ELSEVIER International Journal of Approximate Reasoning 27 (2001) 121–142

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

www.elsevier.com/locate/ijar

Accelerating chromosome evaluation for partial abductive inference in Bayesian networks by means of explanation set absorption

Luis M. de Campos [a], José A. Gámez [b,*], Serafín Moral [a]

[a] *Dpto. de Ciencias de la Computación e I.A., Universidad de Granada, 18071 Granada, Spain*
[b] *Dpto. de Informática., Universidad de Castilla-La Mancha, 02071 Albacete, Spain*

Received 1 September 2000; accepted 1 March 2001

Abstract

Partial abductive inference in Bayesian belief networks (BBNs) is intended as the process of generating the K most probable configurations for a set of unobserved variables (the explanation set). This problem is NP-hard and so exact computation is not always possible. In previous works genetic algorithms (GAs) have been used to solve the problem in an approximate way by using exact *probabilities propagation* as the evaluation function. However, although the translation of a partial abductive inference problem into a (set of) probabilities propagation problem(s) enlarges the class of solvable problems, it is not enough for large networks. In this paper we try to enlarge the class of solvable problems by reducing the size of the graphical structure in which probabilities propagation will be carried out. To achieve this reduction we present a method that yields a (forest of) clique tree(s) from which the variables of the explanation set have been removed, but in which configurations of these variables can be evaluated. Experimental results show a significant speedup of the evaluation function when propagation is performed over the obtained reduced graphical structure. © 2001 Elsevier Science Inc. All rights reserved.

* Corresponding author.
E-mail addresses: lci@decsai.ugr.es (L.M. de Campos), jgamez@info-ab.uclm.es (J.A. Gámez), smc@decsai.ugr.es (S. Moral).

Keywords: Bayesian networks; Abductive reasoning; Maximum a posteriori hypothesis; Probabilistic reasoning; Evidence absorption; Genetic algorithms

1. Introduction

Bayesian belief networks (BBNs) [19,13] are frequently used as the kernel of *Probabilistic Expert Systems*, because they provide an efficient representation of the joint probability distribution, and allow us to calculate probabilities by means of *local* computation, i.e., only relevant information is considered when a probability has to be calculated.

Although the most commonly used type of inference in BBNs is *probabilities* (*or evidence*) *propagation*, in this paper we are interested in another type of inference, known as *abductive reasoning* (also known as *diagnostic reasoning* because it is in the field of diagnostic where abductive reasoning has its most clear application [11]).

Abduction is defined as the process of generating a plausible explanation for a given set of observations or facts [20], and in the context of probabilistic reasoning, abductive inference corresponds to finding the maximum a posteriori probability state of the system variables, given some *evidence* (observed variables) [18]. The increasing attention received by abductive inference in BBNs over the last decade has yielded as a result the development of both exact and approximate solution methods. However, only some of these algorithms [10,17,21,22] will work for every network topology and search for the K most probable configurations and not only for the best one. The algorithms presented in [17,22] are exact and based on clique tree propagation. On the other hand, the algorithms presented in [10,21] are approximate and based on genetic algorithms (GA), which follows from the fact that abductive inference in BBNs can be viewed as a combinatorial optimization problem.

When we are interested in obtaining the K most probable configurations only for a subset of the network variables called *explanation set* [16], the problem is known as the *partial abductive inference* or the *maximum a posteriori hypothesis*. Although this problem seems to be more useful in practical applications (because we can select the relevant [1] variables as the explanation set) than *total* abductive inference, it has received much less attention. Recently, the authors have proposed to solve this problem by combining GAs (to guide the search) and clique tree propagation (as the evaluation/*fitness* function) [7,9].

Our goal in this paper is to improve the efficiency of the evaluation function proposed in [7] by reducing the size of the clique tree in which the propagation

[1] In this paper we suppose that the explanation set is known (provided by the user, by another algorithm, etc.).

is carried out. The rest of the paper is organized as follows: In Section 2 we briefly recall some concepts about BBNs and abductive inference. In Section 3 the process of compiling a network into a clique tree is studied. In Section 4 we revise the method of absorption of evidence introduced by Lauritzen and Spiegelhalter [15]. In Section 5 we discuss how to adapt it in order to absorb the explanation set, and in Section 6 we present an algorithm to accomplish the task of obtaining a reduced clique tree, that is, a clique tree in which the explanation set variables have been absorbed. In Section 7 the experiments carried out are described, and finally, in Section 8, we consider the conclusions.

2. BBNs and abductive inference

A BBN [13,19] is a directed acyclic graph (DAG), $\mathcal{G} = (X_{\mathcal{U}}, \mathcal{E})$, where each node represents a random variable, and the topology of the graph shows the (in)dependence relations among the variables. The quantitative part of the model is given by a probability distribution for each node conditioned to its parents. If $X_{\mathcal{U}} = \{X_1, \ldots, X_n\}$ is the set of variables in the network, then the joint probability can be calculated as:

$$P(X_{\mathcal{U}}) = \prod_{X_i \in X_{\mathcal{U}}} P(X_i | pa(X_i)), \tag{1}$$

where $pa(X_i)$ contains the parents of X_i in $\mathcal{G}$. Eq. (1) is known as the *chain-rule*. In the rest of the paper we will use $F_{X_i} = \{X_i\} \cup pa(X_i)$ to denote the family of a node/variable X_i, and $f_{X_i} = P(X_i|pa(X_i))$ to denote the probability family of variable X_i.

Abductive inference in BBNs, also known as the most probable explanation (MPE) problem, corresponds to finding the maximum a posteriori probability state of the network, given the observed variables (the evidence). In a more formal way: if X_{O} is the set of observed variables and X_{U} is the set of unobserved variables, we aim to obtain the configuration x^*_{U} of X_{U} such that:

$$x^*_{\mathrm{U}} = \arg\max_{x_{\mathrm{U}}} P(x_{\mathrm{U}}|x_{\mathrm{O}}), \tag{2}$$

where $X_{\mathrm{O}} = x_{\mathrm{O}}$ is the observed evidence. Usually, x^*_{U} is known as the MPE. It is well known that the MPE can be found using probabilities propagation methods but replacing summation by maximum in the marginalization operator (due to the distributive property of maximum with respect to multiplication) [6]. Therefore, the process of searching for the most probable explanation has the same complexity as probabilities propagation. However, in order to search for the K MPEs more complex methods have to be used; for example, in [17] clique tree propagation is combined with a divide and conquer algorithm that iteratively identifies the K MPEs.

In *partial* abductive inference, if we denote the explanation set by $X_{\rm E} \subset X_{\rm U}$, then the goal is to obtain the configuration $x^*_{\rm E}$ of $X_{\rm E}$ such that:

$$x^*_{\rm E} = \arg \max_{x_{\rm E}} P(x_{\rm E}|x_{\rm O}) = \arg \max_{x_{\rm E}} \sum_{x_R} P(x_{\rm E}, x_R|x_{\rm O}), \qquad (3)$$

where $X_R = X_{\rm U} \setminus X_{\rm E}$. In general, $x^*_{\rm E}$ is not equal to the projection of the configuration $x^*_{\rm U}$ over $X_{\rm E}$, so we need to obtain $x^*_{\rm E}$ directly (Eq. (3)).

