ABDUCTIVE REASONING IN BAYESIAN BELIEF NETWORKS USING A GENETIC ALGORITHM

E.S. Gelsema, Department of Medical Informatics, Erasmus University,
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands

1. Introduction

Bayesian belief networks (causal networks) have been extensively studied in the past ten years. It has been shown that they provide a sound formalism for probabilistic reasoning, especially if uncertainty is to be represented. A probability space can be modelled as a Bayesian belief network of propositional variables (nodes) which may be pairwise connected by directed arcs. The interpretation is that if an arc exists from node A to node B, the probability of node B assuming a given state b_i depends on the actual state of node A (A is a direct cause of B). The absence of an arc between two nodes implies that there is no such direct dependence. Thus, in a Bayesian belief network, probabilistic dependencies are modelled as arcs between nodes, independencies are implied by the absence of arcs. If for a given probability space, for all states of the root nodes the prior probabilities are known, and in addition, for all non-root nodes the conditional probabilities, given the parent states, the joint probability distribution is completely known. Textbooks on Bayesian belief networks are [1] and [2].

A Bayesian belief network for which no variables are instantiated is in its prior probability state. Since the joint probability distribution is completely defined by the network, the probabilities of all states of all variables may be calculated. The same is true in situations where one or more variables are instantiated to given values. Such computations are known as probability propagation. The complexity of probability propagation depends on the structure of the network: If the graph representing the network is a tree, probability propagation is fairly straightforward. If the graph is singly connected, meaning that there is at most one chain between any pair of variables (a chain is a connection over undirected arcs), the complexity increases only moderately. If the graph is not singly connected, however, the complexity increases dramatically. It has been shown that in general in such networks probability propagation is NP-hard [3].

A special class of problems in Bayesian belief networks is abductive reasoning. Abductive reasoning is inferencing from effects to their best explanations. In the context of Bayesian belief networks this corresponds to finding the maximum a posteriori probability (MAP) instantiation of all the nodes, given the instantiated nodes. Whereas probability propagation gives the conditional probabilities of all individual variables given that some (or none) variables are instantiated, in general it does not give an answer to the question of the state of the network with the highest overall probability. It has been shown that abductive reasoning in non-singly connected Bayesian belief networks is also NP-hard [4].

Since probability propagation and abductive reasoning in non-singly connected Bayesian belief networks is NP-hard, much research has gone into the possibilities of obtaining approximate solutions. The purpose of this paper is to suggest such an approximate solution to abductive reasoning, using a genetic algorithm. In section 2, the mapping of the problem of abductive inference onto a genetic algorithm will be described. Section 3 describes a computational exercise designed to verify whether genetic algorithms can be used for the purpose of abductive reasoning and to establish the efficiency of such an inference scheme. In section 4, results are presented; Finally, the results obtained are discussed in section 5.

2. Abductive reasoning as a genetic algorithm problem.

Genetic algorithms [6] are well known for their ability to efficiently explore large search spaces. In order to tailor a problem to a genetic algorithm structure, three steps must be taken:

- o The mapping of candidate solutions onto linear chromosomal structures;
- o The definition of genetic operators;
- o The definition of a fitness function describing the quality of candidate solutions.

