International Journal of Approximate Reasoning 50 (2009) 472–484

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Triangulation of Bayesian networks with recursive estimation of distribution algorithms

Txomin Romero [a,b,*], Pedro Larrañaga [c]

[a] Department of Computer Science and Artificial Intelligence, University of the Basque Country, Spain
[b] Donostia International Physics Center, Donostia – San Sebastian, Paseo Manuel de Lardizabal 4, 20018 Guipuzcoa, Spain
[c] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Spain

ARTICLE INFO

Article history:
Received 5 September 2007
Received in revised form 18 September 2008
Accepted 18 September 2008
Available online 7 October 2008

Keywords:
Bayesian networks
Estimation of distribution algorithms
Triangulation
Inference
Junction tree

ABSTRACT

Bayesian networks can be used as a model to make inferences in domains with intrinsic uncertainty, that is, to determine the probability distribution of a set of variables given the instantiation of another set. The inference is an NP-hard problem. There are several algorithms to make exact and approximate inference. One of the most popular, and that is also an exact method, is the evidence propagation algorithm of Lauritzen and Spiegelhalter [S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application on expert systems, Journal of the Royal Statistical Society B 50 (2) (1988) 157–224], improved later by Jensen et al. [F.V. Jensen, S.L. Lauritzen, K.G. Olesen, Bayesian updating in causal probabilistic networks by local computations, In Computational Statistics Quaterly 4 (1990) 269–282]. This algorithm needs an ordering of the variables in order to make the triangulation of the moral graph associated with the original Bayesian network structure. The effectiveness of the inference depends on the variable ordering. In this paper, we will use a new paradigm for evolutionary computation, the estimation of distribution algorithms (EDAs), to get the optimal ordering of the variables to obtain the most efficient triangulation. We will also present a new type of evolutionary algorithm, the recursive EDAs (REDAs). We will prove that REDAs improve the behaviour of EDAs in this particular problem, and that their results are competitive with other triangulation techniques.

1. Introduction

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 PGM or *Probabilistic Graphical Model* $(S, \boldsymbol{\theta}_\mathbf{s})$ is a graphical structure $S = (\mathbf{X}, \mathscr{A})$ and a set of local parameters $\boldsymbol{\theta}_\mathbf{s}$. $\mathbf{X}$, a set of nodes, represents the system variables and $\mathscr{A}$, a set of arcs, the conditional dependence/independence relationships among the variables of the structure. A *Bayesian network* (BN) is a PGM where the graphical structure is a directed acyclic graph (DAG), X_i are random discrete variables (called *nodes*) and the set of parameters $\boldsymbol{\theta}_\mathbf{s} = (\theta_{ijk})$, where k goes from 1 to r_i, j from 1 to q_i and i from 1 to n, represents 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 parents variables is in its jth-value. Finally, r_i is the number of different values of ith variable and q_i represents the number of possible instantiations for the set of parents of ith variable.

* Corresponding author. Address: Donostia International Physics Center, Donostia – San Sebastian, Paseo Manuel de Lardizabal 4, 20018 Guipuzcoa, Spain. Tel.: +34 943 018347; fax: +34 943 015600.
E-mail addresses: txomin.romero@ehu.es (T. Romero), pedro.larranaga@fi.upm.es (P. Larrañaga).

0888-613X/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2008.09.002

The BN paradigm can be used as a model to make inferences in domains with intrinsic uncertainty. The factorisation of the joint distribution that a BN represents allows an efficient reasoning inside the model. Introductions and classic textbooks about BN include [10,18] and [30]. As a model to make inferences, we will try to obtain their best triangulation, but we will also use them in the estimation of distribution algorithms themselves.

The most direct way to make inference in a BN is to compute the marginalization over the not instantiated variables. The number of terms involved in the marginalization grows exponentially with the number of variables. Lauritzen and Spiegelhalter show [26] that, instead of calculating each joint probability separately in order to add them in a posterior step, we can group the common factors of all of them in order to make the calculation more efficient. For example, for the Asia network (Fig. 1), we have:

$$\begin{aligned} P(A,X,D) &= \sum_{T,E,L,B,S} P(A,T,X,E,D,L,B,S) \\ &= P(A)\sum_{T} P(T|A)\left[\sum_{E} P(X|E)\left[\sum_{L} P(E|T,L)\left[\sum_{B} P(D|E,B)\left[\sum_{S} P(L|S)P(B|S)P(S)\right]\right]\right]\right] \end{aligned} \quad (1)$$

We can rewrite Eq. (1) as

$$P(A,X,D) = \textstyle\sum_{T,E,L,B,S}\psi(A)\psi(T,A)\psi(X,E)\psi(E,T,L)\psi(D,E,B)\psi(L,S)\psi(B,S)\psi(S) \quad (2)$$

Here, each factor is not a conditional probability, but a *potential function*. The potential functions are, at first, the conditional probabilities, and their parameters are the variables that are connected in the graph. An useful procedure to discover which variables are related to each potential function is to moralize the graph, that is, to add an arc between the parent nodes of each node (see Fig. 2). Nevertheless, when we sum over S in expression $\psi(L,S)\psi(B,S)\psi(S)$ for all the values of S, we obtain a new factor (a new auxiliary potential function) depending on L and B, and between these nodes there is no arc. This is a problem due to the prior definition of potential function. We can solve this problem with the *triangulation* of the graph (see [26] for a more extensive explanation). A graph is triangulated if it has no cycles with a length greater than three without a cord. In the Asia network, it is enough to add $L - B$ arc in order to triangulate it (see Fig. 3), but in more complex networks it is not so easy. A good triangulation could make it possible to obtain a solution to some problems related to graphs in polynomial time instead of exponential time [3].

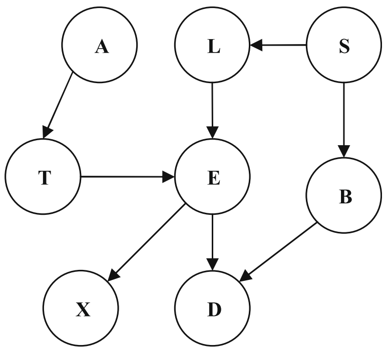


Fig. 1. The Asia Bayesian network.