The process of finding the configuration $x^*_{\rm E}$ is more complex than that of finding $x^*_{\rm U}$, because not all clique trees obtained from the original BBN are valid. In fact, because summation and maximum have to be used simultaneously and these operations do not show a commutative behavior, the variables of $X_{\rm E}$ must form a sub-tree of the complete tree. The problem of finding a *valid* clique tree for a given explanation set $X_{\rm E}$ is studied in [8]. From that study it can be concluded that the size [2] of the clique tree obtained for partial abductive inference starts growing (in general) in an exponential way with respect to the number of variables included in the explanation set, being the worst case when $X_{\rm E}$ contains around half the variables in the network. Later, it decreases being the size in the case in which $X_{\rm E}$ contains all the variables the same than when $X_{\rm E}$ contains a single variable. Therefore, the computer requirements (time and memory) necessary can be so high, that the problem becomes unsolvable by exact computation, even for medium-size networks. Thus, the use of approximate methods for partial abductive inference in BBNs is even more necessary than for total abductive inference. As it has been mentioned in the introduction, given that abductive inference in BBNs can be viewed as a combinatorial optimization problem, optimization metaheuristics in general, and GAs in particular represent a good choice.

The exercise of dealing with partial abductive inference using optimization methods could seem easier than that of dealing with total abductive inference, because the size of the search space in the partial case is considerably smaller than in the total abductive inference problem. However, this is not the case, because of the increasing complexity of the evaluation (*fitness*) function. In fact, as $P(x_{\rm U}|x_{\rm O})$ is proportional to $P(x_{\rm U}, x_{\rm O})$, this value can be used to rank the different configurations, and so in the total case the *chain rule* can be applied in order to evaluate a configuration [10,21]. Thus, in the total case, to evaluate a configuration $|X_{\mathscr{U}}|$ multiplications are carried out.

Nevertheless, in the partial case, as we must remove (by addition) the variables in X_R, the chain rule cannot be used to evaluate a configuration $x_{\rm E}$ because of the large number of necessary operations (additions and multiplications). For example, if we have a network with $|X_{\mathscr{U}}| = 50$, $|X_{\rm E}| = 15$, $|X_R| = 30$ and

[2] The size of a clique is obtained as the multiplication of the number of different cases that each clique variable can take. The size of a clique tree is obtained by summing the size of its cliques.

$|X_O| = 5$, the number of operations is bounded by 2^{30} additions and 50×2^{30} multiplications. Clearly, this is computationally intractable given the large number of evaluated individuals in this kind of algorithms. Because of this, in [7] the fitness $P(x_E, x_O)$ of a configuration x_E is computed by the process described below, where $\mathscr{T} = \{C_1, \ldots, C_t\}$ is a rooted clique tree, being C_1 the root.

1. Enter the evidence x_O in $\mathscr{T}$,
2. Enter (as evidence) the configuration x_E in $\mathscr{T}$,
3. Perform *CollectEvidence* from the root (C_1) (i.e., an upward propagation), and
4. $p(x_E, x_O)$ is equal to the sum of the potential stored in the root (C_1).

Therefore, to evaluate a configuration an exact propagation is carried out, or more correctly half propagation, because only the *upward* phase is performed and not the *downward* one (see [13] for details about clique tree propagation). Furthermore, for this propagation we can use a clique tree obtained without constraints and so its size is much smaller than the clique tree used for exact partial abductive inference. In addition, in [7] it is shown how the tree can be pruned in order to avoid the repetition of unnecessary computations when a new chromosome is being evaluated. An improvement to this algorithm can be found in [9], consisting of the use of specific genetic operators that allow us to take advantage of the calculations previously carried out when a new individual is being evaluated. In this way the need to perform a whole upward propagation is avoided, although more memory is needed.

3. Building a clique tree

Nowadays, the most frequently used propagation algorithms are based on the transformation (compilation) of the BBN in a secondary structure called *clique tree*, in which the calculations are carried out. Using clique tree propagation we do not constrain the topology of the network. To obtain a clique tree a two-stage process has to be carried out: (1) the cliques are determined, and (2) they are given a tree structure in which the *running intersection property* is verified. In this paper we assume that $\mathscr{G}$ is a connected graph, this assumption does not imply any loss of generality, because if $\mathscr{G}$ is a disconnected graph the process can be applied to each component in turn.

In order to obtain the cliques of a graph $\mathscr{G}$, the following steps have to be carried out (see [4] or [13] for a detailed description of the process):

1. Obtain the *moral* graph $\mathscr{G}^M$ by: (1) adding a link between every pair of nodes with a common child in $\mathscr{G}$, and (2) dropping the directionality of the links in $\mathscr{G}$. Fig. 1(b) shows the moral graph for the network in Fig. 1(a).
2. Obtain a *triangulated* graph $\mathscr{G}^T$. A graph is triangulated if there are no cycles of length 4 or more without a *chord* (link). If $\mathscr{G}^M$ is not already triangulated, it will be needed to be *filled in* (by adding links). Fig. 1(c) shows a

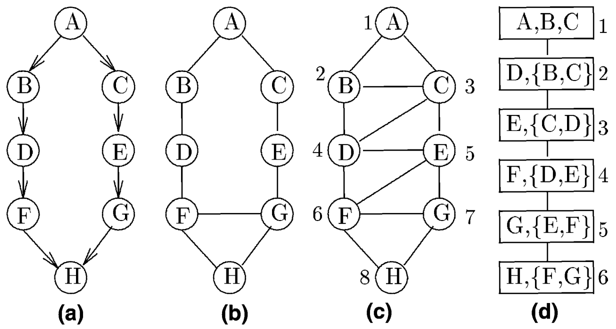

Fig. 1. (a) A belief network $\mathscr{G}$. (b) Moral graph $\mathscr{G}^{\mathrm{M}}$. (c) Triangulated graph $\mathscr{G}^{\mathrm{T}}$. (d) A rooted clique tree $\mathscr{T}$ for $\mathscr{G}$.

triangulation of $\mathscr{G}^{\mathrm{M}}$. The triangulation process is usually guided by a deletion sequence of the nodes in the graph. Different sequences can yield different triangulations, and this is a key point in the process, because clique tree size depends on the deletion sequence used during triangulation. Although searching for the best deletion sequence is an NP-hard problem, suitable heuristic algorithms have been proposed [2,14].

3. Give a label to each node of $\mathscr{G}^{\mathrm{T}}$ using the *maximum cardinality search* [23]. Fig. 1(c) shows a possible labeling for the triangulated graph.
4. Get the cliques (maximal fully connected sub-graphs) of $\mathscr{G}^{\mathrm{T}}$ and rank them according to the label of the nodes included in each clique.