2.1 Mapping onto chromosomal structures.
Consider the Bayesian belief network of Figure 1:

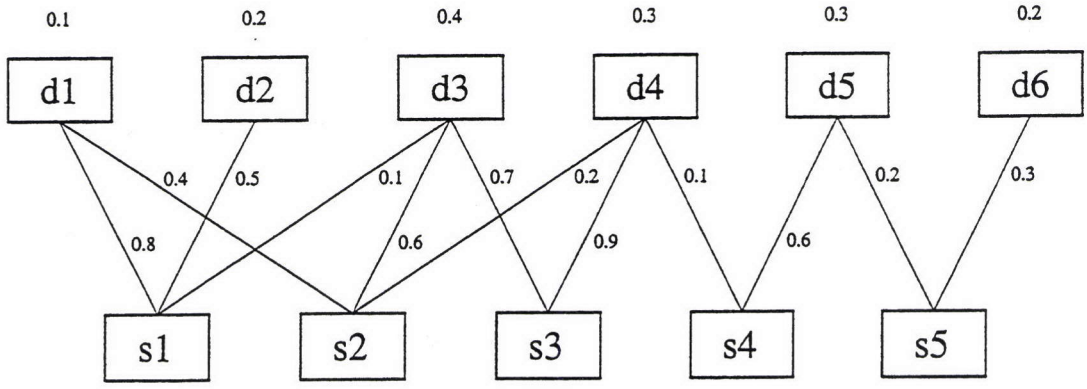

Figure 1: Belief network for the classical diagnostic problem

This network depicts the classical diagnostic problem. In the example, the nodes d_1 through d_6 may be thought of as representing six different diseases, each possibly causing a subset of the five symptoms s_1 through s_5. The structure of the network, i.e. the presence or absence of arcs between pairs of nodes describes the presence or absence of causal relationships. This qualitative, semantic part of the network must be supplemented by quantitative, probabilistic information: the prior probabilities of the states of the root nodes (d_1,...,d_6) and the conditional probabilities of the non-root nodes (s_1,...,s_5) given the states of their parents. It is often difficult to supply estimates of these latter conditional probabilities. However, they may be computed from the probabilities P ($d_i \rightarrow s_j | +d_i$) of the causation events $d_i \rightarrow s_j$ for all relevant combinations of i and j. The causation event $d_i \rightarrow s_j$ as introduced by Peng and Reggia [5] is the event that d_i is the cause of s_j, given that both are present. See also Neapolitan [2]. It must be noted, however, that in more complicated networks it may no longer be possible to obtain the required conditional probabilities from the probabilities of the causation events.

In the example above, each of the nodes may be in exactly one of two states: state = 0 (absent) or state = 1 (present). Although this is a restriction in this example, the genetic algorithmic method proposed here is not restricted to such networks. Increasing the number of possible states of one or more nodes merely increases the complexity of the search space.

It is now easy to map the state of the network onto a chromosomal structure. Assigning each of the nodes to a gene with possible values 0 or 1, the state of the network can be expressed by a string of 11 bits:

d_1	d_2	d_3	d_4	d_5	d_6	s_1	s_2	s_3	s_4	s_5
0	1	1	0	0	0	1	0	0	0	0

The string above expresses the state of the network in which diseases d_2 and d_3 are present in

combination and s_1 is the only symptom present.

2.2 Definition of genetic operators

In genetic algorithms, recombination is the process by which individuals from one generation (as expressed by their chromosomes) are combined to produce offspring for the next generation. In addition to recombination, mutation also plays a role in the creation of a new generation. The genetic operators contain the rules governing the processes of recombination and mutation.

- o recombination:
  The most widely used recombination operator is the two point cross-over operator: on the chromosomes of two parent individulas, two cut points are randomly selected (in both parents at the same location) and the chromosomal material between the cut-points is swapped, producing two offspring individuals.

0 1 0 1 0 0 1 1 1 0 1	**0 1 1 1 1 0 1 1 1 0 1**
↕ ↕	
1 0 1 1 1 0 1 0 1 0 1	1 0 **0 1 0** 0 1 0 1 0 1
two parents	two offspring individuals

- o mutation:
  Whenever offspring individuals are created, their chromosomes may be altered by mutation. Mutation is the process by which a gene is altered to another of its possible states. In this example, mutation changes a gene to a 0 if it is a 1, or vice versa. Mutations are usually invoked with a small probability.

2.3 Definition of a fitness function

In genetic algorithms a fitness function must be specified , which assigns a fitness value to each individual in the current population. The fitness expresses the quality of the individual according to some appropriate criterion. The fitness values are used to favour high-fitness individuals over low-fitness individuals to take part in the process of reproduction.

In the current application, an individual represents a possible state of the entire network. For the purpose of abductive reasoning, the most logical choice for a fitness function is the overall probability associated with the state of the network as represented by the individual. This overall probability is straightforwardly calculated as a product of n multipliers, one for each of the n nodes in the network. The multiplier is the prior probability for a root node and the applicable conditional probability for a non-root node.

In Fig. 1, the prior probabilities of the nodes $d_1,...d_6$ being in state present are given in the top row. The numbers associated with the arcs d_is_j represent the probabilities $P(d_i \rightarrow s_j|+d_i)$ of the causation events $(d_i \rightarrow s_j)$. Absence of an arc from d_i to s_j implies that the presence or absence of d_i has no influence on the state of node s_j. As mentioned above, all required conditional probabilities of the symptom nodes may be derived from the causation event probabilities.

3. The abductive reasoning exercise

The goals of abductive reasoning in Bayesian belief networks may be one of the following:

- o Given a state of instantiation of the network, find the state(s) with the highest overall a posteriori probability.
- o Given a state of instantiation of the network, find the most probable explanation(s).

