Chapter 16

Partial Abductive Inference in Bayesian Networks: An Empirical Comparison Between GAs and EDAs

L.M. de Campos
Department of Computer Science and Artificial Intelligence
University of Granada
lci@decsai.ugr.es

J.A. Gámez
Department of Computer Science
University of Castilla-La Mancha
jgamez@info-ab.uclm.es

P. Larrañaga
Department of Computer Science and Artificial Intelligence
University of the Basque Country
ccplamup@si.ehu.es

S. Moral
Department of Computer Science and Artificial Intelligence
University of Granada
smc@decsai.ugr.es

T. Romero
Department of Computer Science and Artificial Intelligence
University of the Basque Country
scsroast@sc.ehu.es

P. Larrañaga et al. (eds.), *Estimation of Distribution Algorithms*
© Springer Science+Business Media New York 2002

Abstract Partial abductive inference in Bayesian networks is intended as the process of generating the K most probable configurations for a distinguished subset of the network variables (*explanation set*), given some observations (*evidence*). This problem, also known as the *Maximum a Posteriori Problem*, is known to be NP-hard, so exact computation is not always possible. As partial abductive inference in Bayesian networks can be viewed as a combinatorial optimization problem, Genetic Algorithms have been successfully applied to give an approximate algorithm for it (de Campos et al., 1999). In this work we approach the problem by means of Estimation of Distribution Algorithms, and an empirical comparison between the results obtained by Genetic Algorithms and Estimation of Distribution Algorithms is carried out.

Keywords: Abductive inference, most probable explanation, maximum a posteriori problem, probabilistic reasoning, Bayesian networks, Evolutionary Computation, Genetic Algorithms, Estimation of Distribution Algorithms

1. Introduction

As stated in Chapter 2, Bayesian networks (BNs) (Pearl, 1988; Jensen, 1996) exploit independence properties of probability distributions to give a compact and natural representation. BNs also allow us to calculate probabilities by means of *local* computation, i.e. probabilistic computations are carried out over the initial pieces of information rather than using a global distribution.

Most work in probabilistic reasoning has been devoted to *evidence propagation* and *total abductive inference*. However, in this chapter we focus on *partial abductive inference*, a type of diagnostic reasoning that can be viewed as a generalization of total abductive reasoning. Although this problem seems to be more useful in practical applications than *total* abductive inference, it has received much less attention from the BNs community.

The chapter is organized as follows: In Section 2 the basic query types in probabilistic expert systems are introduced. In Section 3 we briefly survey how these queries are solved. In Section 4 we describe how this problem has been approached using Genetic Algorithms (GAs), and in Section 5 we present how to approach the problem using Estimation of Distribution Algorithms (EDAs). Section 6 is devoted to experimental evaluation, and finally, in Section 7, we give our conclusions.

2. Query types in probabilistic expert systems

Assume that we have a n-dimensional variable $\mathbf{X} = \{X_1, \ldots, X_n\}$ whose probability distribution can be obtained by the factorization provided by a BN. Reasoning in a BN is performed by updating the various probabilities in

the light of specific knowledge (*evidence* or *observations*). The basic type of queries in BNs are:

- *Evidence propagation.* In this type of query, given a set of observations ($\boldsymbol{X_O} = \boldsymbol{x_O}$), our task is to compute:

$$p(x_i|\boldsymbol{X_O} = \boldsymbol{x_O}) \tag{16.1}$$

  for all non observed variables X_i in the network.

- *Total Abductive Inference*, also known as the *Most Probable Explanation* (MPE) problem (Pearl, 1987). Here, our task is to find the most probable state of the network given a set of observations ($\boldsymbol{X_O} = \boldsymbol{x_O}$). More formally, if $\boldsymbol{X_U} = \boldsymbol{X} \setminus \boldsymbol{X_O}$ is the set of unobserved variables, then we aim to obtain the configuration $\boldsymbol{x_U^*}$ of $\boldsymbol{X_U}$ such that:

$$\boldsymbol{x_U^*} = \arg\max_{\boldsymbol{x_U}} p(\boldsymbol{x_U}|\boldsymbol{X_O} = \boldsymbol{x_O}). \tag{16.2}$$

- *Partial Abductive Inference*, also known as the *Maximum a Posteriori Problem* (MAP). In this problem, our goal is to obtain the most probable configuration only for a subset of the network variables known as the *explanation set* (Neapolitan, 1990). More formally, if $\boldsymbol{X_E} \subset \boldsymbol{X_U}$ is the explanation set, then we aim to obtain the configuration $\boldsymbol{x_E^*}$ of $\boldsymbol{X_E}$ such that:

$$\boldsymbol{x_E^*} = \arg\max_{\boldsymbol{x_E}} p(\boldsymbol{x_E}|\boldsymbol{X_O} = \boldsymbol{x_O}) = \arg\max_{\boldsymbol{x_E}} \sum_{\boldsymbol{x_R}} p(\boldsymbol{x_E}, \boldsymbol{x_R}|\boldsymbol{x_O}) \tag{16.3}$$

  where $\boldsymbol{X_R} = \boldsymbol{X_U} \setminus \boldsymbol{X_E}$. It is important to note that, in general, $\boldsymbol{x_E^*}$ is not equal to the configuration obtained from $\boldsymbol{x_U^*}$ by removing the *literals* not in $\boldsymbol{X_E}$, so we have to obtain $\boldsymbol{x_E^*}$ directly from Equation 16.3. Example 16.1 illustrates this situation.