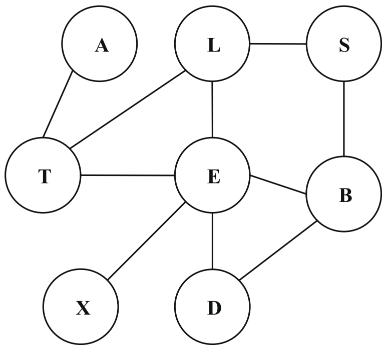


Fig. 2. The moralized Asia network.

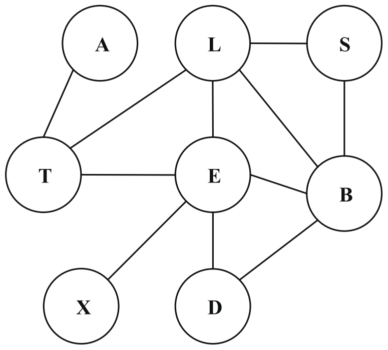


Fig. 3. A triangulation of Asia network.

In our experiments, the algorithm that searches for the best triangulation will use the known method of the vertex elimination. Basically, when a vertex is selected to be removed from the graph, we add all the arcs needed to make the subgraph of their adjacent nodes *complete*. Then, we remove the vertex and all the arcs related to it. The graph that incorporates all the new arcs to its set of original arcs is a triangulated graph. The sequence of removed vertices determines the efficiency of the resulting triangulated graph, given by the size of its cliques.[1] There are several criteria to evaluate the quality of a triangulation [34]. One of the most known criterions is the minimum weight, used in the experiments as the evaluation function of the EDAs. This method associates a weight, $w(C_i)$, to each clique:

$$w(C_i) = \sum_{X_j \in C_i} \log_2(r_j) \tag{3}$$

where r_j is the number of possible states of node X_j that belongs to clique C_i. So, we can define a weight of the triangulated graph:

$$w(\mathbf{G}^t) = \log_2 \sum_{C_i \in C} \prod_{X_j \in C_i} r_j \tag{4}$$

So, our aim is to minimize $w(\mathbf{G}^t)$.

Obtaining the best triangulation for a Bayesian network is an NP-hard problem [2,36]. In this work, we will use the univariate marginal distribution algorithm (UMDA) [28] and the mutual information maximization for input clustering (MIMIC) [12], both of them estimation of distribution algorithms (EDAs), to find the best possible triangulation. EDAs are a new metaheuristic approach belonging to evolutionary computation that are founded in probability theory.

The rest of the paper is organized as follows: in Section 2 we introduce the estimation of distribution algorithms and describe the individual representation. In Section 3 we present the recursive estimation of distribution algorithm (REDA) paradigm. In Section 4 we present the results of the EDA and REDA experiments. In Section 5 we review the early work on the search of the best triangulation, and Section 6 presents the concluding remarks.

2. Estimation of distribution algorithms

EDAs [23,25,27,29] are population-based stochastic heuristics that replace the classic crossover and mutation operators of genetic algorithms by learning the probability distribution from a database and its later simulation. Initially, a random set of individuals is generated. Each individual is evaluated using an objective function. The new population of individuals is sampled from the probability distribution estimated using a subset of individuals selected from the previous generation. Individuals with better function values have a higher chance to be in the subset of individuals selected. The process is iterated until some stop criterion is fulfilled. Thus we can capture all the relationships among the variables explicitly. EDAs have been applied to several problems such as in the graph matching [6], the partial abductive inference in BNs [13] or BNs structure learning [7,31,33].

2.1. General form of an EDA algorithm

We shall establish some definitions before showing the general pseudocode of an EDA:

- $\mathbf{Z} = (Z_1, \ldots, Z_m)$ represents the m-dimensional variable of m components and $\mathbf{z} = (z_1, \ldots, z_m)$ will denote an instantiation of the m-dimensional variable (an individual).

[1] A clique is a subset of nodes which is complete (i.e. there is an edge between every pair of nodes) and maximal.

- $D_l = \{\mathbf{z}^1, \ldots, \mathbf{z}^M\}$ 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 generalised probability distribution of $\mathbf{Z}$ at the lth generation, given D_{l-1}^s. The symbol ρ is used for both discrete and continuous variables.

The general form of the EDA algorithm proposed in this work for searching for the best triangulation can be seen in Fig. 4.

To estimate $\rho_l(\mathbf{Z})$ (the main problem) and then sample the new generation, EDAs construct and use a PGM from the pool of selected individuals D_{l-1}^s. If the variables that the individuals are made of are discrete, the PGM selected is a BN, and if they are continuous, the EDA constructs a gaussian network (GN) [35].

2.2. Individual representation

The EDAs, throughout their evolution, simulate different individuals of m components [$(z_1, z_2, \ldots, z_m)$, where z_i is an integer or real number] that have to be univocally associated with a specific ordering of the n variables. With continuous EDAs, the representation of the individuals represents no problem, because the components of the individual just need to be ordered and each component instantiation associated with its respective index to obtain a valid ordering. That is, in this case $m = n$. The disadvantage consists of the high redundancy of this representation because different continuous individuals can generate the same ordering. For instance, the continuous individual $(0.5, 1.6, 0.2, 0.1)$ will be associated with the ordering $(3-4-2-1)$, but so does the individual $(0.3, 2.0, 0.2, 0.0)$.

In the case of discrete domains, the direct individual representation could be a problem. If we have 4 variables and 4! possible permutations or orderings, we cannot use an individual of 4 components whose variables can take 4 instantiations at the most, because we could obtain, for example, the instantiation $(2-3-4-4)$, which is not a valid ordering. But there is a solution, first used in [33] by Romero et al. 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 way, shown in Table 1. The decomposition of 4! is $4 \cdot 3 \cdot 2 \cdot 1$. If we represent the individual with three components 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. But this representation also has some disadvantages: there are components with a very high number of possible states, and this makes the efficiency of the EDA worse. A partial solution, used in this paper, is the prime factor decomposition of $n!$ (for four variables, the prime factor decomposition $2 \cdot 2 \cdot 3 \cdot 2 \cdot 1$ determines an individual of four components). But if we have a high number of variables, the individual can be very long, and some components can continue having a very high number of possible states. In networks of, for instance, 50 variables, we find that 47 is a prime number: so we will have at least a component with 47 possible states.

```
EDA
  $D_0 \leftarrow$ Generate $M$ individuals randomly (the initial population)
  $l \leftarrow 1$
  Repeat
    For each individual $\mathbf{z}^i \in \mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^M$ do
      Calculate the score $s^i$ of the individual
      If $s_i$ is the best score found then $\mathbf{b}^i \leftarrow \mathbf{z}^i$ End if
    End for
    $D_{l-1}^s \leftarrow$ Select $N \leq M$ individuals from $D_{l-1}$ using a fixed selection
    method
    $\rho_l(\mathbf{Z}) \leftarrow$ Estimate the probability distribution of $\mathbf{Z}$ given $D_{l-1}^s$
    $D_l \leftarrow$ Sample $M$ individuals (the new population) from $\rho_l(\mathbf{Z})$
    $l \leftarrow l + 1$
  Until a stopping criterion is verified
  Return $\mathbf{b}^i$
```

Fig. 4. Main scheme of the EDA approach.

