC and the random experiment for Asia network, and in the next the same evolution for Alarm network.

The search algorithm with the best score is, undoubtedly, MIMIC: the Mann–Whitney test obtains a significance level of $p < 0.001$ among it and the other methods. However, MIMIC$_c$ does not have statistically significant differences relative to UMDA$_c$, and UMDA outperforms it (significance level of $p = 0.034$). Finally, UMDA outperforms UMDA$_c$ with a significance level of $p = 0.049$.

Note that the network obtained with the K2 algorithm and a given ordering does not have to be exactly the original network (Alarm or Asia) at the end of an execution of an EDA, because we are dealing with databases generated with the original network but inevitably incomplete. We do not have all the information needed on the database to generate the original network. In Fig. 9, we can visualize the evolution of the network structure throughout five samples of the execution of an EDA (after the Alarm itself). The first sample has a metric of $-14,536.26$ and a Mixed Distance of 24.0. The last sample has a metric of $-14,417.86$ and a Mixed Distance of 4.5.

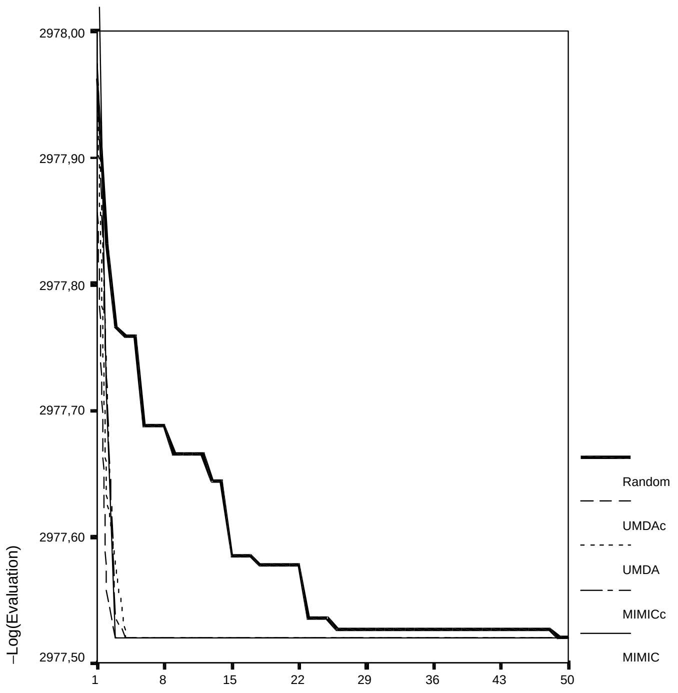


Fig. 7. Evolution over the 50 samples of the average evaluation for Asia network.

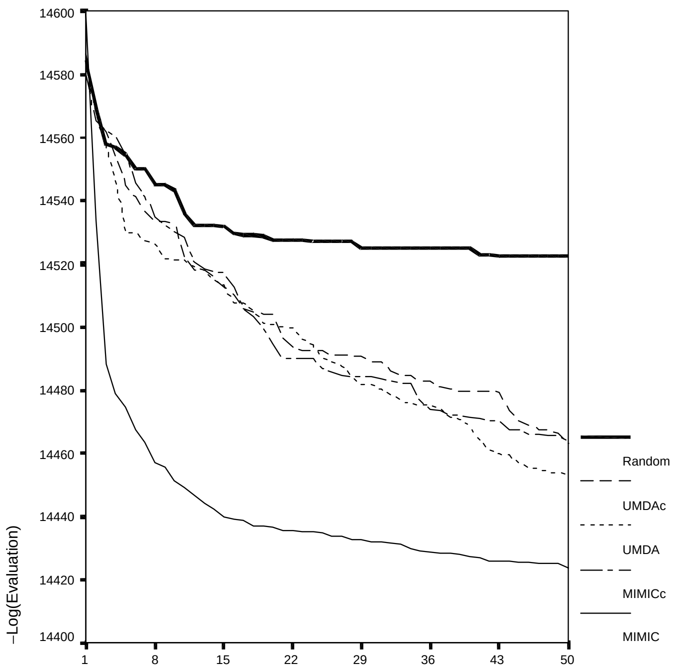


Fig. 8. Evolution over the 50 samples of the average evaluation for Alarm network.