Thus, after this process we obtain an ordered list of cliques $\mathscr{L} = \{C_1, C_2, \ldots, C_t\}$. In Fig. 1(d) we can see the cliques and their associated label.

The *running intersection property* says that is C_i and C_j are two cliques in the tree, then the variables in $C_i \cap C_j$ are contained in every clique along the path between C_i and C_j. To obtain a tree satisfying this property the following process is carried out:

- Let S_i denote the *separator* of clique C_i. Take S_1 as the empty set, and $S_i = C_i \cap (C_1 \cup \cdots \cup C_{i-1})$.
- Let R_i denote the *residual* of clique C_i, with $R_i = C_i \setminus S_i$.
- Set C_1 as the root of the tree, and for $i = 2, \ldots, t$ select as parent of C_i a clique C_j such that $S_i \subset C_j$.

Finally, a potential function $\psi()$ defined over the variables in a clique is associated to every clique in the tree. Each probability family f_{X_i} presented in the network is assigned to one (and only one) clique C_j, such that, $F_{X_i} \subseteq C_j$ (the existence of such a C_j is ensured by moralization). If there is a clique with an empty set of assigned families, it is given an unitary potential. For the remaining cliques, do $\psi(C_i) \leftarrow \prod_{f_{X_j} \in F(C_i)} f_{X_j}$, with $F(C_i)$ being the set of families assigned to C_i.

After this process, the clique tree represents a factorization of the joint probability distribution:

$$P(X_{\mathscr{U}}) = \prod_{C_j \in \mathscr{T}} \psi(C_j). \tag{4}$$

Fig. 1(d) shows a rooted clique tree, with C_1 as the root. Variables inside brackets represent the separator set, and variables outside brackets are the residual set.

Clique tree propagation efficiency depends on the clique tree size, which strongly depends on the largest clique in the tree. As the evaluation of a chromosome is carried out by means of clique tree propagation, by reducing the size of the clique tree we will improve the efficiency of the genetic algorithm (GA). In Section 6 we will present an algorithm that, by taking advantage of the fact that the explanation set is known a priori, allows us to obtain a *reduced* clique tree. Before that, we will revise Lauritzen and Spiegelhalter's absorption of evidence, because our algorithm is based on this idea.

4. Lauritzen and Spiegelhalter's absorption of evidence

Usually, evidence $X_O = x_O$ is entered to the clique tree using the following process (by $x^{\downarrow Y}$ we denote the configuration obtained from x after removing the literals corresponding to the variables not in Y):

- $\forall C_i \in \mathscr{T}$ such that $C_i \cap X_O \neq \emptyset$, $\forall$ configuration c_i of C_i do

$$\psi(c_i) = \begin{cases} \psi(c_i) & \text{if } c_i^{\downarrow C_i \cap X_O} = x_O^{\downarrow C_i \cap X_O}, \\ 0 & \text{in other case.} \end{cases}$$

In this way the clique tree structure is left unmodified and can be used for other problems (supersets of the current evidence, or even with different evidence sets by using *fast retraction of evidence* [5]). However, Lauritzen and Spiegelhalter propose in [15] to enter the evidence by a process that they define as *absorption of evidence*, which basically consists in absorbing evidence by conditioning the potential representation. The process is performed in two steps: (1) projection of the potentials involving evidence items onto a new reduced clique, [3] and (2) removal of non-maximal clusters (if any). This process can be seen in the next example:

[3] As a consequence of this process the clique tree can be split into several disconnected components. In this case we obtain a forest of trees.

Example 4.1. Consider the clique tree in Fig. 1(d) and let us assume that all the variables contained in it are propositional variables with two possible values, for example, $\{a, \overline{a}\}$ for variable A. Suppose that evidence $x_O = (A = a,\ G = g)$ has been found. Then, cliques C_1, C_5 and C_6 are modified by removing the evidence variables, and their potential functions are modified as follows:

$$\begin{aligned}
\psi^*(C_1) &= \psi^*(B,C) \leftarrow \psi(a,B,C),\\
\psi^*(C_5) &= \psi^*(E,F) \leftarrow \psi(E,F,g),\\
\psi^*(C_6) &= \psi^*(F,H) \leftarrow \psi(F,g,H).
\end{aligned}$$

Fig. 2(a) shows the result of this step. As we can see, cliques C_1 and C_5 are subsets of other cliques in the tree. If a clique C_i is a subset of a clique C_j, C_i can be removed by combining its potential with C_j. So, we obtain the clique tree in Fig. 2(b) as follows:

$$\begin{aligned}
\psi^*(C_2) &= \psi^*(B,C,D) \leftarrow \psi(B,C,D) \times \psi^*(B,C),\\
\psi^*(C_4) &= \psi^*(D,E,F) \leftarrow \psi(D,E,F) \times \psi^*(E,F).
\end{aligned}$$

Making $\psi^*(C_i) \leftarrow \psi(C_i)$ for unmodified potentials, in the initial clique tree we have

$$\begin{aligned}
P(B,C,D,E,F,H,a,g) = {} & \psi(B,C,a) \times \psi(B,C,D) \times \psi(C,D,E)\\
& \times \psi(D,E,F) \times \psi(E,F,g) \times \psi(F,H,g),
\end{aligned}$$

and in the reduced clique tree we have

$$\begin{aligned}
P(B,C,D,E,F,H,a,g) = {} & \psi^*(B,C,D) \times \psi^*(C,D,E) \times \psi^*(D,E,F)\\
& \times \psi^*(F,H).
\end{aligned}$$

We can now obtain $P(a,g)$ by means of an upward propagation in the reduced clique tree, whose size is 26, whereas the size of the initial clique tree is 48.

Therefore the propagation over this clique tree will be faster. Van der Gaag [24] has shown how the idea of the evidence absorption can improve the average-case computational cost of probabilistic inference.

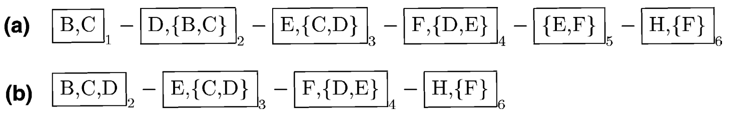

Fig. 2. (a) Join tree after absorbing A and G. (b) Join tree after incorporating non-maximal clusters.

5. Absorption of the explanation set

The following reflection can be made at this point: When the genetic algorithm is running and a new chromosome has to be evaluated, the chromosome represents the state taken by all the variables in the explanation set, i.e. we are trying to evaluate $p(x_{\mathrm{O}}, x_{\mathrm{E}})$. In this computation, x_{O} and x_{E} play a very similar role, with the only difference that x_{O} is fixed for a given problem and that x_{E} changes in different chromosomes. If we assume that the explanation set does not change from case to case, then we can consider that the variables in X_{E} are always evidence variables, and then try to obtain a reduced clique tree, in which the evaluation function will be faster.