Table 1
Possible orderings of 4 nodes

1 – 1234	7 – 2134	13 – 3124	19 – 4123
2 – 1243	8 – 2143	14 – 3142	20 – 4132
3 – 1324	9 – 2314	15 – 3214	21 – 4213
4 – 1342	10 – 2341	16 – 3241	22 – 4231
5 – 1423	11 – 2413	17 – 3412	23 – 4312
6 – 1432	12 – 2431	18 – 3421	24 – 4321

2.3. The estimation of the probability distribution

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 of them will be used in discrete and continuous domains, and take into account only low-order dependencies. The UMDA algorithm, introduced by Mühlenbein [28], assumes marginal independence between the variables (the components of the individuals), i.e., the model used to estimate $p_l(\mathbf{z})$ in the case of the discrete domain is the simplest, using the marginal frequencies to get the probability distribution:

$$p_l(z_i) = \frac{\sum_{j=1}^{N} \delta_j(Z_i = z_i | D^s_{l-1})}{N} \tag{5}$$

where

$$\delta_j(Z_i = z_i | D^s_{l-1}) = \begin{cases} 1 & \text{if in the } j-\text{th case of } D^s_{l-1}, \quad Z_i = z_i, \\ 0 & \text{opposite case.} \end{cases} \tag{6}$$

In the case of the continuous domain, the factorization of the joint density function is composed of normal univariate distributions:

$$f_{\aleph}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \prod_{i=1}^{n} f_{\aleph}(x_i; \mu_i, \sigma_i^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2} \tag{7}$$

and the estimation of the parameters is performed using their maximum likelihood estimates:

$$\widehat{\theta^l_i} = (\widehat{\mu^l_i}, \widehat{\sigma^l_i}) = \left(\frac{1}{N}\sum_{r=1}^{N} x^l_{ir}, \sqrt{\frac{1}{N}\sum_{r=1}^{N}\left(x^l_{ir} - \frac{1}{N}\sum_{r=1}^{N} x^l_{ir}\right)^2} \right)$$

MIMIC or *Mutual Information Maximization for Input Clustering*, proposed by De Bonet et al. [12], takes dependencies into account, but only between pairs of variables. In the continuous domain, GNs (Gaussian Networks) are used. 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 divergence:

$$H^{\pi}_l(\mathbf{x}) = H_l(X_{i_n}) + \sum_{j=1}^{n-1} H_l(X_{i_j} | X_{i_{j+1}}) \tag{8}$$

where $H(X)$ is the Shannon entropy of X variable and $H(X|Y)$ is the conditional entropy of X given Y. We try to search for the permutation π^* that minimizes $H^{\pi}_l(x)$. The test of all the $n!$ permutations can be impossible. So, MIMIC searches for the best permutation selecting as X_{i_n} variable the one that has the minor estimated entropy (the estimation is calculated using D^s_l) and, in each step, selecting the variable whose conditional entropy related to the selected variable in the previous step is the minor one. For an explanation of the continuous case, see [25].

2.4. Sampling the new generation

In both the discrete and continuous domain, we will use the *probabilistic logic sampling* (PLS) method of Henrion [17] to sample the new generation. In this method, the instance of a variable is generated after all its parents have already been sampled, using the distribution $p(z_i|\pi_i)$. The variables must be ordered in ancestral ordering before the simulation is performed. In the case of GNs, normal density simulation is carried out [25].

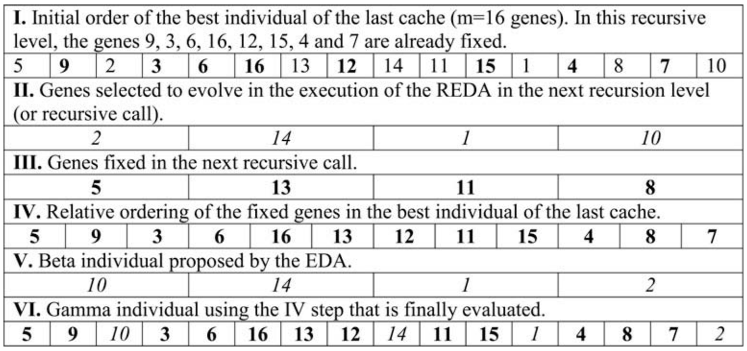
3. Recursive estimation of distribution algorithms (REDAs)

The main idea of the REDAs is to divide, in some part of the execution of the EDA algorithm, the set of nodes of the network (for which we want to obtain an optimal ordering) in two subsets. Then the new algorithm calls a REDA recursively for each subset, trying to obtain in each call an optimal ordering for it. So, in the first call, we will fix one subset of nodes and *evolve recursively* over the other, and in the second call we will do the opposite.

As we can see in [5], the most expensive steps of an EDA are the learning of the structure, the simulation of the new population and, sometimes, the evaluation of the individuals if our evaluation function is very complex. All these problems grow with the number of components of the individual. If we focalize the execution of the EDA in a subset of nodes instead of the whole one, we could obtain good general results in a shorter time, because the associated individual will be smaller too. So we can increase the amount of generations without increasing the execution time.

In order to present the recursive algorithm of the REDAs, we have to define some concepts:

- ζ^{α} will be a cache of orderings of m nodes. This cache is composed of the M best orderings found among the execution of the REDA (where M is the population size). The cache is ordered using the evaluation of the orderings. As we will see later, we will use this cache in the evaluation of the individuals and in the initialization of the population.



I. Initial order of the best individual of the last cache (m=16 genes). In this recursive level, the genes 9, 3, 6, 16, 12, 15, 4 and 7 are already fixed.