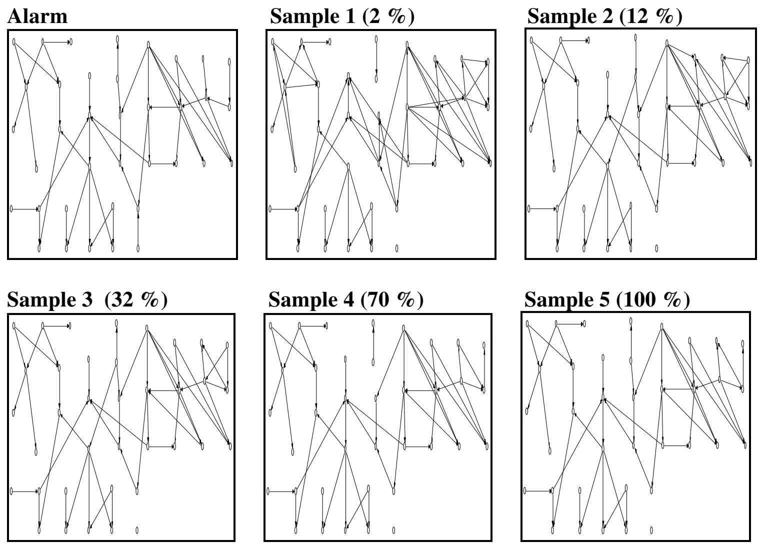


Fig. 9. Evolution of the K2 network over five samples for the experiments with Alarm. The first network is Alarm itself. EDA execution percentage appears next to each sample.

7. Concluding Remarks

In this work we have introduced a new approach to the learning of Bayesian Network from data. The search is done in the space of orderings using a number of instantiations of a new evolutionary computation metaheuristic called Estimation of Distribution Algorithms. We also present two different encodings: a continuous one and a discrete one. The results obtained in the experiments carried out with the different approaches of the Asia and Alarm networks show that the method is competitive with the state of art. Moreover, we have confirmed that the representation of the individual is very important in problems related to orderings, because if it is redundant, i.e. if we can associate two or more different individuals with the same ordering, the results are clearly worse. Regarding the different EDAs and domains, we can see in Fig. 8 that, in discrete domains, MIMIC outperforms all the other methods.

Regarding future work, we plan to study different initialization methods in order to estimate the importance of the initial generation in EDAs.

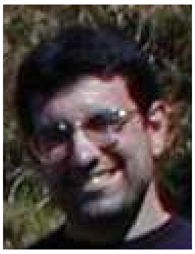
Acknowledgments

This work is partially supported by the University of the Basque Country, by the Department of Education, University and Research of the Basque Government and by the Ministry of Science and Technology under grants 9/UPV/EHU 00140.226-12084/2000, PI 1999-40 and TIC2001-2973-C05-03, respectively.

References

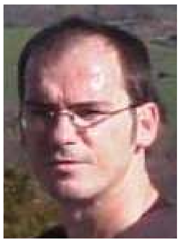
1. S. Acid and L. M. de Campos, An algorithm for finding minimum d-separating sets in belief networks, *Conf. Uncertainty in Artificial Intelligence*, 1996, pp. 3–10.
2. S. Acid, L. M. de Campos and J. F. Huete, The search of causal orderings: a short cut for learning belief networks, *European Conf. Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 2001, pp. 216–227.
3. I. A. Beinlich, H. J. Suermondt, R. M. Chavez and G. F. Cooper, The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks, *Second European Conf. Artificial Intelligence in Medicine*, 1989, pp. 247–256.
4. E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant and C. Boeres, Learning and simulation of Bayesian networks applied to inexact graph matching, *Patt. Recogn.* **35** (2002) 2867–2880.
5. R. Blanco, I. Inza and P. Larrañaga, Learning Bayesian networks from data by novel population-based, stochastic search algorithms, *9th Spanish Association for Artificial Intelligence Conf.*, 2001.
6. R. Blanco, I. Inza and P. Larrañaga, Learning Bayesian network structures by estimation of distribution algorithms, *Int. J. Intell. Syst.* **18** (2003) 205–220.
7. R. R. Bouckaert, Optimizing causal orderings for generating DAGs from data, *Uncertainty in Artificial Intelligence*, Vol. 8 (Morgan Kaufmann Publishers, 1992), pp. 9–16.
8. E. Castillo, J. Gutierrez and A. Hadi, *Expert Systems and Probabilistic Network Models* (Springer-Verlag, NY, 1997).
9. D. Chickering, D. Geiger and D. Heckerman, Learning Bayesian networks is NP-Hard, Technical Report MSR-TR-94-17 (November 1994).
10. G. F. Cooper and E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, *Mach. Learn.* **9** (1992) 309–347.
11. Norsys Software Corp., Netica software package. http://www.norsys.com, 1998.
12. J. S. de Bonet, C. L. Isbell and P. Viola, MIMIC: finding optima by estimating probability densities, *Advances in Neural Information Processing Systems*, Vol. 9, 1997.
13. L. M. de Campos and J. M. Puerta, Stochastic local algorithm for learning belief networks: Searching in the space of orderings, *European Conf. Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 2001, pp. 228–239.
14. M. Dorigo and G. Di Caro, The ant colony optimization meta-heuristic, in *New Ideas in Optimization*, eds. D. Corne, M. Dorigo and F. Glover (McGraw-Hill, London, 1999), pp. 11–32.
15. N. Friedman and D. Koller, Being Bayesian about network structure. An Bayesian approach to structure discovery in Bayesian networks, *Mach. Learn.* **50** (2003) 95–125.
16. P. Hansen and N. Mladenovi, An introduction to variable neighborhood search, in *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization* (Kluwer Academic, 1999), pp. 433–458.
17. M. Henrion, Propagating uncertainty in Bayesian networks by probabilistic logic sampling, *Uncertainty in Artificial Intelligence*, Vol. 2, 1988.
18. F. V. Jensen, *An Introduction to Bayesian Networks* (University College of London, 1996).
19. S. Kirpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science* **220** (1983) 671–680.
20. P. Larrañaga, R. Etxeberria, J. A. Lozano and J. M. Peña, Optimization by learning and simulation of Bayesian and Gaussian networks, Technical Report EHU-KZAA-IK-4/99, University of the Basque Country, 1999.