Example 16.1 *Consider the network specified in Figure 16.1, where D_1, D_2 and S are propositional variables with two possible states each ($\Omega_{D_1} = \{d_1, \neg d_1\}$, $\Omega_{D_2} = \{d_2, \neg d_2\}$, $\Omega_S = \{s, \neg s\}$).*

If we observe that S is present, that is, $S = s$, then the most probable explanation is ($D_1 = \neg d_1, D_2 = d_2$). However, if variable D_2 is selected as the explanation set, then partial abductive inference produces ($D_2 = \neg d_2$) as the most probable explanation, which is different to the configuration

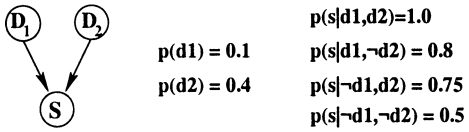

Figure 16.1 A small Bayesian network.

obtained by removing the literal corresponding to D_1 from the configuration obtained by total abductive inference.

In (total and partial) abductive inference, in general, we are interested in obtaining the K most probable configurations and not just the best one.

3. Solving queries

In principle, we can answer all the queries presented in the previous section by "simply" generating the joint distribution, and then taking it as our starting point, summing out (in the case of evidence propagation), searching for the configuration with maximum probability (in the case of total abduction), or applying both of the previous operations (in the case of partial abduction). However, this approach is intractable even for networks with a small number of variables.

In recent years many algorithms have been proposed to solve the problem of evidence propagation by taking advantage of the conditional (in)dependencies among the variables given by the graphical structure. Nowadays, the most practical inference methods for Bayesian networks are those based on the *clique tree* algorithm (Jensen et al., 1990; Lauritzen and Spiegelhalter, 1988; Shenoy and Shafer, 1990). This class of propagation algorithms are based on the transformation (see Figure 16.2) of the Bayesian network into a secondary structure called a *clique tree* (or *join/junction tree*), in which the calculations are carried out. This method is based on the use of two operations: *marginalization* (addition) and *combination* (multiplication); and is divided into two phases: *collectEvidence* (messages are passed from leaves to root) and *distributeEvidence* (messages are passed from root to leaves). See Jensen (1996) and Shafer (1996) for details.

Although the propagation problem is NP-hard (Cooper, 1990) in the worst case, the clique tree algorithms work efficiently for moderately size networks, with their efficiency being strongly related to the size of the clique tree[1] obtained from the Bayesian network. For example, the same algorithm will perform better with the clique tree depicted in Figure 16.2(c) than with the clique tree depicted in Figure 16.2(b).

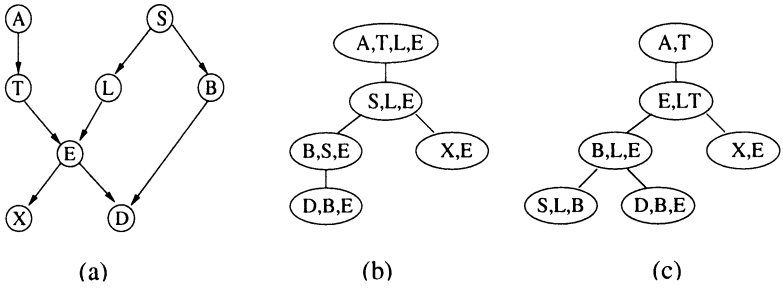

Figure 16.2 Two possible clique trees for the same network.

Dawid (1992) has shown that the MPE can be found using evidence propagation methods but replacing summation with maximum in the marginalization operator (due to the distributive property of maximum with respect to multiplication). Therefore, the process of searching for the most probable explanation has the same complexity as probability propagation. However, searching for K MPEs is a more complex problem, and to obtain K MPEs more intricate methods have to be used (Nilsson, 1998; Seroussi and Goldmard, 1994).

In partial abductive inference, the process of finding the configuration $\boldsymbol{x}_{\boldsymbol{E}}^*$ is more complex than that of finding $\boldsymbol{x}_{\boldsymbol{U}}^*$, because not all clique trees obtained from the original BN are valid. In fact, because summation and maximum have to be used simultaneously and these operations do not have commutative behaviour, the variables of $\boldsymbol{X}_{\boldsymbol{E}}$ must form a sub-tree of the complete tree. The problem of finding a *valid* clique tree for a given explanation set $\boldsymbol{X}_{\boldsymbol{E}}$ is studied in de Campos et al. (2000). From that study it can be concluded that the size of the clique tree obtained for partial abductive inference grows (in general) in an exponential way with respect to the number of variables included in the explanation set, with the worst case being when $\boldsymbol{X}_{\boldsymbol{E}}$ contains around half of the variables in the network. After that, it decreases with the size of the tree when $\boldsymbol{X}_{\boldsymbol{E}}$ contains all the variables being the same as when $\boldsymbol{X}_{\boldsymbol{E}}$ contains a single variable. Therefore, the computer resources (time and memory) needed can be so high that the problem becomes unsolvable by exact computation, even for medium-size networks.