5	**9**	2	**3**	**6**	**16**	13	**12**	14	11	**15**	1	**4**	8	**7**	10

II. Genes selected to evolve in the execution of the REDA in the next recursion level (or recursive call).

2	*14*	*1*	*10*

III. Genes fixed in the next recursive call.

5	**13**	**11**	**8**

IV. Relative ordering of the fixed genes in the best individual of the last cache.

5	**9**	**3**	**6**	**16**	**13**	**12**	**11**	**15**	**4**	**8**	**7**

V. Beta individual proposed by the EDA.

10	*14*	*1*	*2*

VI. Gamma individual using the IV step that is finally evaluated.

5	**9**	*10*	**3**	**6**	**16**	**13**	**12**	*14*	**11**	**15**	*1*	**4**	**8**	**7**	*2*

Fig. 5. Construction of the i^{γ} using the i^{β} for $m = 16$ in the recursive level *1*, when $m_1 = 8$.

```
Procedure RecursiveEda ( $m_r$, $\mathbf{G}_r$ )
    If $m_r < 4$ then
        do nothing
    Else
        BaseEda ( $m_r$, $\mathbf{G}_r$ )
        Select randomly $trunc(\frac{m_r}{2})$ nodes of $\mathbf{G}_r$
        Be $\mathbf{G}_{r+1} \subset \mathbf{G}_r$ the set composed for these $trunc(\frac{m_r}{2})$ selected nodes
        $m_{r+1} \leftarrow trunc(\frac{m_r}{2})$
        RecursiveEda ( $m_{r+1}$, $\mathbf{G}_{r+1}$ )
        $\mathbf{G}'_{r+1} \leftarrow \mathbf{G}_r \cap \mathbf{G}_{r+1}$
        RecursiveEda ( $m_r - m_{r+1}$, $\mathbf{G}'_{r+1}$ )
    EndIf
    BaseEda ( $m_r$, $\mathbf{G}_r$ )
```

Fig. 6. Pseudocode of the REDA. Take into account that REDAs are more efficient than EDAs because both the number of individuals evaluated in *BaseEda* and the size of the individuals (in the majority of the recursive calls) are smaller.

- $\mathbf{G}$ is the set of m nodes of the Bayesian network we are trying to triangulate.
- $\mathbf{G}_r \subset \mathbf{G}$ is the subset of size m_r of $\mathbf{G}$ composed of the nodes that are not fixed in the recursion level number r.[2] The recursion level is the number of consecutive recursive calls. That is, at first, in the main code, the level is 0, and in the first recursive call the level will be 1, and so on.
- g_{m_r} is the number of components of the individual that represents an ordering of m_r nodes. Notice that, whereas in the continuous domain $m_r = g_{m_r}$, this is not true in the discrete domain.
- D_l is a population of individuals of size $g_{m_r} \leqslant g_m$ in the generation l, where r is the recursive level. We could use the D_l^r notation to incorporate the level of the recursion not only to the population, but also to the other variables described here. But it is not necessary to show this information in the description of the pseudocodes because the variables are local ones, and we prefer to simplify the notation.
- c_i^{α} is the ith ordering of m nodes that belongs to ζ^{α}. The c_0^{α} ordering will always be the best one found at each moment.
- c_i^{β} is an ordering of m_r nodes extracted from a c_i^{α}, and is composed of the nodes of c_i^{α} that are not fixed in the recursive level r, maintaining their relative ordering.
- i^{α} is an ordering of m nodes. c^{α} is an i^{α} that belongs to the cache.
- i_{ind}^{α} is the individual of g_m components associated with the ordering i^{α}.
- i^{β} is an ordering of $m_r \leqslant m$ nodes. c^{β} is an i^{β} extracted from a c^{α}.
- i_{ind}^{β} is the individual of g_{m_r} components associated with the ordering i^{β}.
- i is an ordering.
- $Node_j(i)$ is the jth node of the i ordering.
- i^{γ} is an ordering of m nodes created using an i^{β} ordering. We could construct it placing the fixed nodes of the recursive step r in their respective place and then placing the nodes that appear in the i^{β} ordering in the resulting gaps, maintaining their relative subordering.
- $Evaluate^{\alpha}(i^{\alpha})$ if a function that returns the metric of an ordering of size m.
- $Evaluate^{\beta}(i^{\beta}, i^{\gamma})$ is a procedure that constructs the i^{γ} ordering using the i^{β} one, and returns $Evaluate^{\alpha}(i^{\gamma})$, and the i^{γ} ordering itself.

[2] They are the nodes that will appear in the i^{β} orderings and participate in the evolution of the recursive call of the EDA.

```
Procedure InitializePopulationUsingCache ( $\zeta^{\alpha}_{old}$, $D$, $\mathbf{G}_r$, $i^{\alpha}_{best}$ )
    $i^{\alpha}_{best} \leftarrow c^{\alpha}_{old_0}$
    $\zeta^{\alpha} \leftarrow \varnothing$
    For each individual $c^{\alpha}_i \in \zeta^{\alpha}_{old}$ do
        If $i \leq 5$ then
            $s^{\alpha}_i \leftarrow Evaluate^{\alpha}(c^{\alpha}_i)$
            Insert $c^{\alpha}_i$ in $\zeta^{\alpha}$ using the value of $s^{\alpha}_i$
        EndIf
        $Evaluate^{\beta}(i^{\alpha}_{best}, c^{\beta}_i, i^{\gamma}, s^{\beta}_i)$
        Insert $i^{\gamma}$ in $\zeta^{\alpha}$ using the value of $s^{\beta}_i$
        if $\|\zeta^{\alpha}\| > M$, remove the worst (last) individual from $\zeta^{\alpha}$
        Add $c^{\beta}_{ind}$ to $D$
    EndFor
```

Fig. 7. Pseudocode to initialize the population (called from *BaseEda*).

```
Procedure Evaluate$^{\beta}$($i^{\alpha}_{best}$, $i^{\beta}$, $i^{\gamma}$, $s^{\beta}$)
    $k \leftarrow 1$
    For each position $j$ from 1 to $m$ do
        If $Node_j(i^{\alpha}_{best})$ is fixed then
            $Node_j(i^{\gamma}) \leftarrow Node_j(i^{\alpha}_{best})$
        Else
            $Node_j(i^{\gamma}) \leftarrow Node_k(i^{\beta})$
            $k \leftarrow k+1$
        EndIf
    EndFor
    $s^{\beta} \leftarrow Evaluate^{\alpha}(i^{\gamma})$
```

Fig. 8. Pseudocode of the evaluation of an i^{β}.