Returning to our example, if $X_{\mathrm{E}} = \{A, G\}$, we can evaluate the chromosome (a, g) using the reduced clique tree obtained. However, we also need to be able to evaluate the remaining chromosomes, that is, $(a, \overline{g})$, $(\overline{a}, g)$, and $(\overline{a}, \overline{g})$. This is possible if we modify the previous method, as follows:

- Let $C_{i_{\mathrm{E}}}$ be the set of explanation set variables included in clique C_i ($C_{i_{\mathrm{E}}} = C_i \cap X_{\mathrm{E}}$). Now, when $C_{i_{\mathrm{E}}}$ is absorbed, we split the potential $\psi(C_i)$ in as many potentials as different configurations of $C_{i_{\mathrm{E}}}$ exist. That is, for each configuration $c_{i_{\mathrm{E}}}$ of $C_{i_{\mathrm{E}}}$, we make

$$\psi^*(C_i \setminus C_{i_{\mathrm{E}}})[c_{i_{\mathrm{E}}}] \leftarrow \psi(C_i \setminus C_{i_{\mathrm{E}}}, c_{i_{\mathrm{E}}}). \tag{5}$$

Thus we are now storing in $\psi^*(\,)$ an array of potentials rather than a single one.

- If some non-maximal cluster has to be removed by combination with its superset clique, we now have to perform a set of combinations instead of a single one, because the superset clique will contain an array of potentials.

In our example, we will have the following potentials after the absorption phase (only potentials involved in absorption are listed):

$$\psi^*(C_1) = \psi^*(B, C) = \begin{cases} \psi^*(B, C)[a] \leftarrow \psi(B, C, a), \\ \psi^*(B, C)[\bar{a}] \leftarrow \psi(B, C, \bar{a}), \end{cases}$$

$$\psi^*(C_5) = \psi^*(E, F) = \begin{cases} \psi^*(E, F)[g] \leftarrow \psi(E, F, g), \\ \psi^*(E, F)[\bar{g}] \leftarrow \psi(E, F, \bar{g}), \end{cases}$$

$$\psi^*(C_6) = \psi^*(F, H) = \begin{cases} \psi^*(F, H)[g] \leftarrow \psi(F, H, g), \\ \psi^*(F, H)[\bar{g}] \leftarrow \psi(F, H, \bar{g}). \end{cases}$$

The deletion of non-maximal clusters yields the following potentials for C_2 and C_4:

$$\psi^*(C_2) = \begin{cases} \psi^*(C_2)[a] \leftarrow \psi(C_2) \times \psi^*(C_1)[a], \\ \psi^*(C_2)[\bar{a}] \leftarrow \psi(C_2) \times \psi^*(C_1)[\bar{a}], \end{cases}$$

$$\psi^*(C_4) = \begin{cases} \psi^*(C_4)[g] \leftarrow \psi(C_4) \times \psi^*(C_5)[g], \\ \psi^*(C_4)[\bar{g}] \leftarrow \psi(C_4) \times \psi^*(C_5)[\bar{g}]. \end{cases}$$

Although some cliques have arrays of associated potentials, this is not a problem because a single potential will be considered in the course of propagation. The concrete potential to be used in a particular clique C_i is determined by the chromosome x_{E} being evaluated, i.e., we use $x_{\mathrm{E}}^{\downarrow C_{i\mathrm{E}}}$ as the index for the potential to be used in C_i. In our example, if we are evaluating $x_{\mathrm{E}} = (a, \bar{g})$, then $\psi(C_2)[a]$, $\psi(C_4)[\bar{g}]$ and $\psi(C_6)[\bar{g}]$ will be the potentials used in cliques C_2, C_4 and C_6 during the propagation.

6. Obtaining a reduced clique tree

In this section we propose a new algorithm in order to obtain a reduced clique tree. Instead of building a clique tree for all the variables in the BBN, and then reducing it, our idea is to obtain a clique tree containing only the variables in $X_{\mathcal{U}} \setminus X_{\mathrm{E}}$. In this way we hope that the size of the obtained clique tree would be smaller, because we start the triangulation process with a sparser graph than the initial one and the clique tree is specific for this explanation set. That is, instead of optimizing the clique tree size and then reducing it, we reduce the initial undirected graph and then we optimize the clique tree (or better clique tree forest) built from it.

Of course, the obtained cliques have to be enlarged with the variables in the explanation set, in order to obtain a potential representation of the joint probability distribution factorized by the original network. We will denote these new clusters of variables as C_i^e.

We now give the detailed version of the proposed algorithm, which we have called *get-reduced-forest* because in most occasions a forest of trees, and not a single tree is obtained. A detailed description of each step is given below.

Algorithm (*Get-reduced-forest* ($\mathcal{G}$, X_{E})).

Input: The BBN $G = (X_{\mathcal{U}}, \mathcal{E})$ and the explanation set X_{E}.
Output: The reduced forest for X_{E}.

1. Obtain the moral graph $\mathcal{G}^{\mathrm{M}}$.
2. Obtain $\mathcal{G}_p^{\mathrm{M}}$ as the projection [4] of $\mathcal{G}^{\mathrm{M}}$ over the variables in $X_{\mathcal{U}} \setminus X_{\mathrm{E}}$.

[4] The projection is carried out by deleting all the variables in X_{E} and all the edges $X_i - X_j$ such that $\{X_i, X_j\} \cap X_{\mathrm{E}} \neq \emptyset$. This produces a much simpler graph than the marginalization process in [3], but it is enough for a factorization of the joint probability given that the values of variables in the explanation set are known.

3. Obtain a forest $\mathscr{F}$ from $\mathscr{G}_p^{\mathrm{M}}$ by applying the process described in Section 3 to each connected component of $\mathscr{G}_p^{\mathrm{M}}$. Only graphical structure is obtained, not potential representation.
4. Let $\{C_1, \ldots, C_f\}$ be the set of cliques contained in $\mathscr{F}$.
5. For every C_i in $\mathscr{F}$ consider an enlarged cluster C_i^e which is initialized to C_i: $C_i^e \leftarrow C_i$.
6. Let $\mathscr{L} = \{F_{X_1}, \ldots, F_{X_n}\}$ be the list of families contained in the network. Attending to the variables of X_{E} contained in each family, we divide the list of families in three sub-lists:
   $\mathscr{L}_N = \{F_{X_i} | F_{X_i} \subseteq \{X_{\mathscr{U}} \setminus X_{\mathrm{E}}\}\}$,
   $\mathscr{L}_P = \{F_{X_i} | F_{X_i} \cap X_{\mathrm{E}} \neq \emptyset \wedge F_{X_i} \nsubseteq X_{\mathrm{E}}\}$,
   $\mathscr{L}_E = \{F_{X_i} | F_{X_i} \subseteq X_{\mathrm{E}}\}$.
7. For every $F_{X_i} \in \mathscr{L}_N$ assign f_{X_i} to an enlarged cluster C_j^e such that $F_{X_i} \subseteq C_j^e$.
8. For every $F_{X_i} \in \mathscr{L}_P$ assign f_{X_i} to an enlarged cluster C_j^e such that $F_{X_i} \cap \{X_{\mathscr{U}} \setminus X_{\mathrm{E}}\} \subseteq C_j^e$. Make $C_j^e \leftarrow C_j^e \cup F_{X_i}$.
9. Consider that $\mathscr{L}_{\mathrm{E}} = \{F_{X_{i_1}}, \ldots, F_{X_{i_m}}\}$.
   For every $F_{X_i} \in \mathscr{L}_{\mathrm{E}}$ if there is an enlarged cluster C_j^e such that $F_{X_i} \cap C_j^e \neq \emptyset$ then assign f_{X_i} to it. Make $C_j^e \leftarrow C_j^e \cup F_{X_i}$. In other case, [5] consider a new cluster $C_j \leftarrow \emptyset$ and the associated enlarged cluster $C_j^e \leftarrow F_{X_i}$. Assign f_{X_i} to it. This new cluster will be the root of a new tree of the forest.
10. For every C_i^e obtain $\psi(C_i^e)$.
11. For every C_i^e
    if $C_i^e = C_i$ make $\psi^*(C_i) \leftarrow \psi(C_i^e)$,
    if $C_i^e \neq C_i$ assign to $\psi^*(C_i)$ an array of potentials: $\psi^*(C_i)[c] \leftarrow \psi(C_i^e \wedge c)$, with as much potentials as different configurations c of $C_i^e \setminus C_i$ exists.
12. Return $\mathscr{F}$.

