ELSEVIER

Pattern Recognition Letters 20 (1999) 1211–1217

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Partial abductive inference in Bayesian belief networks using a genetic algorithm ☆

L.M. de Campos [a], J.A. Gámez [b,*], S. Moral [a]

[a] *Departamento de Ciencias de la Computación e Inteligencia Artificial, Escuela Técnica Superior de Ingeniería Informática, Universidad de Granada, 18071 Granada, Spain*

[b] *Departamento de Informática, Escuela Politécnica Superior, Universidad de Castilla-La Mancha, 02071 Albacete, Spain*

Abstract

Abductive inference in Bayesian belief networks is the process of generating the K most probable configurations given an observed evidence. When we are only interested in a subset of the network's variables, this problem is called partial abductive inference. Both problems are NP-hard, and so exact computation is not always possible. This paper describes an approximate method based on genetic algorithms to perform partial abductive inference. We have tested the algorithm using the *alarm* network and from the experimental results we can conclude that the algorithm presented here is a good tool to perform this kind of probabilistic reasoning. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Abductive inference; Most probable explanation; Bayesian belief networks; Genetic algorithms; Probabilistic reasoning

1. Introduction

A *Bayesian belief network* (Pearl, 1988; Jensen, 1996) is a directed acyclic graph (DAG) 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 $X_{\mathscr{U}} = \{X_1, \ldots, X_n\}$ is the set of variables in the network, then the joint probability can be calculated as

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

where $pa(X_i)$ contains the parents of X_i. Eq. (1) is known as the *chain-rule*. Bayesian belief networks (BBNs) provide a sound formalism for probabilistic reasoning under uncertainty.

Abduction is defined as the process of generating a plausible explanation for a given set of observations or facts. The main application of abduction is in the field of diagnostic reasoning. In the context of BBN, abductive inference (Pearl, 1988) corresponds to finding the maximum a posteriori probability state of the network, given the instantiated nodes (*the evidence*). If $X_{\mathscr{U}}$ is the set of variables in the network, X_O is the set of observed variables and X_U is the set of unobserved variables, we want to obtain the configuration x_U^* of X_U such that

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

* Corresponding author.

E-mail address: jgamez@info-ab.uclm.es (J.A. Gámez)

0167-8655/99/$ - see front matter
PII: S0167-8655(99)00088-4

$$x_U^* = \arg\max_{x_U} P(x_U \mid x_O), \tag{2}$$

where $X_O = x_O$ is the observed evidence. In general, we are interested in the K most probable explanations (MPE).

Abductive reasoning in BBNs is an NP-hard problem (Shimony, 1994). This fact has motivated the development of approximate algorithms. As abductive inference in BBNs is a combinatorial optimization problem, several authors have used genetic algorithms (e.g., Michalewicz, 1996) to solve it in an approximate way (Gelsema, 1995; Rojas-Guzman and Kramer, 1996). In these papers an individual/chromosome of the population is a configuration of the unobserved variables, represented as a string of integers. Notice that $P(x_U \mid x_O)$ is proportional to $P(x_U, x_O)$, so this value can be used as the fitness for x_U, and the *chain rule* can be applied in order to evaluate a chromosome. Thus, to evaluate a chromosome $|x_{\mathscr{U}}|$ multiplications are done.

In this paper we are interested in obtaining the K MPEs only for a subset of the network's variables called *explanation set* (Neapolitan, 1990). This problem is known as *partial abductive inference* and we think that in practical applications it is more interesting than the classical abductive inference problem, because we can select as the explanation set the variables representing diseases in a medical diagnosis problem, the variables representing critical components (starter, battery, alternator, etc.) in a car diagnosis problem, etc.

If we note by $X_E \subset X_U$ the explanation set, then we want to obtain the configuration x_E^* of X_E such that

$$\begin{aligned} x_E^* &= \arg\max_{x_E} P(x_E \mid x_O) \\ &= \arg\max_{x_E} \sum_{x_R} P(x_E, x_R \mid x_O), \end{aligned} \tag{3}$$

where $X_R = X_U \setminus X_E$. In general, x_E^* is not equal to the configuration obtained by removing from x_U^* the literals corresponding to the variables of X_R. Therefore, we need to obtain x_E^* directly (using the previous equation).