Table 2
Value of *N* for each REDA

M	MIMIC$_c$	UMDA$_c$	MIMIC$_d$	UMDA$_d$
100	90	24	24	50
500	120	370	370	120

We can only calculate the weight of an ordering if it has *m* nodes: so, we have to transform the i^{β} into an i^{γ} before calculating its metric. But, instead of placing the fixed components of the recursive step *r* in the same order all the time, we can use their relative ordering that appears in the best individual of the ζ^{α} cache. We can see an example of this construction in Fig. 5.

Figs. 6–8 show the pseudocode used. As we can see, the size of the subset of nodes that are not fixed determines the general case and the base case of the recursive algorithm. It must be taken into account that, after and before the recursive calls, we use a standard EDA (BaseEda) to evolve over the whole subset of nodes.

If in each recursive call we use a random initialization of the population, then we will not be able to take advantage of the previous good individuals (of the other recursive levels). But we can initialize the population by extracting the relative ordering of the non fixed nodes in the actual recursive level from all the orderings of the now old ζ^{α} cache (not exactly with the relative ordering but with the individual associated with it.[3]) In order to not to determine the evolution of REDA with the first random population too much, we can maintain the five best individuals of the old ζ^{α} in the new cache and fill the rest of the $M-5$ individuals with the best i^{γ} individuals proposed by the REDA in the actual recursive level.

4. The experiments

As we have seen in Section 1, we will use the known method of the vertex elimination in our experiments.

4.1. Networks used in the experiments

The networks used to compare EDAs and REDAs are named sparse and dense [21]. Sparse network is computer-generated and has 50 variables and 100 arcs. Dense, also computer-generated, has 50 variables and a greater number of arcs: 359.

[3] Remember that, in the case of the discrete domain, the individual cannot be obtained directly from the ordering, that is, in this case the relative ordering must be transformed into an individual before adding it to the population.

Table 3
Average results of 50 executions for sparse and EDAs

	MIMIC_c	UMDA_c	Best MIMIC_c	Best UMDA_c
$M = 100$	26.79 ± 2.35	26.80 ± 2.34	23.48	23.48
$M = 500$	27.55 ± 0.23	27.43 ± 0.33	26.65	26.26
	MIMIC_d	UMDA_d	Best MIMIC_d	Best UMDA_d
$M = 100$	27.61 ± 1.61	26.55 ± 0.90	24.20	24.38
$M = 500$	27.72 ± 0.63	23.35 ± 0.09	25.73	22.66

Table 4
Average results of 50 executions for dense and EDAs

	MIMIC_c	UMDA_c	Best MIMIC_c	Best UMDA_c
$M = 100$	58.31 ± 2.11	58.03 ± 2.08	55.04	54.01
$M = 500$	56.65 ± 2.53	56.50 ± 1.80	53.04	53.22
	MIMIC_d	UMDA_d	Best MIMIC_d	Best UMDA_d
$M = 100$	55.16 ± 2.80	56.10 ± 2.72	51.87	52.75
$M = 500$	54.27 ± 2.79	53.78 ± 1.86	51.85	51.81

Strictly speaking, sparse and dense are not Bayesian networks, because the structure in both of them is undirected acyclic graphs. This simplifies the process of the triangulation, because it is not necessary to moralize the network.

In addition, we have used two more complex networks named Medianus I and Medianus II [21] in order to compare the results of REDAs with other triangulation techniques in the case of real world networks. Medianus I is a Bayesian network describing relations between disorders, pathophysiological features and measurements for the human median nerve, and originates from the development of the MUNIN[4] system [1]. It has 43 nodes and 66 arcs. Medianus II is a modified version of Medianus I with 56 nodes and 161 arcs. In both networks, node values are between 3 and 21.

4.2. Choices about the parameters of EDAs and REDAs

- We always use a population size of $M = 100$ and $M = 500$ individuals for sparse and dense networks. We have executed the experiments for each EDA 50 times (continuous UMDA orUMDA_c, discrete UMDA or UMDA_d, continuous MIMIC or MIMIC_c and discrete MIMIC or MIMIC_d).
- In EDAs we print a sample (the best individual) every M different individuals evaluated. In REDAs, every $6 \cdot M$ different evaluations, because we can take advantage of the greater speed of the algorithm when the size of the subindividual is small.
- The EDA stops when we have printed 100 samples: there is not a convergence criterion because it does not guarantee the termination of the algorithm. The REDA, taking advantage of its execution speed, is executed five consecutive times over the successive caches. In each execution we print 50 samples. So, we stop when we have printed a total of 250 samples. Take into account that the printing of the 50 samples are divided between all recursive calls, making the execution of *Base-Eda* faster, because the number of iterations is very small and, in addition, the size of the individual is smaller in the majority of the recursive calls.
- In EDAs, we select the first $N = M/10$ best individuals from the actual population to generate the Bayesian network of EDA and then simulate the next generation. After a previous set of experiments, we selected the values that appear in Table 2 for REDAs.
- In EDAs, we only keep the five best individuals from one generation to the next, sampling $M - 5$ individuals at each step or generation. In REDAs, we sample the M individuals.

4.3. Results with EDAs for sparse and dense networks

In Table 3 we can see the average results of 50 executions for EDAs, in continuous (UMDA_c, MIMIC_c) and discrete (UMDA_d, MIMIC_d) domains, for sparse network. In Table 4 we have the same information for dense network.

The results for the triangulation using the standard EDAs are, in general, not good. Except in the case of UMDA_d, the best average result for sparse is 26.55, when the *worst* for the simulated annealing in [22] is 23.89. In [24], using genetic algorithms, the authors obtain average results <23.05 in 334 of 1296 parameter combinations used in the experiments. In the

[4] Knowledge based system for diagnosing diseases and malfuntions in the human neuromuscular system.

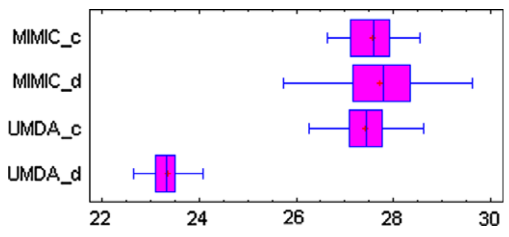


Fig. 9. Comparison for the sparse network, EDAs and $M = 500$.

