Importance Sampling Algorithms for Belief Networks based on Approximate Computation

L.D. Hernández
Dept. Informatics and Systems
University of Murcia
30071-Espinardo-Murcia-SPAIN
e-mail: ldaniel@dif.um.es

S. Moral
Dept. Computer Sciences and A.I.
University of Granada
18071-Granada-SPAIN
e-mail: smc@robinson.ugr.es

A. Salmerón
Dept. Statistics and A.M.
University of Almería
04120-Almería-SPAIN
e-mail: asc@stat.ualm.es

Abstract

In this paper we study a new general class of algorithms for the propagation of probabilities on graphical structures based on importance sampling techniques. The idea is to make an approximate and fast propagation in order to obtain a sampling distribution as close as possible to the true one. Our proposal is based on a deletion sequence of the variables to calculate the 'a posteriori' probability in one variable. The deletion procedure is the basis for the exact propagation algorithms. Here the difference is that sometimes, when the cost of the exact deletion exceeds a given limit, an approximated deletion is done. The calculations of the deletion procedure will be used to obtain in a very fast way a sample for the simulation. Some experimental tests are carried out to compare our procedure with other known methods.

1 INTRODUCTION

Probability propagation in belief networks consists on updating the probability values of the variables in a dependence graph, given some variables that have been observed.

Different exact methods have been developed for this purpose in the last years [11, 13, 15, 16, 17, 20]. In general, it can be said that they take advantage of conditional independences among variables expressed by the graph, to develope a local propagation algorithm. Shachter, Andersen and Szolovits showed how these methods can be included inside a global framework based on the concept of cluster tree [19]. The problem of these algorithms is that the exact propagation probelm is NP-hard in the worst case [4]. This fact makes necessary the use of approximate algorithms to be able to deal with a larger class of problems. Most of these approximate algorithms try to obtain a good estimation of the ditribution in the network applying Monte Carlo techniques. Approximate inference is a NP-hard problem too [5], but the class of solvable problems is wider.

Two different classes of Monte Carlo algorithms can be found in the literature about belief networks: those based on importance sampling and those based on Markov chains. The problem to solve is the same: How to obtain samples from a difficult to manage probability distribution. Importance sampling algorithms use a modified distribution in order to obtain independent samples, that are weighted to resemble the original distribution. The first algorithm of this class, called Probabilistic Logic Sampling, was proposed by Henrion [8]. It works well when no evidences are given. The Likelihood Weighting algorithm, proposed by Fung and Chang [7] and Shachter and Peot [18] improves Henrion's one. This algorithm has a good performance, but the same problem as in logic sampling can arise [2]: One can imagine very simple examples in which all the weights are 0 or 1. Of special interest into this group is the class of algorithms developed by Cano, Hernández and Moral, based on entropy criteria [2]. They use the functions with less entropy (the most informative ones) to simulate the variables, and those with more entropy to weight the simulation (this solves the problem of 0-1 weights). The other class of Monte Carlo algorithms is the so called Markov Chain-Monte Carlo algorithms. In this case the samples are not independent, but they verify the Markov property. Pearl's stochastic simulation [12] is an example of this technique. One generalized approach to stochastic simulation was given by Jensen, Kong and Kjærulff [10], allowing samples to be generated with a greater degree of independence, but with a higher computational cost.

In this paper we consider a general class of importance sampling algorithms. The basic idea is to make a previous approximated propagation, following the concept of node removal [20], very similar to the symbolic propagation due to D'Ambrosio [17] (section 2), and then improve it by sampling the obtained functions. Node removal is done in two steps: firstly, combining the functions associated with the node to be removed, and secondly, marginalizing the resulting function deleting such variable. The main algorithm is presented in section 2.4. Some variations, defining criteria about the functions that must be combined before doing the marginalization are considered (section 2.5). The paper ends with an experimental evaluation of the algorithms (section 3) comparing them with Likelihood Weighting, and conclusions (sect. 4).