4. Tackling the problem with Genetic Algorithms

As we have seen in Chapter 1, Genetic Algorithms (GAs) are now a popular technique for approaching difficult combinatorial problems. GAs have been previously used for solving NP-hard problems related to Bayesian networks, including: triangulation of graphs (Larrañaga et al., 1997), imprecise proba-

bilities propagation (Cano and Moral, 1996), estimation of a causal ordering for the variables (de Campos and Huete, 2000; Larrañaga et al., 1996a), and learning (Larrañaga et al., 1996b). Given the success of these applications, the NP-hardness of the abductive inference problem, and the fact that abductive inference in BNs can be viewed as a combinatorial optimization problem, several authors have used GAs to solve (in an approximate way) both types of abductive inference problems: *total* (Gelsema, 1995; Rojas-Guzman and Kramer, 1996) and *partial* (de Campos et al., 1999).

In Rojas-Guzman and Kramer (1996) a chromosome is represented as a copy of the graph included in the BN, but in which every variable has been instantiated to one of its possible states. This representation makes it possible to implement the crossover operator as the interchange of a subgraph with center in a variable X_i, with X_i selected randomly for each crossover. In Gelsema's algorithm, a chromosome is a configuration of the unobserved variables ($\boldsymbol{X_U} = \boldsymbol{X} \setminus \boldsymbol{X_O}$), i.e. a string of integers. In this case, crossover is implemented as the classical one point crossover.

The common feature of both GAs for total abductive inference is the way in which the fitness of an individual is calculated. Although we want to maximize $p(\boldsymbol{x_U}|\boldsymbol{x_O})$, this expression is proportional to $p(\boldsymbol{x_U}, \boldsymbol{x_O})$, so we can use this instead as the fitness for the chromosome $\boldsymbol{x_U}$. As $(\boldsymbol{x_U}, \boldsymbol{x_O})$ represents a complete instantiation of all the variables in the network, then we can use the factorization of the joint distribution as expressed in Eq. (16.4) to calculate $p(\boldsymbol{x_U}, \boldsymbol{x_O})$. Therefore, the evaluation of a chromosome requires n multiplications:

$$p(\boldsymbol{X} = \boldsymbol{x}) = p(\boldsymbol{X_U} = \boldsymbol{x_U}, \boldsymbol{X_O} = \boldsymbol{x_O}) = \prod_{i=1}^{n} p(x_i|\boldsymbol{pa}_i). \tag{16.4}$$

Dealing with partial abductive inference using GAs appears 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 function. In fact, in the partial case, $(\boldsymbol{x_E}, \boldsymbol{x_O})$ does not represent a configuration of all the variables in the network, so Eq. (16.4) cannot be applied directly. In this case, the variables not observed and not in the explanation set, $\boldsymbol{X_R} = \boldsymbol{X} \setminus (\boldsymbol{X_E} \cup \boldsymbol{X_O})$, have to be removed by addition. Therefore, to evaluate an individual $\boldsymbol{x_E}$ using Eq. (16.4), we have to apply

$$p(\boldsymbol{x_E}|\boldsymbol{x_O}) \propto p(\boldsymbol{x_E}, \boldsymbol{x_O}) = \sum_{\boldsymbol{x_R}} p(\boldsymbol{x_E}, \boldsymbol{x_O}, \boldsymbol{x_R}), \tag{16.5}$$

that is, Eq. (16.4) has to be applied $|\Omega_{\boldsymbol{X_R}}|$ times, where $\Omega_{\boldsymbol{X_R}}$ is the set of possible configurations of $\boldsymbol{X_R}$. For example, if we have a network with 50 propositional binary variables, $|\boldsymbol{X_E}| = 15, |\boldsymbol{X_R}| = 30$ and $|\boldsymbol{X_O}| = 5$, then Eq.

(16.4) has to be applied 2^{30} times. Clearly, this is computationally intractable given the large number of evaluated individuals in the execution of a GA.

For this reason, in de Campos et al. (1999) the fitness $p(\boldsymbol{x}_{\boldsymbol{E}}, \boldsymbol{x}_{\boldsymbol{O}})$ of a chromosome $\boldsymbol{x}_{\boldsymbol{E}}$ is computed by using probability propagation over a clique tree. Below, we describe this evaluation function and some other details of a slightly modified version of the GAs described in de Campos et al. (1999), which will be used for the experiments in this chapter.