21. P. Larrañaga, C. M. H. Kuijpers, R. H. Murga and Y. Yurramendi, Learning Bayesian nework structures by searching for the best ordering with genetic algorithms, *IEEE Trans. Syst. Man Cybern. Part A*: *Syst. Humans* **26** (1996) 487–493.
22. P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation* (Kluwer Academic, 2001).
23. S. L. Lauritzen and D. J. Spiegelhalter, Local computations with probabilities on graphical structures and their application on expert systems, *J. Roy. Stat. Soc. B* **50** (1998) 157–224.
24. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, Berlin, 1992).
25. H. Mühlenbein, The equation for response to selection and its use for prediction, *Evolut. Comput.* **5** (1998) 303–346.
26. H. Mühlenbein and H. Voigt, Gene pool recombination in genetic algorithms, in *Metaheuristics*: *Theory and Applications*, eds. J. P. Kelly and I. H. Osman (Kluwer Academic, 1996), pp. 53–62.
27. P. M. Murphy and D. W. Aha, UCI repository of machine learning databases, http://www.ics.uci.edu/mlearn/MLRepository.html, 1995.
28. J. Pearl, *Probabilistic Reasoning in Intelligent Systems* (Morgan Kauffmann, Palo Alto, CA, 1988).
29. J. M. Peña, J. A. Lozano and P. Larrañaga, Unsupervised learning of Bayesian networks via estimation of distribution algorithms: an application to gene expression data clustering, *Int. J. Uncert. Fuzz. Knowledge-Based Syst.* **12** (2003) 63–82.
30. J. M. Puerta, *Local and Distributed Methods for the Induction of Statistics and Dynamics Belief Networks*, Ph.D. dissertation, E. T. S de Ingenieria Informatica, Universidad de Granada, 2001 (in Spanish).
31. R. Robinson, Counting unlabelled acyclic digraphs, in *Lecture Notes in Mathematics*: *Combinatorial Mathematics V* (Springer-Verlag, 1997), pp. 28–43.
32. R. S. Shachter and C. R. Kenley, Gaussian influence diagrams, *Manag. Sci.* **35** (1989) 527–550.
33. D. Whitley, The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best, in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 116–121.



Txomin Romero is a Ph.D. student at the University of the Basque Country (UPV/EHU) and is also Head of the Computer Center at the Donostia International Physics Center (DIPC). He is a member of the Intelligent System Group at the Department of Computer Science and Artificial Intelligence at UPV/EHU.

His research interests include machine learning, probabilistic graphical models, estimation of distribution algorithms and partial abductive inference.



Basilio Sierra is presently the Director of Computer Sciences and Artificial Intelligence Department at the University of the Basque Country, as well as the coordinator of the Robotics and Autonomous Systems Group in Donostia-San Sebastián. He has specialized in computer sciences at the University of the Basque Country. Dr. Sierra is presently a researcher in the fields of robotics and machine learning, specially in the use of Bayesian networks and neural networks for behavior design.

Pedro Larrañaga is a Professor in the Department of Computer Science and Artificial Intelligence at the University of the Basque Country where he has been leading the Intelligent Systems Group since 1996. He is author or coauthor of three books and more than 70 chapter books and papers in refereed journals in the fields of probabilistic graphical models and evolutionary computation with applications in medicine and computational biology. Recently he has served as guest editor for the journals *Machine Learning*, *Evolutionary Computation* and *Artificial Intelligence in Medicine*.