2 THE ALGORITHMS

2.1 THE PROBLEM

A belief network is a directed acyclic graph where each node represents a random variable. Given the independences associated to the graph a probability distribution is specified by giving a distribution for each node conditioned to its parents.

Let $X = (X_1, \ldots, X_n)$ be the set of variables in the network. Each variable X_i takes values on a finite set U_i. We shall denote by U_I the cartesian product $\prod_{i \in I} U_i$. Given $x \in U_I$ and $J \subseteq I$, we shall denote by $x^{\downarrow J}$ the element from U_J obtained from x dropping the coordinates not in J. Given a function f defined over U_I, $s(f)$ will denote the set of indexes of the variables for which f is defined (i.e. $s(f) = I$). Under these conditions, the conditional distribution of X_i given its parents in the net, $F(i)$, is denoted by

$$f_i(x) = f_i(x^{\downarrow i}, x^{\downarrow F(i)}) \quad \forall i \in N \quad \forall x \in U_{s(f_i)} \tag{1}$$

where $N = \{1, \ldots, n\}$, and

$$\sum_{x_i \in U_i} f_i(x_i, x) = 1 \quad \forall x \in U_{F(i)}$$

Then, the joint probability distribution for the n-dimensional random variable X can be calculated as

$$p(x) = \prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \quad \forall x \in U_N \tag{2}$$

When the value of a variable X_i is known, X_i is called an *observation*, and its value e_i is called *evidence*. The set of evidences will be denoted by e and the set of indexes of the observed variables is denoted by E. One observation X_i has associated a Dirac function defined on U_i by

$$\delta_{e_i}(x) = \begin{cases} 1 & \text{if} \quad e_i = x \\ 0 & \text{if} \quad e_i \neq x \end{cases} \tag{3}$$

Assume that the joint distribution (2) is difficult to manage and that we want to calculate the *'a posteriori'* probability function $p(x|e)$, for every $x \in U_I$, where $I \subseteq N$. This probability is equal to $p(x \cap e)/p(e)$, and as $p(e)$ is constant, it is proportional to $p(x \cap e)$. So, we can know the *'a posteriori'* probability if we calculate for every $x \in U_I$ the value $p(x \cap e)$, normalizing afterwards. $p(x \cap e)$ can be expressed in the following way,

$$p(x \cap e) = \sum_{\substack{y^{\downarrow E} = e \\ y^{\downarrow I} = x}} \prod_{i \in N} f_i(y^{\downarrow s(f_i)})$$

$$= \sum_{y^{\downarrow I} = x} \left(\prod_{i \in N} f_i(y^{\downarrow s(f_i)}) \right) \cdot \left(\prod_{j \in E} \delta_{e_j}(y^{\downarrow j}) \right) \tag{4}$$

2.2 IMPORTANCE SAMPLING

A well known approximate method to calculate the addition in (4) is the importance sampling technique [14]. It is a Monte Carlo procedure that uses an auxiliary probability distribution P^* to obtain a sample from the space U_N. Afterwards, a weight is assigned to each obtained configuration, $x^{(i)} \in U_N$. This weight is,

$$w_i = \frac{(\prod_{i \in N} f_i(x^{\downarrow s(f_i)})).(\prod_{j \in E} \delta_{e_j}(x^{\downarrow j}))}{P^*(x)} \tag{5}$$

The only condition for the sampling distribution is that if $p(x^{\downarrow I} \cap e) > 0$, then $P^*(x) > 0$. If this is verified and $(x^{(1)}, \ldots, x^{(m)})$ is the sample we have obtained, then an unbiased estimation of $p(x \cap e)$ can be obtained by calculating,

$$\frac{\sum_{x^{(i)\downarrow I} = x} w_i}{m} \tag{6}$$

The optimal selection for P^* is equal to $P(.|e)$. That is $P^*(x)$ should be proportional to $p(x \cap e)$. In this way we obtain constant weights and the variance of the estimation is minimal [2, 9, 18]. But this selection is not viable in general, because p is a probability difficult to handle. What we should do is to select P^* in such a way that we obtain weights which are *as constant as possible*, and this is achieved if P^* is very similar to $P(.|e)$. The algorithm is as follows (see [2, 18]):

1. for $i = 1$ to m

(a) Generate a configuration $x^{(i)}$ according to P^*.