4.1 A Genetic Algorithm for partial abductive inference

We briefly describe the representation, the evaluation function and the structure of the GA used in de Campos et al. (1999).

- *Representation.* In our algorithm, a chromosome will be a configuration of the variables in the explanation set, that is, a string of integers of length $|\boldsymbol{X}_{\boldsymbol{E}}|$.

- *The evaluation function.* The fitness of a chromosome $\boldsymbol{x}_{\boldsymbol{E}}$ is computed by the process described below, where $\mathcal{T} = \{C_1, \ldots, C_t\}$ is a rooted clique tree, with root C_1.

  1. Enter the evidence $\boldsymbol{x}_{\boldsymbol{O}}$ in $\mathcal{T}$.
  2. Enter (as evidence) the configuration $\boldsymbol{x}_{\boldsymbol{E}}$ in $\mathcal{T}$.
  3. Perform *CollectEvidence* from the root (C_1) (i.e., an upward propagation).
  4. $p(\boldsymbol{x}_{\boldsymbol{E}}, \boldsymbol{x}_{\boldsymbol{O}})$ 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 Jensen (1996) for details of 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 (de Campos et al., 2000). In addition, in de Campos et al. (1999) it is shown how the tree can be pruned (for a concrete explanation set) in order to avoid the repetition of unnecessary computations when a new chromosome is being evaluated.

- *Structure of the GA.* The GA used in de Campos et al. (1999) is based on the *modified GA* (modGA) proposed by Michalewicz (1996). This GA falls into the category of preservative, generational and elitist selection, and enjoys similar theoretical properties to the classical GA. The main

modification with respect to the classical GA is that in modGA we do not perform the classical selection step, but instead we select independently r distinct chromosomes (usually those that fit best) from $P(t)$ to be copied to $P(t+1)$. In de Campos et al. (1999) the parameters used are the following:

- Select the best 50% chromosomes from $P(t)$ and copy them to $P(t+1)$. In this way we ensure the population diversity and the premature convergence problem is avoided.
- 35% of the new population is obtained by crossover. One parent is selected from $P(t)$ with a probability proportional to its *rank* and the other in a random way. The crossover operator used is the classical two-point crossover, and the two children obtained are copied to $P(t+1)$.
- 15% of the new population is obtained by mutation. Mutation is carried out by selecting a chromosome from $P(t)$ and modifying one of its components, then copying the resulting chromosome to $P(t+1)$. So, we apply genetic operators on whole individuals as opposed to individual bits (classical mutation). As Michalewicz (1996) points out, this would provide a uniform treatment of all operators used in the GA. The *parents* for mutation are selected from $P(t)$ with a probability proportional to their *rank*, except for the *best* chromosome, which is always selected as a parent (thus, the neighbourhood of the best chromosome is explored).

The numbers 50, 35 and 15 have been selected by experimentation. Notice that in $P(t+1)$ only half of the population is new, and so only those chromosomes are candidates to be evaluated in each generation. This fact is important in our problem because of the evaluation function complexity. In addition, a hash table is used to avoid the reevaluation of previously seen individuals. When a new chromosome is evaluated, it is tested for whether it must be included in $Kbest$, an array which contains the K best individuals obtained so far.

5. Tackling the problem with Estimation of Distribution Algorithms

As described in Chapter 3, EDAs constitute a new approach for Evolutionary Computation, where the crossover and mutation operators have been replaced in each generation by the estimation of a probability distribution and its posterior simulation.

As far as we know, this is the first time that the partial abductive inference in Bayesian networks has been tackled by means of EDAs. The characteristics of the proposed approach are as follows:

- *Representation.* An individual in EDAs is equal to a chromosome of the GA: a configuration of the explanation variables, i.e. the Bayesian network used to generate populations will have as variables the set $\boldsymbol{X_E}$.
- *Evaluation function.* This has been described in Section 4.1 in this chapter.
- *Adaptation of EDAs in order to search the K MAPs.* Although we are aware that one ad–hoc approach to the K MAPs problem would imply that the length of the individuals would be $K\times \mid \boldsymbol{X_E} \mid$, in this case we have decided to adapt EDAs in a more general way, that is related to their meta-heuristic character. In this way, we impose in the simulation phase of EDAs the constraint that the generated individuals must be different from the individuals previously simulated, both in the same generation and also in previous generations. Where this condition is not verified after 50 attempts, the repeated individual is added to the population. Once the simulated phase is finished we select the N best individuals from the combined pool of individuals generated in this generation and the individuals used to induce the probabilistic model in the previous generation. From the individuals selected in this manner, a new probabilistic model will be induced (see Section 2 in Chapter 3 for details about the general scheme of EDAs).
- *Types of EDAs used.* For the experiments we have selected three types of EDAs which present an increasing complexity in the factorization of the probability distribution of the selected individuals:
  - UMDA (Mühlenbein, 1998), without dependencies,
  - MIMIC (De Bonet et al., 1997), bivariate dependencies, and
  - EBNA (Etxeberria and Larrañaga, 1999), multiple dependencies.