Now, we describe each step of the algorithm:
- *Steps* 1 and 2. As we want to obtain a reduced tree containing only variables not in X_{E}, we remove variables in X_{E} from the graph. However, we previously obtain the moral graph in order to maintain the original dependences. In Step 2 it is very usual to break the graph up into a set of disconnected components. As was stated in Section 3 this causes no inconvenience because we can treat each component separately, obtaining a forest and not a single clique tree. In fact, it is an advantage because propagation will be faster.
- *Steps* 3 and 4. The graphical structure for the variables not in the explanation set is obtained.
- *Step* 5. In order to build a potential representation of the joint probability distribution, all the families in the network have to be included in at least

[5] For example, if the initial graph is disconnected, and all the variables in a connected component belong to X_{E}.

one clique, so we need to enlarge the cliques with the variables in X_{E}. We begin by making a copy of the original cliques.

- *Step* 6. Depending on the variables contained in each family their treatment will be different, so we begin by classifying the families in three groups: families without variables in X_{E} ($\mathscr{L}_N$), families contained in X_{E} ($\mathscr{L}_{\mathrm{E}}$), and families with variables in X_{E} and in $X_{\mathscr{U}} \setminus X_{\mathrm{E}}$ ($\mathscr{L}_P$).
- *Step* 7. As families in $\mathscr{L}_N$ are contained in $X_{\mathscr{U}} \setminus X_{\mathrm{E}}$, moralization ensures the existence of at least one clique C_i (and so a cluster C_i^e) containing each family F_{X_j} in $\mathscr{L}_N$.
- *Step* 8. In this step we begin to enlarge the clusters with variables belonging to X_{E}. The idea is to find a cluster C_j^e containing all the variables in $F_{X_i} \cap (X_{\mathscr{U}} \setminus X_{\mathrm{E}})$, and enlarge it with the variables in $F_{X_i} \cap X_{\mathrm{E}}$. To ensure the existence of such a cluster, we have to prove that for all family $F_{X_i} \in \mathscr{L}_P$ at least one clique $C_j \in \mathscr{F}$ exists that is a superset of $(F_{X_i})^{\downarrow (X_{\mathscr{U}} \setminus X_{\mathrm{E}})}$. From the following sequence of operations we can see that this clique always exists:
  1. When the moral graph is obtained, variables in F_{X_i} constitute a fully connected sub-graph in $\mathscr{G}^{\mathrm{M}}$.
  2. When $\mathscr{G}_p^{\mathrm{M}}$ is obtained, variables in X_{E} and all their incident links are removed, but no link between variables in $(X_{\mathscr{U}} \setminus X_{\mathrm{E}})$ is removed. Thus, the variables in $F_{X_i} \cap (X_{\mathscr{U}} \setminus X_{\mathrm{E}})$ constitute a fully connected sub-graph and they will be included in the same clique C_j (because triangulation could add links but does not remove any link). It is clear that C_j is the clique we are looking for.

If for a family more than one cluster is found as a candidate to be enlarged, we select (heuristically) the smaller one.

- *Step* 9. The way we proceed in this step is similar to that in the previous one, but now we search for non-empty intersections between the variables belonging to X_{E} and not for the variables belonging to $X_{\mathscr{U}} \setminus X_{\mathrm{E}}$. In this step we can also have several candidate clusters to be enlarged for a given family. In this situation we operate as in the previous step, that is, enlarging the candidate of smallest size.
- *Step* 10. The potential associated with every enlarged cluster is obtained. At this point

  $$\prod_{i=1..t} \psi(C_i^e)$$

  is a factorization of the joint probability distribution stored in $\mathscr{G}$.
- *Step* 11. Finally, the array of potentials associated with every clique C_i is obtained. Notice, that for those cliques C_i such that $C_i = C_i^e$ the array will have dimension equal to 1.

We apply now the algorithm to our previous example. Fig. 3(a) shows the projection of the moral graph over the variables not in the explanation set

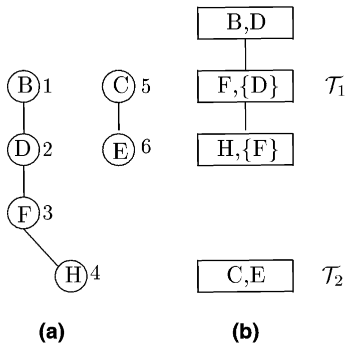

Fig. 3. (a) Projected moral graph $\mathcal{G}_p^{\mathrm{M}}$ (Note that this graph is also $\mathcal{G}_p^{\mathrm{T}}$). (b) A forest $\mathcal{F}$ for $\mathcal{G}_p^{\mathrm{T}}$.

$\{A, G\}$; as we can see, the resulting graph has two disconnected components, and so, a forest with two trees $\mathcal{F} = \{\mathcal{T}_1, \mathcal{T}_2\}$ is obtained (Fig. 3(b)).

Once we have obtained the graphical structure, we begin to process the list of families. Attending to the variables in X_{E} the following classification is obtained:

$$\mathcal{L}_N = \begin{cases} F_D = \{D, B\}, \\ F_{\mathrm{E}} = \{E, C\}, \\ F_F = \{F, D\}, \end{cases} \quad \mathcal{L}_P = \begin{cases} F_B = \{B, A\}, \\ F_C = \{C, A\}, \\ F_G = \{G, E\}, \\ F_H = \{H, F, G\}, \end{cases} \quad \mathcal{L}_{\mathrm{E}} = \{F_A = \{A\}.$$

The treatment of the families belonging to $\mathcal{L}_N$ yields the following initial assignment:

$$\{B, D\} \leftarrow f_D, \quad \{F, D\} \leftarrow f_F, \quad \{H, F\} \leftarrow 1, \quad \{C, E\} \leftarrow f_{\mathrm{E}}.$$

We begin now to enlarge the cliques by processing the families in $\mathcal{L}_P$:

- *Processing F_B*: The only candidate clique to be enlarged is $\{B, D\}$, so we assign the family to this clique and enlarge it with variable A. Fig. 4(a) shows the result.