(b) Do

$$w_i = \frac{\prod_{i \in N} f_i(x^{\downarrow i}, x^{\downarrow F(i)})}{P^*(x)} \times \prod_{j \in E} \delta_{e_j}(x^{\downarrow j}) \quad (7)$$

2. For every $x \in U_I$

   (c) Estimate $p(x \cap e)$ by using formula (6)

3. Normalize values $p(x \cap e)$

2.3 TECHNICAL DEFINITIONS

Given a function f over a set of variables X_I and $J \subseteq I$, we define *marginalization* of f over a variable X_J (or the *deletion* of variables in $I - J$) as a new function $f^{\downarrow J}$ defined over the set J given by the expression

$$f^{\downarrow J}(x) = \sum_{\substack{y \in U_I \\ y^{\downarrow J} = x}} f(y) \quad \forall x \in U_J \quad (8)$$

Given r functions $f_1, \ldots, f_r$ each one defined over the sets $I_1, \ldots, I_r$, a new function, called the *combination* of them, is defined over the set $I = \bigcup_{i=1}^r I_i$ as

$$f(x) = \bigotimes_{i=1}^{r} f_i(x^{\downarrow I_i}) = \prod_{i=1}^{r} f_i(x^{\downarrow I_i}) \quad \forall x \in U_I \quad (9)$$

Note that the product above can be done in any ordering. Thus, combination is a commutative operator.

Let f be a function defined over a set I of indexes, and assume a set of variables X_J, $J \subset I$ whose values $x^{\downarrow J}$ are fixed ($x^{\downarrow J} = x_0$). The *restriction* of f to the values x_0 is a new function f' defined on $I - J$ according to the following expression:

$$f'(x) = f(y) \quad (10)$$

such that $y \in U_I$, $y^{\downarrow I-J} = x^{\downarrow I-J}$ and $y^{\downarrow J} = x_0$.

Reduction can simplify the problem when we have observations. Note as formula (4) is still valid if we replace each function f_i by its reduction to the evidence e. This way we can work with simpler functions.

2.4 THE MAIN ALGORITHM

The key point in an importance sampling algorithm is to obtain sampling distributions as similar as possible to the original ones. In a belief network, the original distribution is presented as a product of conditional distributions. Thus, algorithms proposed by Shachter and Peot [18], Cano, Hernández and Moral [2] and Fung and Chang [7] use sampling distributions close to the original conditional ones, in order to obtain more uniform weights in the simulation process.

Here it is proposed the idea of using, in each moment, all the information available about each variable, that is, use all the functions in which a variable takes part in order to sample it. This, obviously, will not be always possible, because the complexity of this process would be the same of the exact computation of the probabilities in the network. However, defining criteria about combination of functions, this operation can be done in an approximate way, so that calculation will be faster. Then, weighting the obtained samples correctly, an importance sampling algorithm that uses sampling functions very close to the original ones is obtained. More concretely, the algorithm starts with a family of functions given by the original set of conditional probabilities and the observations,

$$H = \{f_1, \ldots, f_n\} \cup \{\delta_{e_l}\}_{l \in E} \quad (11)$$

Then it considers an ordering of the variables given by a permutation σ on the set $\{1, \ldots, n\}$ and proceeds by *deleting* the variables on the order given by σ. The *deletion* of a variable $X_{\sigma(i)}$ should be done in the following way (**Exact**):

– It combines all the functions which are defined for variable $X_{\sigma(i)}$, obtaining a function h_i. It deletes $X_{\sigma(i)}$ from the combination, h_i, by marginalizing the result to $s(h) - \{\sigma(i)\}$. The result is added to H. All the functions which were combined to obtain h_i are removed from H.