Recently, Gámez (1998) has shown that partial abductive inference in BBN is more complex than classical abductive inference, because the size of the junction tree used for the probabilistic propagation grows in relation to the size of the junction tree obtained without restrictions in the deletion sequence (used in the classical abductive problem or in probability propagation).

In this work we propose to solve the problem using a genetic algorithm (GA). The paper is organized as follows. In Section 2 we study how to evaluate the individuals. In Section 3, the main components of the genetic algorithm are detailed. In Section 4 the experiments carried out are described. In Section 5 we discuss the algorithm and, finally, in Section 6, we consider the conclusions.

2. The evaluation function

In our problem a chromosome of the population is represented as a configuration of the explanation set's variables. However, as we must remove (by addition) the variables in X_R, the chain rule cannot be used to evaluate a chromosome because of the high number of necessary operations (additions and multiplications). For example, if we have a network with $|X_{\mathscr{U}}| = 50$, $|X_E| = 15$, $|X_R| = 30$ and $|X_O| = 5$, the number of operations is bounded by 2^{30} additions and $50 \cdot 2^{30}$ multiplications. Clearly, this is computationally intractable given the big number of evaluated individuals in a GA. Because of this fact, our proposal is to evaluate a chromosome by means of a probabilistic propagation.

Given an evidence $X_O = x_O$, propagation algorithms allow us to calculate $p(X_i \mid x_O)$ for every $X_i \in X_{\mathscr{U}} \setminus X_O$. The calculations are carried out in a secondary structure (obtained from the original BBN) called a *junction tree*, where the evidence x_O has been inserted. The method is based on the use of two operations: *marginalization* (addition) and *combination* (multiplication); and the propagation is performed in two phases:

1. a *CollectEvidence* phase, where messages are sent from the leaves to a chosen root, and
2. a *DistributeEvidence* phase, where messages are reversed.

For a detailed study of propagation see (Jensen, 1996).

To evaluate a chromosome we need to obtain $p(x_E, x_O)$, so the previous propagation scheme is modified in the following way:

1. Instantiate the evidence x_O in the junction tree.
2. Instantiate (as evidence) the chromosome x_E in the junction tree.
3. Perform *CollectEvidence* from the root (i.e., an ascendent propagation).
4. $p(x_E, x_O)$ is equal to the sum of the potential stored in the root.

The junction tree used here can be obtained without restrictions in the deletion sequence and its size is smaller than the one used for exact partial abduction. However, even with this general join tree, this is not a simple cost-calculation function and two operations have been defined in order to improve the evaluation process.

2.1. Consistent marginalization

In order to avoid the need of reloading the initial potentials of the junction tree when a new chromosome must be evaluated, we introduce the following marginalization operator.

Definition 1. If $\psi(X_I, X_J)$ is the potential defined over the set of variables $X_I \cup X_J$, X_K is a set of variables, Ω_{X_K} is the set of possible configurations for X_K, and $x_K \in \Omega_{X_K}$ is one of its possible configurations, then we define the marginalization of $X_I \cup X_J$ to X_I consistent with x_K as

$$\psi^{\downarrow_{C\{X_K=x_K\}}X_I}(x_I) = \begin{cases} \psi^{\downarrow X_I}(x_I) = \sum_{x_J} \psi(x_I, x_J) & \text{if } X_K \cap X_J = \emptyset, \\ \sum_{x_J : x_J^{\downarrow (X_K \cap X_J)} = x_K^{\downarrow (X_K \cap X_J)}} \psi(x_I, x_J) & \text{if } X_K \cap X_J \neq \emptyset. \end{cases}$$

In propagation, we use *marginalization consistent* with x_E (the chromosome being evaluated), so it is not necessary to instantiate it in the junction tree, and we do not need to reload the initial potentials when a new chromosome must be evaluated.

2.2. Precomputation

Because of the ascendent character of the propagation, it can be observed that some computations are always the same, with independence of the configuration being evaluated. Concretely, if a leaf node of the junction tree does not contain any variable of X_E (to be removed by marginalization), then this message is the same for all configurations $x_E \in \Omega_{X_E}$. In this case we propose to calculate this message only the first time, then it is combined with the potential stored in its father node and the leaf node is deleted. This process is repeated (from leaves to root) until no more "leaf" nodes can be deleted. The complexity of this process is equivalent to performing an ascendent propagation.