  More information about these algorithms can be found in Section 3 in Chapter 3 of this book.

6. Experimental evaluation

In order to perform the empirical comparison among the proposed algorithms, we have carried out five experiments. Three of these experiments have been carried out on the well known *Alarm* network (Beinlich et al., 1989), and the others on two artificially generated Bayesian networks: *random100* and

random100e. The networks *random100* and *random100e* have been generated using the same procedure for their structure, but different procedures for their probability tables. In *random100*, the probabilities were generated using uniform random numbers, but in *random100e* the process is more complex: two uniform random numbers, x and y were generated, and the probability of the two values (marginals for root nodes and conditionals for the rest) of a variable are determined by normalizing x^5 and y^5, which give rise to extreme probabilities. Table 16.1 gives some information about these networks, where *min*, *max* and *mean* refer to the size of the probability table attached to each node.

Table 16.1 Some characteristics of the networks used in the experiments.

Network	*nodes*	*arcs*	*states*	*min*	*max*	*mean*
Alarm	37	46	$\{2,3,4\}$	2	108	20.3
random100	100	122	2	2	32	5.88
random100e	100	128	2	2	64	6.54

Table 16.2 shows a brief description of each experiment. Column $|\boldsymbol{X_E}|$ gives the number of variables included in the explanation set, while column $\boldsymbol{X_E}$ shows the way that these variables were selected for inclusion in the explanation set. In all the experiments, the variables to be included in the explanation set were selected in a *pseudo-random* way, that is, several sets containing $|\boldsymbol{X_E}|$ variables were randomly generated, and the most difficult one to be solved by exact computation was chosen. The difficulty of a problem was measured as a function of the time and space needed to solve the problem exactly. To solve the problem exactly we have used software implemented in Java and running on an Intel Pentium III (600 MHz) with 384 MB of RAM, a Linux operating system, and the JDK 1.2 virtual machine. The time needed to exactly solve experiments 1, 2 and 3 was between one and one and a half hours, while solving a total abductive inference problem using this software takes less than 0.5 seconds. For experiments 4 and 5, we have not been able to solve the problem exactly because of memory requirements, i.e. the "out of memory error" was obtained in both cases. This error is due to the enormous size of the clique trees obtained from these networks, by means of a compilation constrained by the selected explanation sets. In these networks, total abductive inference takes less than 9 seconds.

In all the experiments five variables have been selected as evidence, and have been instantiated to their "a priori" least probable state. In the five experiments we have taken $K = 50$, that is, we look for the 50 MAPs.

Table 16.2 Description of the experiments.

#exp.	$\lvert X_E \rvert$	*network*	X_E	$\lvert \Omega_{X_E} \rvert$
1	18	*Alarm*	pseudo-random	143,327,232
2	19	*Alarm*	pseudo-random	214,990,848
3	20	*Alarm*	pseudo-random	382,205,952
4	30	*random100*	pseudo-random	1,073,741,824
5	30	*random100e*	pseudo-random	1,073,741,824

The data we have collected during execution of the algorithms is related to the probability mass of the K MAPs found. Thus, *mass1, mass10, mass25* and *mass50* represent the probability mass of the first 1, 10, 25 and 50 MAPs found by the exact algorithm, and *mass1′, mass10′, mass25′* and *mass50′* represent the probability mass of the first 1, 10, 25, and 50 MAPs found by the proposed algorithms. For experiments 1, 2 and 3, we present the percentage of probability mass obtained with respect to the exact algorithm ($\%massX' = \frac{massX' * 100}{massX}$). For experiments 4 and 5, because of the absence of exact results, we present $massX'$ directly. In order to test the *anytime* behaviour of the algorithms, results are presented for (approximately) every 500 different evaluated individuals. Finally, all the algorithms have been run 50 times, so all the results are averages.

Finally, the experimentation has consisted of applying the four algorithms presented in this chapter (UMDA, MIMIC, EBNA, and GA) to the five examples shown. In each experiment we have considered 8 different population sizes (50, 100, 150, 200, 250, 300, 400 and 500), but due to space limitations we only present for each algorithm the results related to one selected size (that for which the best results –on average– were obtained). In all cases, the initial population has been generated randomly, and the algorithm stops when the number of different evaluated individuals is greater than 5000. Note that the algorithm cannot stop when exactly 5000 individuals have been evaluated, because the current generation has to be finished. The same applies to the intermediate points selected to study the anytime behaviour (500, 1000, ...). Tables 16.3 to 16.7 show the obtained results for $(\%)mass1'$, $(\%)mass10'$, $(\%)mass25'$, and $(\%)mass50'$ in each experiment, where the entries are interpreted as $average \pm standard\ deviation$. Figures 16.3 to 16.7 show the anytime behaviour of the four algorithms with respect to $(\%)mass1'$.