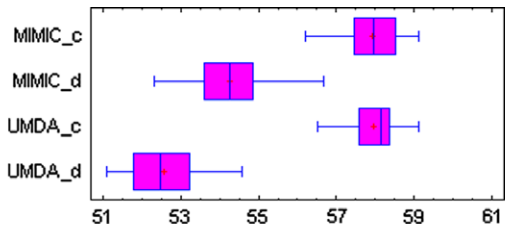


Fig. 10. Comparison for the dense network, EDAs and $M = 500$.

case of dense network, 55.16 is the best average result (without taking into account the case of UMDA$_d$), that is far from the worst result for the simulated annealing in [22] (54.25) or the 428 possible parameter combinations with an average weight lower than 52.88 for the genetic algorithms in [24]. However, the results of UMDA$_d$ are competitive with simulated annealing and genetic algorithms.

We can see a comparison between the different EDAs in Figs. 9 and 10 for $M = 500$ using a box-and-whisker plot. With sparse network and $M = 500$, UMDA$_d$ outperforms all the other discrete algorithms with a p-value near zero in the Wilcoxon rank-sum test. There are no statistical significant differences between the rest of the algorithms, except in the case of MIMIC$_d$ and UMDA$_c$ (p-value of 0.04). With dense, the result of comparisons is the same: UMDA$_d$ outperforms the rest of the algorithms with p-values $<10^{-9}$, and MIMIC$_d$ outperforms MIMIC$_c$ and UMDA$_c$ with a p-value near zero.

4.4. Results with REDAs for sparse and dense networks

The behaviour of REDAs is more competitive than those of EDAs. In Tables 5 and 6 we can see the average results of 50 executions for REDAs, in continuous and discrete domains respectively, for sparse and dense networks.

We can see a comparison between the different REDAs in Figs. 11 and 12 for $M = 500$ using a box-and-whisker plot. REDAs undoubtedly outperforms EDAs. In the case of UMDA$_d$, REDA outperforms the EDA's one with a p-value of 0.0003 in the case

Table 5
Average results of 50 executions for sparse and REDAs

	MIMIC$_c$	UMDA$_c$	Best MIMIC$_c$	Best UMDA$_c$
$M = 100$	23.28 ± 0.43	23.42 ± 0.35	22.67	22.63
$M = 500$	23.01 ± 0.09	23.20 ± 0.24	22.66	22.66
	MIMIC$_d$	UMDA$_d$	Best MIMIC$_d$	Best UMDA$_d$
$M = 100$	23.28 ± 0.17	23.23 ± 0.20	22.63	22.66
$M = 500$	23.30 ± 0.25	23.15 ± 0.22	22.61	22.64

Table 6
Average results of 50 executions for dense and REDAs

	MIMIC$_c$	UMDA$_c$	Best MIMIC$_c$	Best UMDA$_c$
$M = 100$	53.09 ± 1.13	53.04 ± 1.05	51.15	51.15
$M = 500$	52.73 ± 1.58	52.70 ± 1.44	50.88	50.88
	MIMIC$_d$	UMDA$_d$	Best MIMIC$_d$	Best UMDA$_d$
$M = 100$	53.04 ± 0.85	52.88 ± 1.19	50.88	50.99
$M = 500$	52.75 ± 0.73	52.62 ± 1.31	50.89	50.88

of sparse network, and there are no statistical significant differences with dense. In addition, independent of population size, kind of the algorithm (UMDA or MIMIC) or characteristics of the domain (continuous or discrete), REDAs seem to be more stable in their results. For sparse network and $M = 500$, there are only statistically significant differences between MIMIC$_c$ and MIMIC$_d$ (p-value = 0.00077) and MIMIC$_c$ and UMDA$_d$ (p-value = 0.033). But in the case of dense network, there are no significant differences between any of the algorithms. If we compare the differences between population sizes ($M = 100$ and $M = 500$), the behaviour is similar: there are significant statistical differences only in the case of MIMIC$_c$ (p-value 0.01). In other experiments, not presented in this paper, we have stated that standard EDAs are very sensible to the size of the set of selected individuals used to generate the Bayesian network, but we have not detected the same tendency in REDAs. In Fig. 13 we can see a comparison for a population of 100 individuals, sparse network and UMDA$_c$ algorithm between REDAs and EDAs of the average results of 10 executions, with several different sizes of the set. The results shows more stability in the case of REDAs.

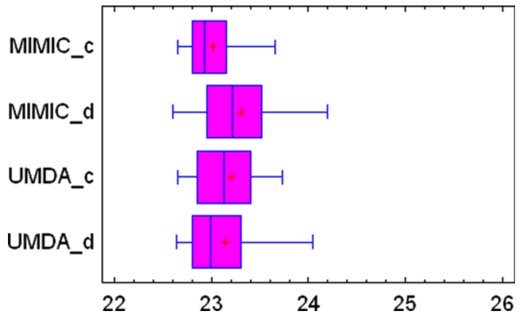


Fig. 11. Comparison for the sparse network, REDAs and $M = 500$.

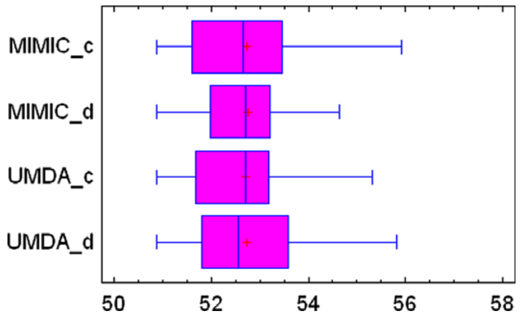


Fig. 12. Comparison for the dense network, REDAs and $M = 500$.

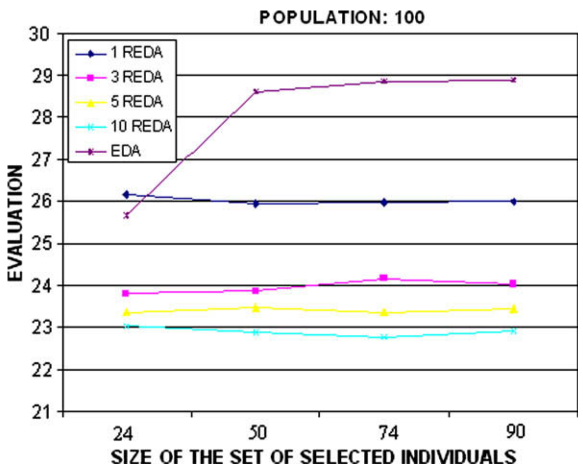


Fig. 13. Comparison for the sparse network and $M = 100$ of the average results of 10 executions of UMDA$_c$ for REDAs and EDAs with different sizes of the set of selected individuals. n REDA means n consecutive executions of the REDA over the successive caches. In the experiments of this paper, we have used $n = 5$.