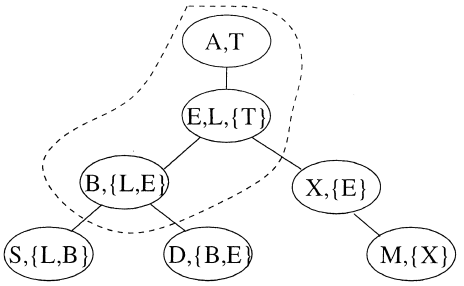

Fig. 1. Precomputation taking $X_E = \{A, B, E\}$.

Therefore, given the explanation set, the initial junction tree is pruned in order to avoid unnecessary computations. Thus, the precomputated junction tree is smaller than the initial one and the probabilistic propagations are carried out in a more efficient manner. For example, taking the junction tree in Fig. 1 (where the variables outside the brackets are the variables which have to be removed) and $X_E = \{A, B, E\}$ as the explanation set, the precomputated junction tree contains only the nodes inside the dashed region.

3. The genetic algorithm

With this evaluation function, we use an *elitist* genetic algorithm in the following way:

Problem representation. We represent a configuration x_E as a string of integers of length $|X_E|$. Each position in the string contains the state (codified as an integer) of its corresponding variable in X_E.

Generation of the initial population. The individuals of the initial population are created in two manners:

- 50% randomly (the search starts with points in the whole search space).
- 50% using the BBN a priori probabilities and the observed evidence (x_O). Concretely, we generate a configuration of all the variables in the network by fixing the variables instantiated (X_O) to their corresponding values (x_O) and generating values for the rest of the variables using probabilistic logic sampling (Henrion, 1988), finally the obtained configuration is projected over X_E. This procedure is similar to the one proposed in (Gelsema, 1995) and ensures that the search starts in promising regions.

Transition from one population $P(t)$ to the next $P(t+1)$. To obtain a new generation a similar procedure to the *modGA* proposed in (Michalewicz, 1996) is used. This genetic algorithm falls into the category of preservative, generational and elitist selection and has the following parameters:

- 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 fitness, while the other parent is selected in a random way. The crossover operator used is the classical two-points crossover, and the two children obtained are copied to $P(t+1)$.
- 15% of the new population is obtained by mutation. We use the following procedure: select a chromosome from $P(t)$ and mutate its components in accordance with a probability of mutation $p_m = 1/|X_E|$, and copy the new chromosome to $P(t+1)$. The *parents* for mutation are selected from $P(t)$ randomly, 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 these chromosomes are candidates to be evaluated in each generation. This fact is important in our problem because of the evaluation function complexity.

Stop criterion. The algorithm stops when a fixed number of iterations (N) has been executed. Then, the probability of the K MPEs found is divided by $p(x_O)$, in order to obtain $p(x_E|x_O)$.

If the population size is noted by S, then $S + S \cdot N/2$ is an upper bound of the total number of evaluated chromosomes (probabilistic propagations). We say that this amount is an upper bound, because due to the evaluation function cost, a record of (previously) evaluated chromosomes is maintained in order to avoid repeated computations.

4. Experimental evaluation

We have used [1] the *alarm* belief network (Beinlich et al., 1989) to evaluate our algorithm. This network has been used often in the literature to test several kinds of algorithms (learning, propagation, etc.). The alarm belief network (Fig. 2) has 37 variables, each of them can take 2, 3 or 4 different states.

We have carried out three experiments. In each experiment three or four variables were selected (randomly) as evidence (x_O), and the $K = 50$ most probable explanations were calculated using (1) an exact algorithm and (2) our genetic algorithm. Table 1 shows some data about the experiments.

At the end of each experiment several statistics were obtained:

- top1. – Percentage of runs in which the best MPE was obtained.
- top10. – Percentage (on average) of MPEs found between the second and the 10th.

For example, top10 = 33.3% means that three explanations in the range 2..10 were found. However, the positions are not considered, so (2,3,6) or (7,8,9) yield the same value for top10.

- top25. – Similar to top10 but with the range 11..25.

[1] In (Gámez, 1998) experiments with other networks have been carried out.