If we are able to delete all the variables by repeating this step from 1 to n, then there is no difficulty in sampling with a probability proportional to $p(x \cap e)$. In fact what we are doing is an exact propagation algorithm [20], and the following facts can be proved:

– If h_n is the function obtained when we are deleting $X_{\sigma(n)}$ then for all $x \in U_{\sigma(n)}$, $h_n(x)$ is proportional to $p(x|e)$.

– If h_i is the function obtained when we are deleting $X_{\sigma(i)}$ ($i < n$), $\Sigma(i) = \{\sigma(i+1), \ldots, \sigma(n)\}$, and $x_0 \in U_{\Sigma(i) \cap s(h_i)}$, then the restriction of h_i to x_0, h'_i (see equation (10)) is proportional to the probability $p(.|e, x_0)$: $\forall x \in U_{\sigma(i)}, h'_i(x) \propto p(x|e, x_0)$.

These two properties allow us to simulate a value $x \in U_N$ with a probability equal to $p(x|e)$. We only have to obtain values for the variables in the order, $X_{\sigma(n)}, \ldots, X_{\sigma(1)}$. We only have to use for each $X_{\sigma(i)}$ the function h_i, making the restriction of it to the values already obtained for the other variables, and normalizing afterwards. The main problem is that not always the deletion of a variable can be done on an exact way. In some cases the size of h_i would be so big that this calculation is unfeasible. In this case this step would have to be approximate. There will be different types of approximated calculations, but in general, they will be particular cases of the following scheme (**Approximate**):

- Let $H(i) = \{h \in H \mid \sigma(i) \in s(h)\}$, the set of functions which are defined for variable $X_{\sigma(i)}$. Remove $H(i)$ from H.
- Transform $H(i)$ by combination. We repeat several times the following process: take $R \subset H(i)$. Combine all the functions in R. Add the combination to $H(i)$. Remove R from $H(i)$.
- Calculate $H^+(i)$ from $H(i)$ by deleting $X_{\sigma(i)}$ of all the functions belonging to $H(i)$. Add $H^+(i)$ to H.

If in the second step of the approximated calculation we combine all the functions in $H(i)$, then we obtain the exact calculation. The idea of the approximated calculation is that when this is not possible, we make only some of the combinations (until a size threshold). This step is not exact, because we should combine all of them, but is an approximation, which has an important property: it does not introduce new 0 values. That is, if $x \in U_N$ is such that $h(x^{\downarrow s(h)}) \neq 0$ for every $h \in H$, before deleting $X_{\sigma(i)}$, this property is verified after the deletion of the variable. If the algorithm is used to sample values for the variables X_N, and we are going to obtain a value for the variable $X_{\sigma(i)}$, the process is as follows:

- Let $H(i)$ the set calculated in step 2 of the deletion procedure.
- Restrict each function in $H(i)$ to the previously obtained values for other variables. Combine all the functions in $H(i)$, obtaining a function h'_i defined on $U_{\sigma(i)}$.
- If $N(h'_i)$ is the normalization of h'_i, obtain a value for $X_{\sigma(i)}$ following the probability distribution $N(h'_i)$.

With all these elements a global view of the algorithm is as follows:

```
1. Let $H = \{f_i \mid i = 1, \ldots, n\}$.
2. Select an ordering $\sigma$ for the variables in $G$.
3. Incorporate observations:
   (a) Restrict all the functions in $H$ to the evidences $\{e_l\}_{l \in E}$ according to equation (10).
   (b) For every observed variable $X_l$, $l \in E$ do
          $H = H \cup \{\delta_{e_l}\}$
4. for $i = 1$ to $n$ do
   (a) Delete $X_{\sigma(i)}$ by the approximated procedure[1]
5. for $j = 1$ to $m$ do
   (a) $w_j = 1.0$
   (b) for $i = n$ downto 1 do
       i. Obtain a value for $X_{\sigma(i)}$, $x_i^{(j)}$, according to $N(h'_i)$.
       ii. Do
              $w_j = \frac{w_j}{N(h'_i)(x_i^{(j)})}$
   (c) Do
          $w_j = w_j \times \prod_{i=1}^{n} f_i(x_i^{(j)}) \times \delta_{e_i}(x_i^{(j)})$
6. Estimate desired probabilities according to equation (6).
```