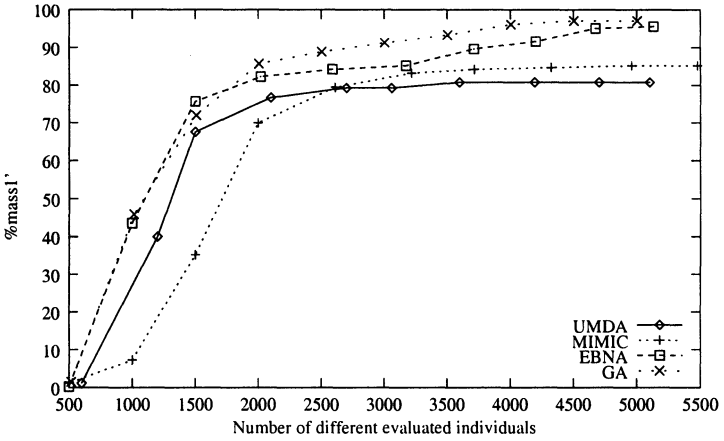

Figure 16.3 A plot of *%mass1'* for experiment 1.

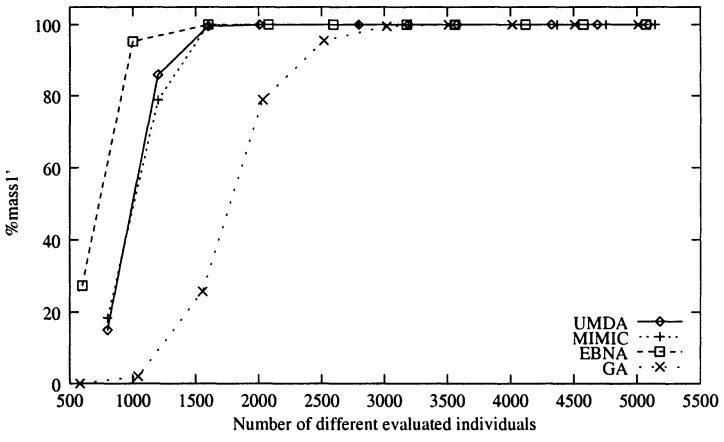

Figure 16.4 A plot of *%mass1'* for experiment 2.

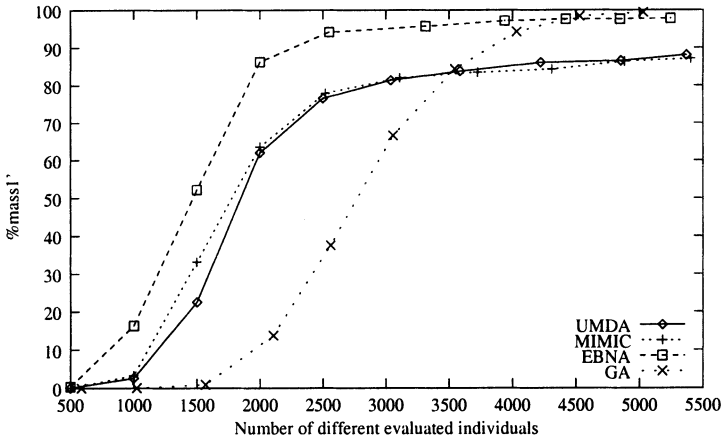

Figure 16.5 A plot of *%mass1'* for experiment 3.

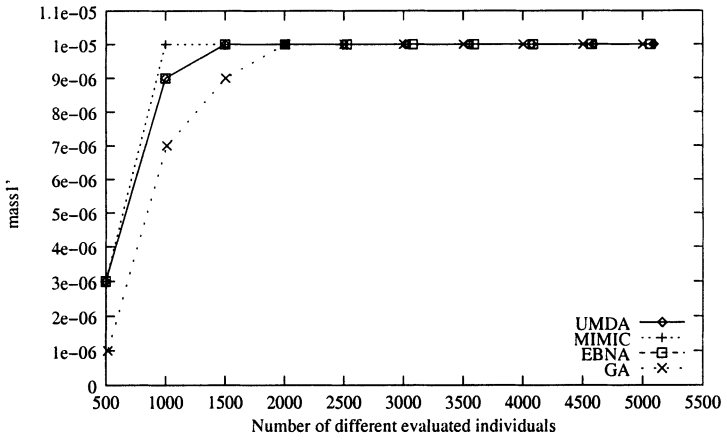

Figure 16.6 A plot of *mass1'* for experiment 4.

Table 16.3 Results for experiment 1. Population size was 300 for UMDA, 500 for MIMIC, 250 for EBNA and 100 for GA.

	%mass1'	*%mass10'*	*%mass25'*	*%mass50'*
UMDA	80.77 ± 10.32	75.16 ± 12.25	71.07 ± 12.52	68.25 ± 12.43
MIMIC	85.21 ± 12.20	80.43 ± 15.22	76.66 ± 17.07	74.19 ± 17.91
EBNA	95.56 ± 9.57	93.62 ± 12.74	92.44 ± 14.95	91.75 ± 16.21
GA	97.04 ± 0.81	89.63 ± 1.38	85.16 ± 1.88	83.19 ± 2.09

Table 16.4 Results for experiment 2. Population size was 400 for UMDA, 400 for MIMIC, 200 for EBNA and 200 for GA.

