A MODIFIED SIMULATION SCHEME FOR INFERENCE IN BAYESIAN NETWORKS

Remco R. Bouckaert
Utrecht University,
Netherlands [*]

Enrique Castillo and José Manuel Gutiérrez
University of Cantabria,
Spain [†]

October 4, 1994

ABSTRACT

In this paper we introduce an approximation method for uncertainty propagation which is based on a modification of the stratified simulation. The method uses a deterministic or perfect sample and calculates the number of times simulated instantiations are selected avoiding the repetition of identical instantiations which occurs in the standard stratified simulation method. A theoretical analysis to evaluate the performance of the method as compared with the stratified simulation scheme and to select the required step for the estimation of probabilities with a given error is presented. Some experimental studies compare the proposed with other simulation methods and show a much better performance not only in computer time but also in simulation errors.

Key Words: Bayesian networks, Simulation, Stratified simulation.

1 Introduction

Bayesian belief networks offer a practical methodology for handling uncertainty in knowledge based systems which have a firm theoretical foundation in statistics. One of Bayesian networks main applications is the usage as inference engine; the calculation

[*] Department of Computer Science, P.O.Box 80.089, 3508 TB Utrecht, The Netherlands, remco@cs.ruu.nl

[†] Department of Applied Mathematics and Computational Sciences, Avda. Castros s/n., 39005 Santander, Spain, castillo@fltq.es. and gutierrj@ccucvx.unican.es

of beliefs of events given the observation of other events which is called evidence. For a Bayesian belief network, this task consists of the calculation of the probability of the occurrence of some events given the evidence.

Several algorithms exist for the exact calculation of these probabilities, [14, 11, 15]. However, these algorithms are not generally applicable due to computational limitations. All these algorithms have difficulties with certain types of network structures. This is not surprising since the task has been proven to be NP-hard [3].

A widely used method for handling the computational burden in decision theory and Bayesian statistics is the use of approximation methods. Instead of the exact calculation of probabilities, representative samples of the variables in the Bayesian network can be generated [10, 13, 16]. These so called simulation algorithms have the advantage that the run time is known in advance and that their performance is hardly influenced by the network structure. However, the problem of approximating the probability of an event given the evidence when a given error-bound is required has also been proven to be NP-hard [6]. Another problem with simulation algorithms is their sensitivity to the selection of non-representative samples, a problem that has been recognized for simulation algorithms in Bayesian networks [4].

A well known method for selecting more representative samples in statistics is the use of stratification. The stratified simulation method for Bayesian networks was initially suggested by Bouckaert [2] who presented several variants. These algorithms give more representative samples and, due to the possibility of an efficient implementation, are faster than the previously known simulation algorithms. The most successful version chooses a deterministic sample consisting in equally spaced sample values. Here we present an improved version of his optimal stratification algorithm, which saves computational effort. We analyze theoretically the performance of the algorithm and the influences of the improvement and we make an experimental analysis.

In Section 2, we start with a general description of sampling methods and their application to inference in Bayesian networks. In Section 3, the stratified sampling algorithm is explained in detail and some improvements are introduced. In Section 4, we make a theoretical analysis of the performance of the algorithm and how this is influenced by the imposed improvements. In Section 5, experimental test results are presented and interpreted. And, we conclude with some final considerations and suggestions for further research in Section 6.

2 Sampling

Sampling is a method for the approximation of $\sum_{x \in X} f(x)$ for some function f. Instead of performing the summation over all elements in X, randomly a set of values S, called the *sample*, is chosen. An *instantiation* is an element of a sample. Only for the instantiations in the sample, the value of f is calculated. The obtained values are added and the result is the *score* c. By normalizing the score we obtain $c \cdot |X|/|S|$ (where $|.|$ denotes the cardinality of a set), which is an estimate of $\sum_{x \in X} f(x)$.

The selection of samples of X influences the quality of the approximation considerably; when $f(s)$ is close to zero for almost all instantiations $s \in X$ but very large for only a few, one should have some of the latter instantiations to get a satisfactory score.