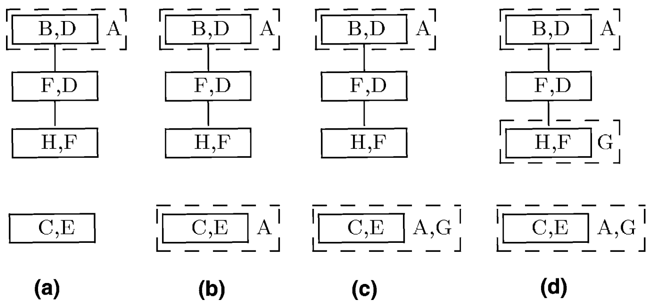

Fig. 4. Processing $\mathcal{L}_P$.

- *Processing F_C*: In this case we also have only one candidate to be enlarged: $\{C,E\}$. Fig. 4(b) shows the result.
- *Processing F_G*: The only candidate is $\{C,E,A\}$, so we enlarge it. Fig. 4(c) shows the result.
- *Processing F_H*: The only candidate is $\{F,H\}$, so we enlarge it. Fig. 4(d) shows the result.

As the treatment of F_A does not yield the enlargement of any cluster we can select randomly between the two candidates $\{B,D,A\}$ and $\{C,E,A,G\}$. If the first one is selected, the following assignment of families to enlarged clusters is obtained:

$$\begin{aligned}&\{B,D,A\} \leftarrow f_D \times f_B \times f_A, \quad &&\{F,D\} \leftarrow f_F,\\ &\{H,F,G\} \leftarrow f_H, &&\{C,E,A,G\} \leftarrow f_{\mathrm{E}} \times f_C \times f_G.\end{aligned}$$

Finally, we obtain the following array of potentials for the cliques in $\mathscr{F}$:

$$\begin{aligned}&\psi^*(B,D) \leftarrow \{\psi(B,D)[a],\ \psi(B,D)[\bar{a}]\},\\ &\psi^*(F,D) \leftarrow \psi(F,H),\\ &\psi^*(H,F) \leftarrow \{\psi(H,F)[g],\ \psi(H,F)[\bar{g}]\},\\ &\psi^*(C,E) \leftarrow \{\psi(C,E)[a,g],\ \psi(C,E)[a,\bar{g}],\ \psi(C,E)[\bar{a},g],\ \psi(C,E)[\bar{a},\bar{g}]\}.\end{aligned}$$

After this process we have obtained a (forest of) clique tree(s), defined over $X_{\mathscr{U}} \setminus X_{\mathrm{E}}$. However, the potentials associated with the cliques in that graphical structure do not represent a factorization of $P(X_{\mathscr{U}} \setminus X_{\mathrm{E}})$, but a factorization of $P(X_{\mathscr{U}} \setminus X_{\mathrm{E}},\ x_{\mathrm{E}})$ for all configuration x_{E} of X_{E}, by selecting from the array of potentials associated with each clique the potential indexed by x_{E}.

Therefore, when we want to evaluate a configuration x_{E} we work as if x_{E} had been entered as evidence. It is well known that if the clique tree factorizes over $P(X_{\mathscr{U}} \setminus X_{\mathrm{E}},\ x_{\mathrm{E}})$, and a *collect evidence* phase is called on some clique C_i (usually the root), then the normalization factor f can be obtained by the following marginalization:

$$f = \sum_{C_i} \psi(C_i),$$

and this normalization factor coincides with the probability of the instantiated evidence, that is, $f = P(x_{\mathrm{E}})$. If some observed evidence $X_{\mathrm{O}} = x_{\mathrm{O}}$ has been entered, then $f = P(x_{\mathrm{E}}, x_{\mathrm{O}})$. This is the basis for the evaluation function proposed in [7] and described in Section 2.

If we have a forest $\mathscr{F}$ of clique trees $\{\mathscr{T}_1, \ldots, \mathscr{T}_n\}$, then we can operate separately over each tree, avoiding the computational cost of passing messages that serve to rescale the clique potentials [12]. In this case, due to the independence relations that allow us to work with a forest of clique trees, we can obtain the normalization factors $f_1, \ldots, f_n$ by applying the previous procedure over each clique tree, and then to obtain the whole normalization factor, that is, $P(x_E)$ or $P(x_E, x_O)$ as

$$f = \prod_{i=1}^{n} f_i.$$

If we compare the result obtained when we apply Lauritzen and Spiegelhalter absorption of the explanation set (Fig. 2(b)) with that obtained by our algorithm (Fig. 3(b)), we can see that our expectations have materialized, because of the smaller size of the structure obtained by our method. Furthermore, we have obtained a forest and not a single tree, so that if a multiprocessor computer is available, propagation over each tree could be carried out in a different processor, and the gain would be greater.

7. Experimental evaluation

In this section we compare the efficiency of evaluating a chromosome by using the reduced forest with respect to the evaluation functions proposed in previous works [7,9]. We have carried out different experiments using four networks. The first one is the *Alarm* network [1], a real network that has been very commonly used in specialized literature. As this network can be considered to be *small*, we have generated three random networks with 50, 75 and 100 variables. The number of cases for each variable is 2. Table 1 shows some information about the four networks.

As the efficiency of our evaluation function depends on the size of the clique tree in which the propagation is carried out, we have performed the following experiment over the four networks:

Experiment 1. First, we obtain a clique tree for the network. The size of this clique tree and the size of its largest clique will be used for comparisons with the reduced trees obtained when an explanation set is considered. Second, we have randomly generated 50 explanation sets for each considered size. For each explanation set a reduced forest is obtained by applying our algorithm, storing the size of the whole forest, the size of the largest tree in the forest, and the size

Table 1
Some data about the networks used in the experiments

Network	Nodes	Links	Node potential size		
			Min	Max	Mean
Alarm	37	46	2	108	20.3
Random50	50	116	2	256	22.16
Random75	75	155	2	128	12.98
Random100	100	185	2	128	11.82

of the largest clique in the forest. We also compute the amount of memory necessary to store the different structures, assuming that 4 bytes are used for a real number. Therefore, for the clique tree memory requirements are obtained as clique tree size $\times 4$. However, the memory requirements for a forest are calculated using the size of the array of potentials, that is, the enlarged cluster size and not the clique size is used. The obtained results (averaged over the 50 different explanation sets) are presented in Tables 2–5.