These goals differ in an important way. An explanation is usually defined as a certain

configuration of root node states, irrespective of the states of the non-root nodes, except the instantiated ones. The probability of an explanation is calculated as the sum of the probabilities of all states with the given configuration of root node states.
The fitness function as defined above allows to rank all states in a population according to overall probability. If the Bayesian belief network is in its prior probability state, the corresponding overall probabilities can be computed. If the network is in a state with one or more nodes instantiated, the overall probabilities can be calculated only up to a proportionality factor, thus still allowing to rank the solutions. Finding a ranking of the most probable explanations is more problematic but it is also attempted in the exercise described below.

The execution of a genetic algorithm consists of the following steps:

- o generation of an initial population;
- o transition from one generation to the next;
- o iteration control.

3.1 Generation of the initial population
An initial population was generated as follows: For each individual to be generated, the genes corresponding to instantiated nodes in the network were given their corresponding values. Then, for each gene corresponding to a non-instantiated root node, a value of 0 or 1 was randomly generated with equal probability. Finally, for each gene corresponding to a non-instantiated, non-root node, a value of 0 or 1 was generated with probability equal to the corresponding conditional probabilities. This procedure ensures that the exploration of the search space starts in promising regions. In the experiments, population sizes of 50 and 25 were used.

3.2 Transition from one generation to the next
All individuals in each generation are evaluated on the basis of the fitness function as defined in section 2. The best n different individuals are preserved for the next generation, where n is a parameter of the procedure, in all experiments described here set to n = 5. If the total population size is fixed at 50 for all generations, 45 new individuals are created by the process of cross-over and mutation as explained in Section 2. Genes corresponding to instantiated nodes must preserve their assigned values throughout the generations. The process of cross-over does not alter the values of such genes, since they have the same value in both parents selected for reproduction. However, genes corresponding to instantiated nodes are excluded from the process of mutation.
Parent selection for cross-over is done as follows: based on the value of the fitness function, all individuals are assigned a rank order. The rank orders are used in a roulette wheel selection procedure [6].

While the process described above yields, in each generation, the five best different solutions, establishing the best explanations is more complicated. Since an explanation is a set of states of the network with the same root node configuration, the merit of each explanation in each generation was approximated by adding the fitnesses of all different members encountered in that generation. Since with increasing generation number, the fittest members are expected to dominate, this procedure is expected to give a good estimate of the probabilities of the best explanations.

3.3 Iteration control
In order to control the process of reproduction, the sum of all fitnesses of different individuals in each generation was calculated. The process of reproduction was terminated when from one generation to the next, the total fitness did not change more than 1 %. The best five solutions and the best five explanations were then recorded as the final result.
For each experiment described below, the genetic algorithm was executed 50 times.

The experiment was performed in the following three situations:

- the network in its prior probability state with a population size of 50;
- the network in a state with node s_1 instantiated to "present" with a population size of 50;
- the network in a state with nodes s_1 and s_2 instantiated to "present" with population size of 25.

The total search space has a cardinality of 2048, 1024, and 512, respectively in these three situations.

4. Results

At the end of each run in each experiment, the best five solutions and the best five explanations were recorded. In the tables below, the best five solutions and the best five explanations encountered in experiment 1 after 50 complete runs are recorded. For each of these, it is indicated in which fraction of the total number of 50 runs it appeared as number 1, 2, ..., 5. Moreover, having identified the overall best five solutions and explanations, for each run it was counted how many of these appeared in the group of five best for that run. Tables listing the fractions of runs with ≥1, ≥2,..., =5 overall best in the group of five best are also given. Corresponding statistics were collected for experiments 2 and 3.

5. Discussion and conclusions

Given a state of instantiation of a Bayesian network, finding the set of n best solutions, i.e. the n solutions with the highest overall probability, for $n > 2$ is not a trivial matter. The results presented in the previous section indicate that the method using a genetic algorithm in most cases yields highly probable solutions. A comparison of the efficiency of the method with random sampling will be given below.

Solutions:
In experiment 1, with no nodes instantiated, the search space has a cardinality of $2^{11} = 2048$. The average number of generations required up to convergence is 6.54. In 6.54 generations, a number of 50 + 5.54 x 45 = 300 solutions (not necessarily different ones) has been explored. In 90% of the 50 runs, the best true solution is found. One may calculate the probability of finding the best, the two best, etc. solutions if one were to randomly sample the search space, taking 300 samples. Comparing these probabilities with the probabilities as found in the experiment, one may then calculate how many of such random sampling experiments (each time sampling 300 solutions) are required to reach the probabilities as found using the genetic algorithm. This number will be called the efficiency of the method. The efficiencies which depend on the number of best solutions to be found, for all three experiments are listed in the table below.

goal	Exp. 1	Exp. 2	Exp. 3
find the best solution	17	7	6
find 2 best solutions	60	19	7
find 3 best solutions	255	82	36
find 4 best solutions	982	187	172
find 5 best solutions	3324	450	334

It appears that with increasing complexity of the search space, the efficiency of the genetic algorithm as compared to random sampling increases. Also, the efficiency increases with increasing complexity of the goal.