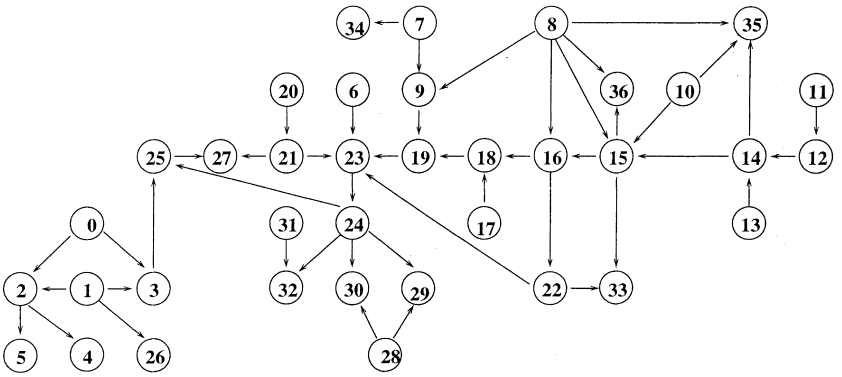

Fig. 2. The *alarm* belief network.

Table 1
Description of the experiments[a]

Experiment	$\lvert X_E \rvert$	X_E	$\lvert \Omega_{X_E} \rvert$
1	12	Roots	9.216
2	12	Randomly	248.832
3	15	Randomly	4.478.976

- top50. – Similar to top10 but with the range 26..50.
- ind. – Percentage of distinct chromosomes evaluated with respect to the search space cardinality.
- *mass*1. – If p_1 is the probability of the real true MPE, and $\hat{p}_1$ is the probability of the best MPE found by the algorithm, then $mass1 = (\hat{p}_1 \cdot 100)/p_1$.
- *massT*. – Similar to mass1, but taking $p_k = \sum_{i=1..K} p_i$ and $\hat{p}_k = \sum_{i=1..K} \hat{p}_i$.

Tables 2–4 show the results calculated over 100 runs. The population size is 125 for the first experiment, 300 for the second and 500 for the third. The number of generations is noted by *ng*.

The use of the proposed precomputation operator has obtained a significant reduction in the size of the join-tree (27%, 20.4% and 15%, respectively), so the evaluation of the chromosomes has been faster.

5. Discussion

As Gelsema (1995) points out, finding the K MPEs in a BBN is not a trivial matter. The complexity of the problem increases in the partial case (Gámez, 1998), because of the large increment in the size of the junction tree (used for exact propagation).

In Tables 2–4 we can see that the genetic algorithm proposed here is able to find the best explanation in a highest number of cases, and also

Table 2
Results for the experiment 1

ng	top1	top10	top25	top50	ind	mass1	massT
2	96.00	52.11	46.07	29.00	1.10	99.54	57.59
4	98.00	66.44	60.00	40.32	2.17	99.97	71.44
6	100.00	75.00	69.73	49.32	2.17	100.00	79.32
8	100.00	82.89	75.47	57.20	2.37	100.00	85.04
10	100.00	89.33	80.27	62.68	3.15	100.00	89.75
12	100.00	93.11	83.40	67.84	3.26	100.00	92.55
14	100.00	95.44	86.20	72.52	3.27	100.00	94.59
16	100.00	97.11	88.53	76.40	3.48	100.00	95.85
18	100.00	98.67	90.20	79.80	4.04	100.00	96.96
20	100.00	99.11	91.13	82.80	4.33	100.00	97.48

Table 3
Results for the experiment 2

ng	top1	top10	top25	top50	ind	mass1	massT
3	81.00	47.56	42.87	30.88	0.23	92.37	56.49
6	90.00	66.89	53.80	41.72	0.31	96.34	70.50
9	96.00	82.56	63.80	51.48	0.37	98.88	81.01
12	98.00	91.22	72.13	62.16	0.44	99.31	87.68
15	100.00	96.67	78.00	69.52	0.50	100.00	91.77
18	100.00	99.11	82.47	75.40	0.56	100.00	94.00
21	100.00	99.44	86.67	80.80	0.61	100.00	95.54
24	100.00	99.78	89.07	84.60	0.67	100.00	96.46
27	100.00	99.89	91.33	87.36	0.72	100.00	97.21
30	100.00	99.89	92.80	90.24	0.77	100.00	97.74