Note that at the end of loop 5.(b), the weight w_j takes the value

$$w_j = 1/P^*(x_i^{(j)})$$

and after step 5.(c) the resulting weight is the correct one for importance sampling (equation (7)). Observe that the efficiency of the algorithm depends on the ordering of the variables at step 2.

2.5 PARTICULAR CASES

Here we shall study different criteria to select the subsets R of $H(i)$ that will be combined before marginalizing (calculating $H^+(i)$). If we take $R = H(i)$, then we are following the exact deletion algorithm. This procedure, as it was commented early, has not too much interest, because if one has been able to delete all the variables on an exact way, then a propagation will calculate the marginal probabilities for each single variable without simulation, by using only an additional time similar to the time of deleting all the variables. The only situation in which this can be useful is when

[1] remember that the exact calculation is a particular case of this step; so the exact deletion is also possible

we want to know a probability involving several variables, for example a logical combination of events associated to several variables [3]. Then the complexity of the exact procedures increases, while simulation can be addatep to make the calculations keeping the same complexity.

Next criterium we will consider is to do everything approximated without combining. That is, $H(i)$ does not change in the second step of the approximated calculation. This is the faster procedure, because combination costs are avoided. On the contrary, the process is less exact than any other, because we are loosing information in each deletion. Also, simulation time grows because each time a variable is being simulated, it is necessary to combine all the functions for that variable. We call this criterium 1.

Another idea (crit. 2) is to combine all the functions concerning a variable only if the size of the resulting function does not exceed the size of the largest function existing in the original network. That is, we make an exact deletion if the size of the combination is not bigger than the size of the larger potential in the initial graph, and as in criterium 1 otherwise. This way we avoid adding complexity to the original network. By the size of a function we understand the product of the number of cases of all the variables for which the function is defined. This method seems good, but can be improved considering that maybe we cannot combine all the functions, but perhaps we can combine some of them. That is, we select R in the approximated deletion by including functions until the resulting combination does not surpass the limit. In that case, the ordering in which the functions to be combined are selected becomes important. We shall consider two options here: combine all functions in sequence while maximum size is not exceeded (crit. 3) or combine first those functions whose domains are less different, that is, those that are defined over sets of variables with the highest number of coincident variables (crit. 4).

There is another question to establish about the main algorithm: the treatment of observed variables. We have decided to incorporate them before sampling distributions are calculated. This option has the following advantages: first, the functions are restricted to the observed values, so that the size of some functions will be reduced. Moreover, it is avoided distinguishing between observed or not observed variables in simulation time. This fact reduces the possibility of obtaining 0 weights, because the simulation is done with functions already reduced to the observed values. Thus, any sample obtained will be coherent with the observations. An important cause of 0 in existing simulation algorithms is the discordance of the simulated values for the variables and the observations. The troublesome is that the dinamical inclusion of new evidences would produce a new calculation of sampling distributions. The second option is not to restrict the functions to the observations. In this case, the simulation process changes. When one observed variable is to be simulated, then it takes the observed value directly, without simulating. This idea is followed in the Likelihood Weighting algorithm. The troublesome is that we increase the possibility of obtaining 0 weights. The advantage is that evidences can be easily updated.

3 EXPERIMENTAL EVALUATION

In this section we carry out an empirical test of the performance of the algorithms. For this purpose we construct one graph with 40 variables. The structure of this graph consists of both dense and sparse zones. The number of possible values for the variables is between 2 and 4. Three different experiments have been carried out.