A method to select more representative samples is by selecting instantiations s with probability proportional to $f(s)$. The probability with which s is chosen from X is called the *sampling distribution* denoted as P_S. To compensate for the extra selection of instantiations with large values of f, the score is not updated with $f(s)$ but with the weighted value of $f(s)$,

$$f(s)/P_S(s). \tag{1}$$

A *Bayesian belief network* B over a set of variables $V = \{x_1, \ldots, x_n\}$ is a pair (B_S, B_P). B_S is a directed acyclic graph over V, called *network structure.* The parents of a node x_i in the network structure are denoted by π_i. B_P is a set of conditional probabilities, one for each variable in V given its parents, called *assessment functions.* A Bayesian belief network defines a probability distribution [14]

$$P_B(V) = \prod_{i=1}^{n} P(x_i|\pi_i). \tag{2}$$

It is this probability distribution P_B that is used in knowledge based systems. Inference in such knowledge based systems over V consists of the calculations of the marginals of the represented distribution for each variable in V. In other words, inference is the calculation for each $x_i \in V$ of the function $\sum_{x_j \in V \backslash x_i} P_B(V)$, where P_B is the probability distribution defined by the Bayesian network. However, the task is different when evidence is observed, that is, when values of certain variables $E \subset V$ are known to have values e_i for $x_i \in E$. The nodes in E are called *evidence nodes.* With observed evidence, inference is the task of calculating for each $x_i \in V \backslash E$ the probability $P(x_i|E)$, that is, the function

$$\sum_{x_j \in V \backslash E x_i} P_B(V|E), \tag{3}$$

where $P_B(V|E)$ is the probability distribution defined by the Bayes network in which the variables in E are instantiated according to their observed values.

The function (3) can be approximated by sampling, as we just described. Figure 1 shows the general framework of sampling algorithms for inference in Bayesian networks. After a sampling scheme dependent initialization step, m instantiations are generated. An instantiation S is a value assignment to all the variables in $V \backslash E$. A value of Formula (1) is calculated as the quotient of the value of the function that

```
Initialize
for $i \leftarrow 1$ to $m$ do
    $S \leftarrow$ generate instantiation $i$
    $p = P_B(S)/P_S(S)$
    update scores $p$
Normalize scores
```

Figure 1: General sampling framework.

is approximated, that is, the probability of the sample according to the distribution represented by the belief network $P_B(S)$, and the sampling distribution $P_S(S)$.

So, there are three components in a sampling algorithm for Bayesian networks:

1. a sampling distribution,
2. an instantiation generation method, and
3. a scoring method.

In all known simulation methods for Bayesian networks, the sampling distribution can be written as the product of the sampling distributions of the nodes that are sampled. So let $U \subseteq V$ be the set of sampled nodes and $P_S(x_i)$ the sampling distribution of node x_i for $x_i \in U$, then

$$P_S(V) = \prod_{x_i \in U} P_S(x_i). \tag{4}$$

In most generation schemes, all nodes but the evidence nodes are sampled. We differentiate four different sampling distributions for nodes: the uniform distribution, the forward sampling distribution, the backward sampling distribution, and the Markov blanket distribution.

The uniform distribution assigns an equal probability to each value of a variable, that is, $P_S(x_i) = \frac{1}{|x_i|}$ [10]. For an instantiation that is chosen with uniform distributions, we have $p = \prod_{x_i \in U} P_B(x_i|\pi_i)$. This distribution leads to unsatisfactory results since many non-representative samples are generated.

A forward sampling distribution for x_i, is a distribution that uses the assessment function of the Bayesian network for node x_i [10]. Condition is that all parents of x_i must have been assigned a value already. The parents could have got their values either because they are evidence nodes or because they have been forward sampled before. The sampling distribution is the part of the assessment function that applies to the instantiation of the parents, so $P_S(x_i) = P_B(x_i|\pi_i)$. For instantiations chosen with the forward sampling, we have $p = \prod_{x_i \in V \setminus U} P_B(x_i|\pi_i)$. The method results in good samples as long as the probability of the observed evidence is not very close to zero.

The backward sampling distribution is a recently introduced distribution [9] where values of the parents of a node x_i get assigned a value under the condition that node x_i already has a value. Node x_i may have taken this value either because it is an evidence node or because it is backward sampled before. The values of π_i are generated according to the assessment function of node x_i, namely with probability $P_S(\pi_i) = \frac{P_B(x_i|\pi_i)}{\alpha_i}$, where α_i is a normalizing constant to make the sampling distribution sum to unity. The backward sampling distribution is much less sensitive to probabilities close to zero than the other distributions. Since not all nodes can be backward sampled, for example, because there is no evidence yet, backward sampling must be mixed with another sampling method like for example forward sampling. Of course, the score have to be compensated for this action. In [9] it was shown that this can be calculated by $p = \prod_{x_i \in E \setminus B} P(x_i|\pi_i) \prod_{x_j \in B} \alpha_j$, where B is the set of nodes on which backward sampling was applied and α_j the normalizing constants as in the sampling distribution.