Experiment 2. The aim of this experiment is to observe the impact of replacing previous evaluation functions in the GA by that proposed in this paper. We have performed the comparison in particular for the three following GAs:

Table 2
Experiments with the *Alarm* network (average over 50 runs)

	Forest size	Max clique tree size	Max clique size	Memory size (Kb)
$\|X_E\| = 4$	771.64	763.64	102.36	6.3
$\|X_E\| = 8$	511.18	467.52	84.00	10.4
$\|X_E\| = 12$	303.14	254.12	57.00	18.6
$\|X_E\| = 16$	198.92	155.56	43.12	47.7
Clique tree	1014.0		108.00	4.0

Table 3
Experiments with the *Random50* network (average over 50 runs)

	Forest size	Max clique tree size	Max clique size	Memory size (Mb)
$\|X_E\| = 5$	61,542.04	61,540.32	24,657.92	0.80
$\|X_E\| = 10$	13,496.16	13,492.48	4623.36	0.94
$\|X_E\| = 15$	3444.12	3437.92	1212.16	1.75
$\|X_E\| = 20$	699.40	689.04	201.92	3.84
$\|X_E\| = 25$	184.44	168.24	40.80	6.84
Clique tree	226,292.0		65,536.00	0.86

Table 4
Experiments with the *Random75* network (average over 50 runs)

	Forest size	Max clique tree size	Max clique size	Memory size (Mb)
$\|X_E\| = 7$	501,471.04	501,467.92	265,420.92	14.54
$\|X_E\| = 14$	71,897.40	71,890.48	30,924.80	18.63
$\|X_E\| = 21$	10,288.80	10,279.28	3287.04	12.25
$\|X_E\| = 28$	1612.04	1598.32	339.20	18.77
$\|X_E\| = 35$	411.76	394.24	60.96	20.24
Clique tree	2,477,312.0		524,288.00	9.45

Table 5
Experiments with the *Random100* network (average over 50 runs)

	Forest size	Max clique tree size	Max clique size	Memory size (Mb)
$\lvert X_E \rvert = 10$	202,609.00	202,602.56	85,852.16	6.17
$\lvert X_E \rvert = 20$	33,804.32	33,791.76	11,991.04	13.93
$\lvert X_E \rvert = 30$	4828.92	4810.48	1367.04	17.68
$\lvert X_E \rvert = 40$	930.56	902.64	159.68	20.47
$\lvert X_E \rvert = 50$	288.08	238.56	28.96	36.79
Clique tree		1,296,524.0	524,288.0	4.94

GA1. An upward propagation over the clique tree is used as the evaluation function [7].

GA2. The first population is evaluated as in the previous point, but for the remaining of populations, only partial upward propagation is required, because of the problem-specific genetic operators used in the algorithm [9].

GA3. An upward propagation over the reduced forest obtained by applying the algorithm presented in Section 6 is used as evaluation function.

The problem-specific genetic operators introduced in [9] cannot be directly combined with the use of a reduced forest, because these operators are based on the presence of explanation set variables in cliques.

In this paper we do not show results about the accuracy of GA3. This is due to the fact that GA3 has the same accuracy as GA1 (propagation over a clique tree defined over $X_{\mathcal{U}}$) [7], because both methods implement exact propagation. Experiments showing the *good* behavior of GA1 (and so of GA3) can be found in [7]. A detailed comparison between the accuracy of GA1 and GA2 can be found in [9].

The number of generations and the population size are the same for the three GAs, and these values have been increased with the number of variables in the

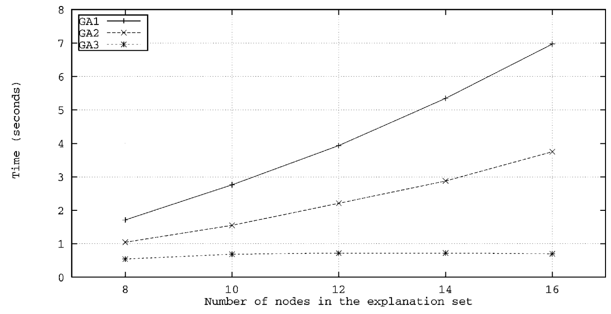

Fig. 5. Time comparison for the *Alarm* network using the different alternatives of chromosome evaluation.

Table 6
One chromosome evaluation CPU time for the *random50* network (average over 10 runs)

	CPU Time (s)	Speedup
$\lvert X_E \rvert = 10$	0.00141	×3156
$\lvert X_E \rvert = 15$	0.00033	×13,484
$\lvert X_E \rvert = 20$	0.00006	×74,166
Clique tree	4.45	

Table 7
One chromosome evaluation CPU time for the *random75* network (average over 10 runs)

	CPU Time (s)	Speedup
$\lvert X_E \rvert = 7$	0.05270	×884
$\lvert X_E \rvert = 14$	0.00594	×7845
$\lvert X_E \rvert = 21$	0.00077	×60,519
$\lvert X_E \rvert = 28$	0.00018	×258,888
Clique Tree	46.6	

Table 8
One chromosome evaluation CPU time for the *random100* network (average over 10 runs)

	CPU Time (s)	Speedup
$\lvert X_E \rvert = 10$	0.04127	×969
$\lvert X_E \rvert = 20$	0.00591	×6768
$\lvert X_E \rvert = 30$	0.00083	×48,192
$\lvert X_E \rvert = 40$	0.00016	×250,000
Clique tree	40.0	

explanation set. Fig. 5 shows a plot of the CPU time used by the GAs for the Alarm network.

For the other networks we show (Tables 6–8) the CPU time required to evaluate *one* chromosome using propagation over the clique tree and propagation over a reduced forest. This experiment cannot be performed for the evaluation function used in GA2 [9] because this evaluation function is based on the information stored when the chromosomes were evaluated. In any case, we can see from Fig. 5 (and also from experiments performed in [9]) that this evaluation function requires, roughly speaking, between 50% and 70% of the time required by propagation over the clique tree (GA1).

In these experiments the results are averaged over 10 different runs, that is, 10 different explanation sets for each size.

7.1. Experimental conclusions

After analyzing the results we are in a position to draw the following conclusions:

- The size of the graphical structure (to be used during propagation) obtained with the method proposed in this work is significantly smaller than the size of a clique tree obtained for all the variables in the network.
- The size of the obtained reduced forest decreases (in general) with the number of variables in the explanation set. This is due to the following: (1) if we have fewer variables the probability tables will be smaller, and (2) the removal of links yields a reduction in the number of variables included in a clique, so its associated potential will have a smaller size.
- The amount of memory necessary to store the whole structure, that is, to store the array of potentials is significantly greater than that needed to store the clique tree.
- The amount of memory necessary to store the array of potentials increases with the number of variables in the explanation set. This is due to the way these variables are assigned to the enlarged clusters.
- The use of this new approach to evaluate the chromosomes yields a significant time gain even for small networks.
- In the execution of a GA, hundreds or thousands of chromosomes have to be evaluated, so the amount of time needed to perform an evaluation should be very small. From the experiments performed in this paper, we can deduce that for large networks the evaluation functions proposed in [7,9] are unsuitable because of the amount of time required for propagation. However, we can see that propagation over a reduced forest can be successfully applied.