In the first one, the conditional distributions have been randomly generated following an uniform distribution. No observations are considered. In the second experiment, the conditional distributions are constructed as in the first one, but four varibles have been instanciated, always to its first value. In the third experiment, four observed variables are considered, and the conditional distributions are randomly generated for all the variables except for one observed variable: the conditional probability of obtaining the observed value given the parents of the variable has been set to 0, except for one configuration of the parents for which it is equal to 1. This way we try to reproduce the problems of Logic Sampling in the Likelihood Weighting algorithm (see [2]). In all the experiments, five algorithms have been compared:

ALG 1	Likelihood Weighting
ALG 2	Importance sampling, criterium 1
ALG 3	Importance sampling, criterium 2
ALG 4	Importance sampling, criterium 3
ALG 5	Importance sampling, criterium 4

The limit of the combinations of functions in R has been set to the size of the larger potential in the original graph. When evidences are given, they are instantiated before obtaining the sampling distributions in importance sampling algortihms (see section 2.5). The ordering considered for the variables is the same they have in the network. The number of runs of the simulation algorithms has been set to 3000, and the run time and error of the estimations have been calculated. For one variable X_l, the goodness of the estimation is measured as [6]:

$$G(X_l) = \sqrt{\frac{1}{|U_l|} \sum_{a_l \in U_l} \frac{(p'(a_l|E) - p(a_l|E))^2}{p(a_l|E)(1 - p(a_l|E))}}$$

where $p(a_l|E)$ is the true *a posteriori* probability, $p'(a_l|E)$ is the estimated value and $|U_l|$ is the number of cases of variable X_l. For a set of variables $(X_i)_{i\in I}$, the goodness of the estimation is:

$$G((X_i)_{i\in I}) = \sqrt{\sum_{i\in I} G(X_i)^2}$$

Each algorithm has been executed 100 times, and the mean time and error have been calculated. Results are shown in tables 1,2, 3, for experiments 1,2 and 3 respectively.

Table 1: Results for experiment 1.

	Time (seconds)	Error
ALG 1	13.62	0.116171
ALG 2	25.66	0.114686
ALG 3	24.43	0.113296
ALG 4	23.95	0.116629
ALG 5	23.56	0.114554

Table 2: Results for experiment 2.

	Time (seconds)	Error
ALG 1	13.84	0.127102
ALG 2	25.55	0.114576
ALG 3	24.37	0.112247
ALG 4	23.05	0.109560
ALG 5	22.11	0.107585

Table 3: Results for experiment 3.

	Time (seconds)	Error
ALG 1	13.8	U
ALG 2	25.75	0.053266
ALG 3	24.63	0.058059
ALG 4	23.51	0.044827
ALG 5	23.21	0.048057

The letter U in table (3) for Likelihood Weighting algorithm means that it has been unable to obtain an estimation of the probabilities because all weights have resulted to be 0. Attending the experimental results, the following can be said:

- The performance of algorithm 1 (Likelihood Weighting) decreases when evidences are given. The extreme case is when values 0 are introduced (experiment 3). In this case no estimation is given because all weights are 0. However, algorithms 2-5 do not have this problem. Estimations for these algorithms are even better for experiment 3. When there are no observed variables (experiment 1) all methods provide similar results.
- There are no substantial diferences among ALG 2-5 for the graph considered, except for the third experiment, where the two last criteria give better results. This was expectable, because in this case making the propagation requieres more precission in the calculations.
- There is no important differences between the different criteria to select R (Algorithms 4 and 5). This does not mean that they will have always the same performance. We are conscious that this experimention is really limited and more extensive tests should be made to asses the differences among the different algorithms.