In the Markov blanket sampling distribution, the value of a node x_i is chosen with probability proportional to the product of the probabilities of its so called *Markov blanket* M_i [13]. M_i is the set of parents of x_i, children of x_i, and parents of the children of x_i except x_i itself. Condition for sampling x_i is that all nodes in the Markov blanket have been assigned a value. It is usual to take the values in the previous instantiation. So, x_i is chosen according to the sampling distribution $P_S(x_i) = \alpha_i \prod_{x_j \in M_i} P_B(x_j|\pi_j)$, where α_i is a normalizing constant to make the sampling distribution add to unity and x_i and π_i are instantiated conform the previous instantiation. The score p is 1. Also this distribution results in good samples, as long as the probability tables do not contain values close to zero.

The second component of a simulation algorithm for Bayesian networks is the generation of instantiations. When all nodes are sampled with a uniform sampling distribution, in random order the variables can be assigned a value. This method is known as *equiprobable sampling.*

When all but the evidence nodes are forward sampled, we speak of *logic sampling* [10], *evidence weighting* [8, 16], or *likelihood weighing.* In this paper, we use the last term. Likelihood weighing is performed by first assigning values to the root nodes (if they are not evidence nodes), and then assigning values to nodes of which all parents have been assigned a value, until all nodes have been assigned a value.

When nodes are interchangingly forward and backward sampled, we have an ordering that may be completely different from the topological ordering on the nodes; the ordering start at the parents of a node with evidence and proceeds upwards against the direction of the arcs. Actually, the three requirements on the ordering are:

1. A node that is backward sampled must be instantiated.

2. A node that is forward sampled must have instantiated parents.

3. A node that is not in the ordering is a predecessor of a backward sampled node.

The nodes are assigned a value in the order that fulfills the three requirements.

When all nodes are sampled with a Markov blanket sampling distribution, an initial instantiation need to be generated. This can be done by one of the other simulation methods. The nodes can be assigned a value in random order for the next instantiation.

The third component of a sampling algorithm is the scoring method. Since an instantiation has a value of all variables in $V \backslash E$, the score of the instantiation contributes to all variables; for all variables and for each value a score is updated by adding p to the value that the variable has in the instantiation S. Another scoring method, is to add p weighted by the product of probabilities over the nodes in the Markov blanket. This is computational more expensive when likelihood weighing is used. However, for Gibbs sampling the weighing values are already computed for generating the instantiation, so there is no significant computational cost.

3 Stratified Simulation

Stratified simulation is a well known statistical technique which leads to a better performance of the simulation avoiding rare or desequilibrated samples, and samples with outliers. The basic idea is to divide the sample space into several so called strata and choose in each stratum a previously selected optimal number of samples. This leads to a better representation of the sample space than that obtained by standard samples and better estimates are obtained for a given sample size or a smaller sample size is required for a given predefined error.

3.1 Stratification in Bayesian networks

To understand the basic idea of the method, assume that we have an ordered set V of discrete variables $x_1, x_2, \ldots, x_n$. Assume also that variable x_i has arity r_i and takes values $(0, 1, \ldots, r_i - 1)$ and that we know the conditional forward sampling distributions of each of the nodes given their parents $P_S(x_i|\pi_i)$ for $i = 1, 2, \ldots, n$. Then, we can generate all instantiations and calculate their probabilities of occurrence. We can also order the set of instantiations of V in the following way. Let $I_1 = (x_1, x_2, \ldots, x_n)$ and $I_2 = (y_1, y_2, \ldots, y_n)$, then

$$I_1 < I_2 \Leftrightarrow \exists k, \forall j < k \; x_j = y_j \text{ and } x_k < y_k. \qquad (5)$$

We say that $(x_1, x_2, \ldots, x_n)$ *precedes* $(y_1, y_2, \ldots, y_n)$.