	%mass1'	*%mass10'*	*%mass25'*	*%mass50'*
UMDA	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	98.95 ± 0.25
MIMIC	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	98.99 ± 0.28
EBNA	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.62 ± 0.46
GA	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	99.38 ± 0.05

Table 16.5 Results for experiment 3. Population size was 500 for UMDA, 500 for MIMIC, 500 for EBNA and 300 for GA.

	%mass1'	*%mass10'*	*%mass25'*	*%mass50'*
UMDA	88.07 ± 29.04	81.32 ± 34.75	77.62 ± 35.81	74.54 ± 36.35
MIMIC	87.14 ± 30.28	81.06 ± 34.01	76.67 ± 35.49	73.26 ± 36.02
EBNA	97.76 ± 12.62	96.79 ± 13.12	95.19 ± 14.86	93.80 ± 16.98
GA	99.37 ± 0.21	97.36 ± 0.47	93.00 ± 1.5	87.55 ± 5.33

6.1 Experimental conclusions

As we can see from Tables 16.3 to 16.7 the results obtained by the four algorithms are similar in experiments 2, 4 and 5, while significant differences can be observed in the other two experiments. As an explanation of this fact we conjecture that the problems considered in experiments 2, 4 and 5 give rise to less complex search spaces than those generated by the problems considered

Table 16.6 Results for experiment 4. Population size was 100 for UMDA, 100 for MIMIC, 100 for EBNA and 100 for GA.

	mass1'	*mass10'*	*mass25'*	*mass50'*
UMDA	0.000010 ± 0	0.000077 ± 0	0.000167 ± 0	0.000295 ± 0
MIMIC	0.000010 ± 0	0.000077 ± 0	0.000167 ± 0	0.000295 ± 0
EBNA	0.000010 ± 0	0.000077 ± 0	0.000167 ± 0	0.000295 ± 0
GA	0.000010 ± 0	0.000076 ± 0	0.000166 ± 0	0.000297 ± 0

Table 16.7 Results for experiment 5. Population size was 500 for UMDA, 500 for MIMIC, 300 for EBNA and 200 for GA.

	mass1'	*mass10'*	*mass25'*	*mass50'*
UMDA	0.014197 ± 0	0.090864 ± 0	0.164966 ± 0.002	0.237114 ± 0.006
MIMIC	0.014197 ± 0	0.090848 ± 0	0.163928 ± 0.002	0.232105 ± 0.007
EBNA	0.014197 ± 0	0.091064 ± 0	0.168794 ± 0.003	0.247711 ± 0.008
GA	0.014197 ± 0	0.091073 ± 0	0.168202 ± 0.003	0.244589 ± 0.008

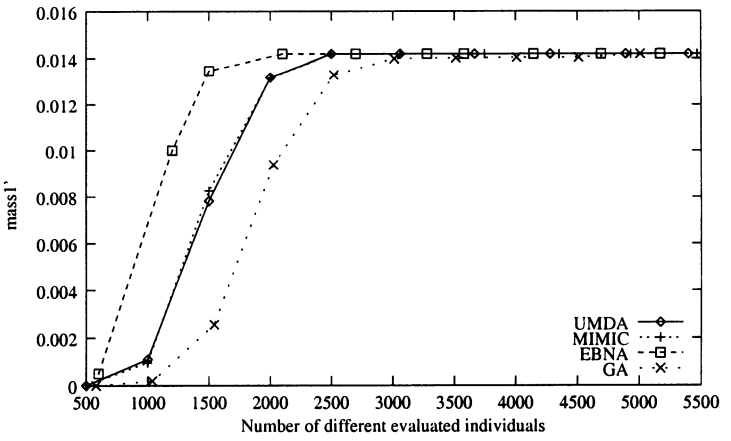

Figure 16.7 A plot of *mass1'* for experiment 5.

in experiments 1 and 3. From analysis of Tables 16.3 and 16.5 (experiments 1 and 3) we obtain the following conclusions:

- Among the three EDAs used in the experimental evaluation, EBNA clearly outperforms UMDA and MIMIC. This fact is not surprising because EBNA can deal with unconstrained probability distributions, while UMDA and MIMIC have to deal, in general, with approximations.
- The performance of the GA is superior to UMDA and MIMIC in all the cases.
- The comparison between the GA and EBNA needs to be more detailed. With respect to $\%mass1'$ the GA seems to have a better behaviour than EBNA, although it is outperformed by EBNA in the search of the K most probable explanations.
- It is interesting to analyze the variability shown by the four algorithms. Thus, from the point of view of the standard deviations we can clearly establish the following pattern: MIMIC > UMDA > EBNA >> GA.
- As we have mentioned before, we have experimented with eight different population sizes, and from the results obtained (and not shown here due to space limitations) it is observed that the behaviour of the EDA approach with respect to changes in the population size is more robust than the one shown by the GA.