4 CONCLUSIONS

A new class of algorithms for probability propagation in causal belief networks has been presented in this paper. The new algorithms are more robust than Likelihood Weighting in the general case (when there are observations). With no observations they all have a similar preformance. One important advantage of our procedures is that it is detected when an exact propagation can be done. In this case, we could give exact results.

Many aspects of these algorithms are to be studied in future works. For example, new criteria for selecting what functions are to be combined. The initial ordering of the variables is another point to study. Orderings resulting of graph-triangulation processes can be considered. Moreover, other simulation techniques can be related with those proposed in this paper, as it could be stratified sampling [1]. Now, we are applying stratified sampling to these new algorithms.

References

[1] R.R. Bouckaert, E. Castillo, J.M. Gutiérrez (1995) A modified simulation scheme for inference in bayesian networks. To appear in: *International Journal of Approximate Reasoning.*

[2] J.E. Cano, L.D. Hernández, S. Moral (1995) Importance sampling algorithms for belief networks. To appear in: *International Journal of Approximate Reasoning.*

[3] G.F. Cooper (1989) An algorithm for computing probabilistic propositions. In: *Uncertainty in Artificial Intelligence, 3* (L.N. Kanal, T.S. Levitt, J.F. Lemmer, eds.) North-Holland (Amsterdam) 3-14.

[4] G.F. Cooper (1990) The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42, 393-405.

[5] P. Dagum, M. Luby (1993) Approximating probabilistic inference in Bayesian networks is NP-hard. *Artificial Intelligence* 60, 141-153.

[6] K.W. Fertig, N.R. Mann (1980) An accurate approximation to the sampling distribution of the studentized extreme-valued statistic. *Technometrics* 22, 83-90.

[7] R. Fung, K.C. Chang (1990) Weighting and integrating evidence for stochastic simulation in Bayesian networks. In: *Uncertainty in Artificial Intelligence, 5* (M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer, eds.) North-Holland (Amsterdam) 209-220.

[8] M. Henrion (1988) Propagating uncertainty by logic sampling in Bayes' networks. In: *Uncertainty in Artificial Intelligence, 2* (J.F. Lemmer, L.N. Kanal, eds.) Norht-Holland (Amsterdam) 317-324.

[9] L.D. Hernández, S. Moral (1995) Mixing exact and importance sampling propagation algorithms in dependence graphs. Submitted to: *International Journal of Intelligent Systems.*

[10] C.S. Jensen, A. Kong, U. Kjærulff (1993) Blocking Gibbs sampling in very large probabilistic expert systems. Technical Report R-93-2031. Institute for Electronic Systems, Aalborg University.

[11] S.L. Lauritzen, D.J. Spiegelhalter (1988) Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, B* 50, 157-224.

[12] J. Pearl (1987) Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence* 32, 247-257.

[13] J. Pearl (1988) *Probabilistic Reasoning in Intelligent Systems.* Morgan-Kauffman (San Mateo).

[14] R.Y. Rubinstein (1981) *Simulation and the Monte Carlo Method.* Wiley (New York).

[15] R.D. Shachter (1986) Evaluating influence diagrams. *Operations Research* 34, 871-882.

[16] R.D. Shachter (1988) Probabilistic inference and influence diagrams. *Operations Research* 36, 589-605.

[17] R.D. Shachter, B. D'Ambrosio, B.A. Del Favero (1990) Symbolic probabilistic inference in belief networks. Proceedings of the AAAI'90 Conference, Vol.1, 126-131.

[18] R.D. Shachter, M.A. Peot (1990) Simulation approaches to general probabilistic inference on belief networks. In: *Uncertainty in Artificial Intelligence, 5* (M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer, eds.) North Holland (Amsterdam) 221-231.

[19] R.D. Shachter, S.K. Andersen, P. Szolovits (1991) The equivalence of exact methods for probabilistic inference on belief networks. Submitted to *Artificial Intelligence.*

[20] G. Shafer, P.P. Shenoy (1990) Probability propagation. *Annals of Mathematical and Artificial Intelligence* 2, 327-351.