Table 4
Results for the experiment 3

ng	top1	top10	top25	top50	ind	mass1	massT
10	100.00	47.44	50.13	44.76	0.04	100.0	65.28
20	100.00	68.56	63.53	58.60	0.06	100.0	79.34
30	100.00	83.67	75.53	71.52	0.08	100.0	89.19
40	100.00	93.56	84.60	81.96	0.10	100.0	94.93
50	100.00	97.78	91.20	89.04	0.12	100.0	97.78
60	100.00	99.44	94.60	94.36	0.14	100.0	98.99
70	100.00	99.78	97.47	97.16	0.15	100.0	99.53
80	100.00	100.00	98.53	98.68	0.17	100.0	99.77
90	100.00	100.00	99.53	99.44	0.18	100.0	99.92
100	100.00	100.00	99.80	99.72	0.20	100.0	99.96

the rest of the K MPEs. The population size and the number of generations have been selected in function of each experiment, and they ensure a good time of response. Furthermore, we can see that if we stop the algorithm before it finishes, the obtained solutions are also of high quality,

In comparison with related works, only Gelsema (1995) has studied this problem using genetic algorithms. The algorithm proposed in (Gelsema, 1995) was developed to solve the problem of "total" abductive inference, however, the task to approximate the best set of explanations is attempted by integrating the appropriate members occurring in the final population of the genetic algorithm. By experimentation we have seen that using the same time of computation our algorithm improves the results obtained by this method. We have realized that using Gelsema's method the explanations rarely are ranked in correct order. For example, in the second experiment, the final population obtained by this method contains the best explanation in 42% of times, but this explanation is never classified as the best. With respect to the rest of MPEs, only 30% of them (on average) were presented in the final population (and most of them were not ranked in the correct order). We think that this is a logical consequence of our algorithm as it is explicitly designed to solve the partial abductive problem, while the method described in (Gelsema, 1995) was designed to solve total abduction.

6. Conclusions

We have presented a genetic algorithm for partial abductive reasoning in BBNs. The algorithm has an *anytime* performance, i.e., it can continue the search while time is available or can be stopped at any time to yield the K MPEs found until this moment. From the experimental results, we can conclude that the genetic algorithm

presented here is a good tool to perform partial abductive reasoning in complex belief networks, where exact computation is not possible.

The algorithm is approximate in the sense that we cannot be sure that the optimum has been really found, but as the evaluation function used here is an exact propagation, we can be sure that if explanation e_1 is ranked before explanation e_2, then e_1 is more probable than e_2.

From the experiments we have learnt that using the precomputation operator would be always recommended, because the evaluation of the chromosomes is faster. In future we plan to study new genetic operators and new ways to evaluate the individuals in order to improve the efficiency of the genetic algorithm.

Discussion

Gelsema: You mentioned that in future work you want to experiment with other genetic operators. Why are the operators that you use not sufficient? And what kind of new operators are you thinking of?

Gámez: The problem in this genetic algorithm is that to evaluate a chromosome, it is necessary to perform a complete upward propagation in the junction tree. Our idea is to define crossover and mutation operators over the junction tree. In this way, to evaluate a new chromosome, only a partial propagation in the junction tree is necessary and not a complete one, as in the present algorithm.

References

Beinlich, I., Suermondt, H., Chavez, R., Cooper, G., 1989. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. Technical Report KSL-88-84, Knowledge Systems Laboratory, Medical Computer Science, Stanford University.

Gámez, J., 1998. Inferencia abductiva en redes causales. Ph.D. Thesis. Departamento de Ciencias de la Computación e I.A. Escuela Técnica Superior de Ingeniería Informática. Universidad de Granada.

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

Henrion, M., 1988. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In: Lemmer, J., Kanal, L. (Eds.), Uncertainty in Artificial Intelligence 2. North-Holland, Amsterdam, pp. 149–163.

Jensen, F., 1996. An introduction to Bayesian Networks. UCL Press.

Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin.

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

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Mateo.

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

Shimony, S., 1994. Finding maps for belief networks is NP-hard. Artificial Intelligence 68, 399–410.