8. Conclusions

The problem of performing partial abductive inference in BBNs has been studied. This type of probabilistic reasoning has the disadvantage that depending on the selected explanation set, exact inference may be unfeasible even for networks in which exact inference is feasible for other types of probabilistic reasoning (such as probabilities propagation or *total* abductive inference).

In previous attempts to find an approximate solution to this problem [7,9], the evaluation function was based on an exact probabilities propagation over a clique tree. This type of approach requires assuming the network to be such that an exact probabilistic propagation is feasible. However, for complex problems this assumption falls short and requires redefining as follows: *we assume that the network is such that an exact probabilistic propagation is feasible and can be performed swiftly*, because hundreds or thousands of propagations have to be carried out during the execution of the GA.

In this paper we have presented an algorithm that by using our a priori knowledge (the explanation set), allows us to obtain a forest of trees with a size considerably smaller than the original clique tree defined over all the variables in the network. By using this forest for propagation we have significantly improved the evaluation function (up to 250,000 $\times$ speedups in some cases).

The disadvantage involved in using this new structure to evaluate a chromosome is the greater demand upon computer memory requirements. This might seem to contradict the foregoing, where we argue that the reduced forest has a size considerably smaller than the initial clique tree, but the contradiction is only apparent. In the computer memory we have to store an array of potentials for each clique in the forest, which needs as much memory as the enlarged cliques obtained by the algorithm. However, when propagation is carried out a single potential is considered for each clique, so that its size is much smaller. The concrete potential used for each clique is determined *on line* using the chromosome being evaluated as the index for the different arrays.

We are also aware that the obtained reduced forest is only valid for a specific problem, i.e., is only valid for the explanation set used to reduce the clique tree. However, it is clear that this particular solution performs better than the general one (clique tree).

Another advantage stemming from the approach taken in this paper is the ease with which a certain parallelism level may be implemented. In fact, coarse grain parallelism can be implemented by distributing the different trees included in the graph among the available processors.

In order to reduce the amount of memory needed by the algorithm, in future work we plan to investigate the following *potential* solutions:

- Using the method proposed by Lauritzen and Spiegelhalter but without performing the step in which non-maximal clusters are absorbed. In this case the amount of memory necessary will be smaller than that required to store the clique tree defined over all the variables. However, the structure used to perform the propagation will have a greater size than the reduced forest obtained when non-maximal clusters are absorbed, with the result that propagation will be slower. It is, of course, the case that the strategy of absorbing only some non-maximal clusters can be used, by trying to achieve an equilibrium between the size of the structure used for propagation and the amount of memory necessary to store the array of potentials.
- Modifying our algorithm by allowing the presence of clusters that are not cliques in the *junction tree*. This case is similar to the previous one.
- Modifying the step in which families of $\mathscr{L}_{\mathrm{E}}$ are assigned, by allowing the repetition of some variables. In this way we expect to find some smaller-sized arrays.

For the future we also plan to investigate the adaptation of the specific genetic operators designed in [9] to this reduced graphical structure and the use of the method proposed above to evaluate individuals in other schemes, such as *simulated annealing* or *tabu search*.

Acknowledgements

This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) under Project TIC97-1135-CO4-01.

References

[1] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, G.F. Cooper, The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, in: Proceedings of the Second European Conference on Artificial Intelligence in Medicine, Springer, Berlin, 1989, pp. 247–256.

[2] A. Cano, S. Moral, Heuristic algorithms for the triangulation of graphs, in: Proceedings of the 5th International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU), vol. 1, Paris (France), 1994, pp. 166–171.

[3] E. Castillo, J. Ferrándiz, P. Sanmartín, Marginalizing in undirected graph and hypergraph models, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98), Morgan Kaufmann, San Mateo, 1998, pp. 69–78.

[4] E. Castillo, J.M. Gutiérrez, A.S. Hadi, Expert systems and probabilistic network models, in: Monographs in Computer Science, Springer, New York, 1997.

[5] R.G. Cowell, A.P. Dawid, Fast retraction of evidence in a probabilistic expert system, Statistics and Computing 2 (1992) 37–40.

[6] A.P. Dawid, Applications of a general propagation algorithm for probabilistic expert systems, Statistics and Computing 2 (1992) 25–36.

[7] 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.

[8] L.M. de Campos, J.A. Gámez, S. Moral, On the problem of performing exact partial abductive inference in Bayesian belief networks using junction trees, in: Proceedings of the 8th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00), 2000, pp. 1270–1277.

[9] L.M. de Campos, J.A. Gámez, S. Moral, Partial abductive inference in Bayesian belief networks: An evolutionary computation approach by using problem specific genetic operators, Technical Report #DECSAI-00-02-02, Dpto. de Ciencias de la Computación e IA, Universidad de Granada, 2000.

[10] E.S. Gelsema, Abductive reasoning in Bayesian belief networks using a genetic algorithm, Pattern Recognition Letters 16 (1995) 865–871.

[11] E.S. Gelsema, Diagnostic reasoning based on a genetic algorithm operating in a Bayesian belief network, Pattern Recognition Letters 17 (1996) 1047–1055.

[12] C. Huang, A. Darwiche, Inferencie in belief networks: A procedural guide, International Journal of Approximate Reasoning 15 (1996) 225–263.

[13] F.V. Jensen, An introduction to Bayesian Networks, UCL Press, 1996.

[14] U. Kjærulff, Triangulation of graphs – algorithms giving small total space, Technical Report R 90-09, Department of Mathematics and Computer Science, Institute of Electronic Systems, Aalborg University, March 1990.

[15] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, Journal of Royal Statistics Society B 50 (2) (1988) 157–224.

[16] R.E. Neapolitan, Probabilistic Reasoning in Expert Systems. Theory and Algorithms, Wiley/Interscience, New York, 1990.

[17] D. Nilsson, An efficient algorithm for finding the M most probable configurations in Bayesian networks, Statistics and Computing 8 (1998) 159–173.
[18] J. Pearl, Distributed revision of composite beliefs, Artificial Intelligence 33 (1987) 173–215.
[19] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Mateo, 1988.
[20] H.E. Pople, On the mechanization of abductive logic, in: Proceedings of the 3rd International Joint Conference on Artificial Intelligence, 1973, pp. 147–152.
[21] C. Rojas-Guzman, M.A. Kramer, Galgo: A genetic algorithm decision support tool for complex uncertain systems modeled with Bayesian belief networks, in: Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence, Morgan and Kauffmann, San Mateo, 1993, pp. 368–375.
[22] B. Seroussi, J.L. Goldmard, An algorithm directly finding the K most probable configurations in Bayesian networks, International Journal of Approximate Reasoning 11 (1994) 205–233.
[23] R.E. Tarjan, M. Yannakakis, Simple linear-time algorithms to test chordality of grapstext acyclicity of hypergraphs and selectively reduce acyclic hypergraphs, SIAM Journal Computing 13 (1984) 566–579.
[24] L.C. van der Gaag, On evidence absorption for belief networks, International Journal of Approximate Reasoning. 15 (1996) 265–286.