Details of the results for Experiment 1:

Nodes instantiated:	none
Cardinality search space:	2048
Population size:	50
Number of runs:	50

Average number of generations: 6.54 (range 4-10)

Solutions found:

best solution	prob.	fraction				
		#1	#2	#3	#4	#5
00000000000	0.169	0.90	-	-	-	-
00010000100	0.047	0.04	0.62	-	-	-
00100001100	0.042	0.04	0.24	0.40	-	-
00001000010	0.034	0.00	0.08	0.12	0.22	-
00000100000	0.029	0.02	0.02	0.32	0.14	0.10

Fraction of runs with ≥1, ≥2, ≥3, ≥4, =5 of the overall best solutions amongst the five best found in that run.

≥1	≥2	≥3	≥4	=5
1.00	0.96	0.84	0.36	0.10

Explanations found:

best explanation	prob	fraction				
		#1	#2	#3	#4	#5
000000	0.169	0.90	0.00	0.00	0.00	0.00
001000	0.113	0.04	0.56	0.16	0.10	0.04
000100	0.069	0.00	0.28	0.36	0.02	0.02
000010	0.067	0.00	0.10	0.12	0.20	0.20
001100	0.044	0.00	0.02	0.12	0.30	0.18

Fraction of runs with ≥1, ≥2, ≥3, ≥4, =5 of the overall best explanations amongst the five best found in that run.

≥1	≥2	≥3	≥4	=5
1.00	1.00	0.96	0.66	0.16

Explanations:
In a medical diagnostic problem, one may be primarily interested in the best explanation, regardless of the states of the unobserved symptoms. However, when presented with a set of best solutions, one may extract information on the symptoms most critical for discrimination and try to establish the presence/absence of these. The algorithm may then be rerun in a different state of instantiation.
In Experiment 1 and 2, for at least 96% of the runs, the set of 5 best explanations found contains at least 3 of the 5 overall best. This result no longer holds for Experiment 3. This is probably a result of the reduced populatoin size, even though the search space is correspondingly less complex.

One may again compare the efficiency of finding the n best explanations with random sampling. It should be noted that, although the total number of possible explanations is much smaller than the number of possible solutions, the search space is not correspondingly less complex: in order to establish the merit of an explanation, the space of all possible solutions has to be searched. Efficiencies calculated from the probabilities of finding explanations are comparable to those found for solutions.

In this contribution, it has been shown that genetic algorithms can be used for the purpose of abductive reasoning in Bayesian belief networks. In the example given, good quality solutions are frequently found. The efficiency of the algorithm compares favourably with random sampling, especially in the higher complexity regions. This justifies the expectation that the results will scale to larger networks, although this remains to be shown.
It is a well-known fact that the performance of genetic algorithms is strongly dependent on the parameters (population size, selection strategy, convergence criterion). No attempt was made to carefully tune these. Also, the variant of a static genetic algorithm, in which the population is updated after each single recombination may prove to perform better than the generation variant used here. These aspects of scaling and tuning are the subject of future research.
It should also be mentioned that although the method has been illustrated for the case of the classical diagnostic problem, it can equally well be applied to more complex belief networks. Also, the methodology used here can be easily modified to be applicable to influence diagrams. In that case, the utility function is a natural choice to play the role of fitness function.

References.

1. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Morgan Kaufmann Publ. Inc., San Mateo (CA), 1988.
2. R.E. Neapolitan, *Probabilistic Reasoning in Expert Systems. Theory and Algorithms,* John Wiley & Sons, Inc., New York, 1989.
3. G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, *Artif. Intell.* 42 (1990) 393-405.
4. S.E. Shimony, Finding MAPs for belief networks is NP-hard, *Artif. Intell.* 68 (1994) 399-410.
5. Y. Peng and J.A. Reggia, A probabilistic causal model for diagnostic problem solving - Parts I and II, *IEEE Trans. Syst., Man, Cybern.* SMC-17 (1987) pages?
6. L. Davis, *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, New York, 1991.