With respect to the anytime behaviour of the proposed algorithms (see Figures 16.3 to 16.7), it is clear that the number of evaluations required by EDAs in order to get good solutions is smaller than the number required by the GA, which shows a slower convergence.

7. Concluding remarks

In this chapter we have studied the problem of partial abductive inference in Bayesian networks. The problem has been approached using a previously known GA (de Campos et al., 1999) and three different algorithms (UMDA, MIMIC and EBNA) based on the novel approach of EDAs.

From the empirical comparison carried out we can conclude that UMDA and MIMIC are clearly outperformed by GA and EBNA, while differences between GA and EBNA are small and dependent on the parameter being considered (searching for the best explanation or for the K best). Anyway, given the obtained results, both algorithms (GA and EBNA) constitute a good choice for approaching the problem considered here.

Regarding future work, as it seems that GAs and EDAs outperform each other with respect to different parameters ($(\%)mass(1', 10', 25', 50')$, standard deviation, convergence speed) we plan to experiment with the hybridization of both types of algorithm in order to ascertain whether the joint approach is an improvement on these individual approaches. Furthermore, we plan to perform

a deeper study of the adequate population size for each algorithm. Another interesting starting point for future work could be to take advantage of the initially known structure of the Bayesian network in order to constrain the graphical model to be learnt during the search.

Acknowledgments

This work has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) under Projects TIC97-1135-CO4-01 and TIC97-1135-C04-03.

Notes

1. The size of a clique tree is the sum of the sizes associated with each of its cliques. The size of a clique is the product of the number of different states that each variable within the clique can take.

References

Beinlich, I.A., Suermondt, H.J., Chavez, R.M., and Cooper, G.F. (1989). 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*, pages 247–256. Springer-Verlag.

Cano, A. and Moral, S. (1996). A genetic algorithm to approximate convex sets of probabilities. In *Procceedings of the 6th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'96)*, pages 847–852.

Cooper, G.F. (1990). Probabilistic inference using belief networks is NP-hard. *Artificial Intelligence*, 42(2-3):393–405.

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

De Bonet J.S., Isbell, C.L., and Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Vol. 9.

de Campos, L.M., Gámez, J.A., and Moral, S. (1999). Partial Abductive Inference in Bayesian Belief Networks using a Genetic Algorithm. *Pattern Recognition Letters*, 20(11-13):1211–1217.

de Campos, L.M., Gámez, J.A., and Moral, S. (2000). 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)*, pages 1270–1277.

de Campos, L.M. and Huete, J.F. (2000). Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing. In

8th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00), pages 333–340.

Etxeberria, R. and Larrañaga, P. (1999). Global optimization with Bayesian networks. In *II Symposium on Artificial Intelligence. CIMAF99. Special Session on Distributions and Evolutionary Optimization*, pages 332–339.

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

Jensen, F.V. (1996). *An introduction to Bayesian Networks.* UCL Press.

Jensen, F.V., Lauritzen, S.L., and Olesen, K.G. (1990). Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, 4:269–282.

Larrañaga, P., Kuijpers, C., Murga, R., and Y. Yurramendi (1996a). Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on System, Man and Cybernetics*, 26(4):487–493.

Larrañaga, P., Kuijpers, C., Poza, M., and Murga, R. (1997). Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, 7:19–34.

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R., and Kuijpers, C. (1996b). Structure learning of Bayesian networks by genetic algorithms. A perfomance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926.

Lauritzen, S.L. and Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J.R. Statistics Society. Serie B*, 50(2):157–224.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag.

Mühlenbein, H.M. (1998). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346.

Neapolitan, R.E. (1990). *Probabilistic Reasoning in Expert Systems. Theory and Algorithms.* Wiley Interscience.

Nilsson, D. (1998). An efficient algorithm for finding the M most probable configurations in Bayesian networks. *Statistics and Computing*, 2:159–173.

Pearl, J. (1987). Distributed revision of composite beliefs. *Artificial Intelligence*, 33:173–215.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann.

Rojas-Guzman, C. and Kramer, M.A. (1996). An evolutionary computing approach to probabilistic reasoning in Bayesian networks. *Evolutionary Computation*, 4:57–85.

Seroussi, B. and Goldmard, J.L. (1994). An algorithm directly finding the K most probable configurations in Bayesian networks. *International Journal of Approximate Reasoning*, 11:205–233.

Shafer, G.R. (1996). *Probabilistic Expert Systems*. Society for Industrial and Applied Mathematics (SIAM).

Shenoy, P.P. and Shafer, G.R. (1990). Axioms for probability and belief-function propagation. In Shachter, R., Levitt, T., Kanal, L., and Lemmer, J., editors, *Uncertainty in Artificial Intelligence, 4*, 169–198. Elsevier Science Publishers B.V. (North-Holland).