Table 7
Average results of 50 executions for Medianus I, Medianus II and REDAs

	$UMDA_c$	$UMDA_d$	Best $UMDA_c$	Best $UMDA_d$
Medianus I and $M = 100$	23.36 ± 0.23	23.35 ± 0.21	22.63	22.63
Medianus I and $M = 500$	23.30 ± 0.19	23.22 ± 0.10	22.63	22.63
Medianus II and M = 100	21.78 ± 0.66	21.32 ± 0.63	20.54	20.46
Medianus II and M = 500	21.41 ± 0.45	21.16 ± 0.40	20.45	20.45

Table 8
Results of several triangulation techniques for Medianus I

	Average	Best
Random elimination	41.66	27.75
Lexicographic search	30.29	23.44
Maximum cardinality	29.00	23.63
Minimum size heuristic	23.27	22.71
Minimum weight heuristic	22.99	22.94
Minimum fill heuristic	23.14	22.71
Annealing (best control scheme)	22.71	22.63

Table 9
Results of several triangulation techniques for Medianus II

	Average	Best
Random elimination	46.35	29.28
Lexicographic search	30.59	22.28
Maximum cardinality	29.17	22.38
Minimum size heuristic	22.19	20.50
Minimum weight heuristic	20.65	20.65
Minimum fill heuristic	21.78	20.57
Annealing (best control scheme)	20.54	20.49

4.5. Comparison between REDAs and other triangulation methods with real world networks

We have used the discrete and continuous version of the UMDA algorithm (in the *BaseEda* call) to compare the results of the associated REDA with other triangulation techniques. We have applied the REDAs to the Medianus I and II networks. As we can state in the Tables 7–9, the REDAs are competitive with other triangulation techniques, not only in the case of the average evaluation, but also in the case of the best one.

5. Previous work in triangulation

The method related to minimize $w(\mathbf{G}^t)$ is not the only one proposed in the early work to best triangulate a Bayesian network. Robertson and Seymour [32] and Becker and Geiger [4] do not try to minimize the weight of the triangulated graph, but want to obtain the one that has a clique number less than $k + 1$, given a k, being that all the nodes of the graph are binary (the *treewidth* criterion). The clique number is the size of the bigger clique of the graph. Their algorithms assure an approximate solution to the best one by a constant.

Won et al. [37] have discovered a bijective relationship between a hypergraph and a relational database scheme. There is also a biunivocal relation between the set of triangulated graphs and the set of acyclic hypergraphs. So, the authors are able to use well known properties and methods associated with the database schemes in order to obtain the associated triangulated graph.

Bodlaender et al. [8] try to preprocess the original graph, making it smaller but without losing the possibility of extrapolating the results for the preprocessed graph to the original one. The reduction is managed by rules such as; if V is a simplicial vertex with a degree $d \leqslant 0$, then we can remove V and update the found maximum degree, where a vertex is simplicial if its neighbours conform a clique (and its degree is the size of this clique). The algorithm applies these rules until it can not go any further. But this algorithm does not take the different values of each vertex into account.

Darwiche and Hopkins [11] work on the balance of a d-tree, because it is easy to obtain a junction tree from a d-tree or an ordering for the removal of the nodes. A d-tree for a graph $\mathbf{G}$ is a complete binary tree where the leaves are the families of the graph and the nodes that are not leaves serve to divide the families of $\mathbf{G}$ into two branches.

In other strategies related to the direct vertex removing, Kjaerulff proposes three heuristics in [21]:

- H1 – Removes, in each step, the variable that produces the smaller maximal clique *Mindgree*.
- H2 – Removes the variable that minimizes the number of added arcs.
- H3 – Removes the variable that minimizes the whole weight of the triangulated graph.

None of the three algorithms can assure the three minimizations at the same time. H1 is the fastest one, but it can return a bad ordering if the graph is already triangulated [34]. Cano et al. [9] use modifications of H1 also taking into account the number of values of the variable to remove, or the increment of the greatest clique size. Gamez et al. [16] propose an ant colony optimization algorithm (ACO), a multi-agent bio-inspired algorithm. This method achieves good and faster solutions than other combinatorial optimization methods.

Kjaerulff tries in [21] to eliminate the unnecessary arcs added by a previous triangulation done by means of a heuristic, and uses an ordering of the arcs. In [22] the same author proposes a search for an ordering of the nodes using simulated annealing [20], and obtains good results but sacrifices execution speed. Wen also uses the simulated annealing in [36], but with two different methods of perturbation and more restrictive state transitions. Here, we also found a sacrifice of the speed for the optimization of the resulting ordering.

Larrañaga et al. [24] use the genetic algorithms to search in the space of orderings. They use a complete set of crossover and mutation operators. Normally, the winning combination contains the most expensive operators (in the sense of the execution speed).

Flores and Gámez [14] triangulate the network not over the complete graph but over subsets of it (Maximal Prime Subgraphs or MPS), working over a smaller space, as REDAs do. This technique is useful only if the network can be decomposed into more than one MPS. The same authors show in [15] a review related to BNs triangulation.

6. Concluding remarks

The EDA parametrization is easier than other evolutionary methods, such as Genetic Algorithms, but continues to be a difficult task that determines the results of a standard EDA. It is difficult to select a priori a specific EDA algorithm (UMDA or MIMIC in our case) for a specific problem (the triangulation of Bayesian networks in this paper). We can say the same thing about the type of domain of EDA individuals (continuous or discrete). But with REDAs we can obtain good results, improve the speed (because in the majority of the recursive calls, the size of the individual is smaller than in standard EDA) and make the execution of the EDA relative independent of the parametrization, the specific EDA algorithm and the type of the domain of the components.

Regarding future work, we plan to study the stability of REDAs with other problems, like the learning of a Bayesian structure from a database of cases or the partial abductive inference, and make them parallel in order to increase the speed of their execution.

Acknowledgements

This work has been partially supported by the following grants: TIN2005-03824 and Consolider Ingenio 2010-CSD2007-00018 projects (Spanish Ministry of Education and Science) and COMBIOMED network in computational biomedicine (Carlos III Health Institute).

References

[1] S. Andreassen, F.V. Jensen, S.K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A.R. Sørensen, A. Rosenfalck, F. Jensen, Munin. an emg assistant, Computer-aided surface needle electromyography, Expert Systems in Diagnosis, (1989).
[2] S. Arnborg, D.G. Corneil, A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM Journal of Algorithms and Discrete Methods 8 (1987) 277–284.
[3] S. Arnborg, J. Lagergren, D. Seese, Easy problems for tree-decomposable graphs, Journal of Algorithms 12 (2) (1991) 308–340.
[4] A. Becker, D. Geiger, A sufficiently fast algorithm for finding close to optimal junction trees, in: Morgan Kaufmann (Ed.), Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, 1996, pp. 81–89.
[5] E. Bengoetxea, Inexact graph matching using estimation of distribution algorithms. Ph.D Thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, December 2002.
[6] E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant, C. Boeres, Learning and simulation of Bayesian networks applied to inexact graph matching, Pattern Recognition 35 (12) (2002) 2867–2880.
[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 18 (2003) 205–220.
[8] H. Bodlaender, A. Koster, F. van den Eijkhof, L. van der Gaag, Preprocessing for triangulation of probabilistic networks, in: Morgan Kaufmann (Ed.), Uncertainty in Artificial Intelligence (UAI-01), 2001, pp. 32–39.
[9] A. Cano, S. Moral, Heuristic algorithms for the triangulation of graphs, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems, París, 1994, pp. 98–107.
[10] E. Castillo, J. Gutierrez, A. Hadi, Expert Systems and Probabilistic Network Models, Springer-Verlag, New York, 1997.
[11] A. Darwiche, M. Hopkins. Using recursive decomposition to construct elimination orders, jointrees, and dtrees, in: ECSQARU'01: Proceedings of the 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer-Verlag, London, UK, 2001, pp. 180–191.
[12] J.S. de Bonet, C.L. Isbell, P. Viola. MIMIC. Finding optima by estimating probability densities. in: Advances in Neural Information Processing Systems, vol. 9, 1997, pp. 424–430.
[13] L.M. de Campos, J.A. Gámez, S. Moral, Partial abductive inference in Bayesian belief networks using a genetic algorithm, Pattern Recognition Letters 20 (11–13) (1999) 1211–1217.

[14] M.J. Flores, J.A. Gámez, Triangulation of Bayesian networks by retriangulation, International Journal of Intelligent Systems 18 (2) (2003) 153–164.
[15] M.J. Flores, J.A. Gámez, Review on distinct methods and approaches to perform triangulation for Bayesian networks, Advances in Probabilistic Graphical Models, Studies in Fuzziness and Soft Computing, vol. 213, Springer-Verlag, 2007 (chapter A).
[16] J.A. Gámez, J.M. Puerta, Searching for the best elimination sequence in Bayesian networks by using ant colony based optimization, Pattern Recognition Letters 23 (1–3) (2002) 261–277.
[17] M. Henrion, Propagating uncertainty in Bayesian networks by probabilistic logic sampling, in: J.F. Lemmer, L.N. Kanal (Eds.), Uncertainty in Artificial Intelligence, 2, North-Holland, Amsterdam, 1988, pp. 16–149.
[18] F.V. Jensen, An Introduction to Bayesian Networks, University College of London, 1996.
[19] F.V. Jensen, S.L. Lauritzen, K.G. Olesen, Bayesian updating in causal probablistic networks by local computations, In Computational Statistics Quaterly 4 (1990) 269–282.
[20] S. Kirpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[21] U. Kjaerulff, Triangulation of graphs – algorithms giving small total state space, Technical Report R90-09, Department of Computer Science, University of Aalborg, March 1990.
[22] U. Kjaerulff, Optimal decomposition of probabilistic networks by simulated annealing, Statistics and Computing 2 (1992) 7–17.
[23] P. Larrañaga, R. Etxeberria, J.A. Lozano, J.M. Peña, Combinatorial optimization by learning and simulation of Bayesian networks, in: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, 2000, pp. 343–352.
[24] P. Larrañaga, C. Kuijpers, M. Poza, R. Murga, Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms, Statistics and Computing 7 (1) (1997) 19–34.
[25] P. Larrañaga, J.A. Lozano, Estimation of distribution algorithms, A New Tool for Evolutionary Computation, Kluwer Academic Publishers, 2001.
[26] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application on expert systems, Journal of the Royal Statistical Society B 50 (2) (1988) 157–224.
[27] J.A. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea, Towards a new evolutionary computation: advances on estimation of distribution algorithms, Studies in Fuzziness and Soft Computing, vol. 192, Springer-Verlag, 2006.
[28] H. Mühlenbein, The equation for response to selection and its use for prediction, Evolutionary Computation 5 (1998) 303–346.
[29] H. Mühlenbein, H. Voigt, Gene pool recombination in genetic algorithms, in: J.P. Kelly, I.H. Osman (Eds.), Metaheuristics: Theory and Applications, Kluwer Academic Publishers, 1996, pp. 53–62.
[30] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
[31] J.M. Peña, J.A. Lozano, P. Larrañaga, Unsupervised learning of Bayesian networks via estimation of distribution algorithms: an application to gene expression data clustering, In International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 12 (2004) 63–82.
[32] N. Robertson, P.D. Seymour, Graph minors XIII the disjoint paths problem, Journal of Combinatorial Theory, Series B 63 (1995) 65–110.
[33] T. Romero, P. Larrañaga, B. Sierra, Learning Bayesian networks in the space of orderings with estimation of distribution algorithms, International Journal of Pattern Recognition and Artificial Intelligence 18 (2004) 607–625.
[34] D.J. Rose, A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, Graph Theory and Computing (1973) 183–217.
[35] R.S. Shachter, C.R. Kenley, Gaussian influence diagrams, Management Science 35 (5) (1989) 527–550.
[36] W.X. Wen, Optimal decomposition of belief networks, in: Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI-90), Elsevier Science Inc., New York, NY, USA, 1990, pp. 209–224.
[37] S.K.M Wong, D. Wu, C.J. Butz. Triangulation of Bayesian networks: a relational database perspective, in: TSCTC'02: Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing, Springer-Verlag, London, UK, 2002, pp. 389–396.