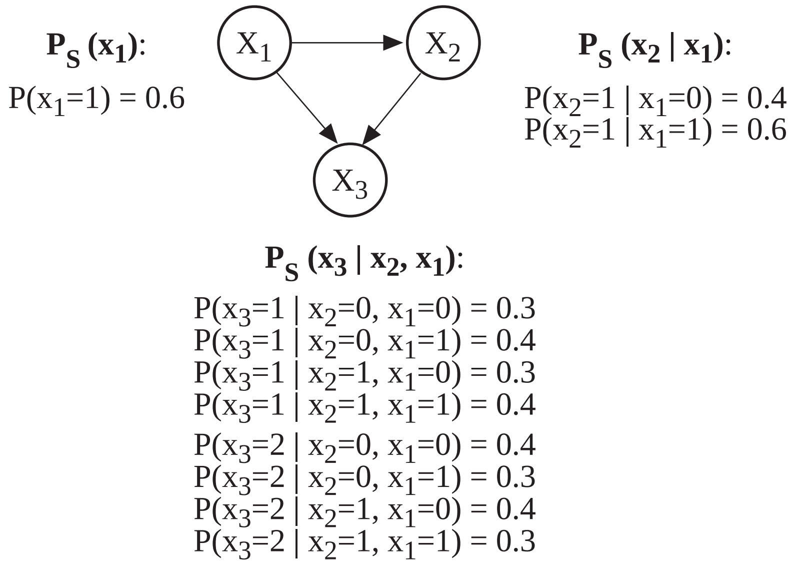

P_S (x_1):
P(x_1=1) = 0.6
X_1
X_2
X_3
P_S (x_2 | x_1):
P(x_2=1 | x_1=0) = 0.4
P(x_2=1 | x_1=1) = 0.6
P_S (x_3 | x_2, x_1):
P(x_3=1 | x_2=0, x_1=0) = 0.3
P(x_3=1 | x_2=0, x_1=1) = 0.4
P(x_3=1 | x_2=1, x_1=0) = 0.3
P(x_3=1 | x_2=1, x_1=1) = 0.4
P(x_3=2 | x_2=0, x_1=0) = 0.4
P(x_3=2 | x_2=0, x_1=1) = 0.3
P(x_3=2 | x_2=1, x_1=0) = 0.4
P(x_3=2 | x_2=1, x_1=1) = 0.3

Figure 2: Bayesian network used in the stratified simulation example.

For example, let $V = \{x_1, x_2, x_3\}$ with arities $r_1 = r_2 = 2$, and $r_3 = 3$ and consider the Bayesian network in Figure 2. The set of ordered instantiations and their associated probabilities of occurrence are given in Table 1, where the accumulated probabilities are also shown. Then, we associate each instantiation with a subinterval of $[0, 1]$, $I_i = [l(i), h(i))$, corresponding to the cumulative probabilities, that is,

$$l(i) = \sum_{j<i} P_S(I_j) \text{ and } h(i) = l(i) + P_S(I_i). \tag{6}$$

where I_j is the j-th instantiation. Figure 3 shows the instantiations, the accumulated probabilities and their associated intervals, which are also shown in Table 1.

The method consists in dividing the $[0, 1]$ interval in equally spaced values, that is , if we desire m steps, we generate the $\{f_i = (i - 0.5)/m; i = 1, 2, \ldots, m\}$ sequence of values and select the associated instantiations. Thus, we start by the first value in the sequence $0.5/m$ and we determine the associated instantiation; then we increase this value by $1/m$ and determine the new instantiation, and we repeat the process until we reach the last value $(m - 0.5)/m$.

For example, assume $m = 4$, then the sequence $f_i = (i - 0.5)/4$ of numbers is

$$(0.125, 0.375, 0.625, 0.875)$$

and the generated sample of instantiations becomes (see Figure 3):

$$(001), (012), (102), (111).$$

It is clear that when m increases the frequency of a given instantiation tends to the exact frequency. Due to the deterministic character of the procedure, no generation of random numbers is required.

Instantiation	Probability	Accumulated probability	Associated interval
(0,0,0)	0.072	0.072	(0.000,0.072)
(0,0,1)	0.072	0.144	(0.072,0.144)
(0,0,2)	0.096	0.240	(0.144,0.240)
(0,1,0)	0.048	0.288	(0.240,0.288)
(0,1,1)	0.048	0.336	(0.288,0.336)
(0,1,2)	0.064	0.400	(0.336,0.400)
(1,0,0)	0.072	0.472	(0.400,0.472)
(1,0,1)	0.096	0.568	(0.472,0.568)
(1,0,2)	0.072	0.640	(0.568,0.640)
(1,1,0)	0.108	0.748	(0.640,0.748)
(1,1,1)	0.144	0.892	(0.748,0.892)
(1,1,2)	0.108	1.000	(0.892,1.000)

Table 1: Ordered instantiations and associated absolute and accumulated probabilities and intervals.

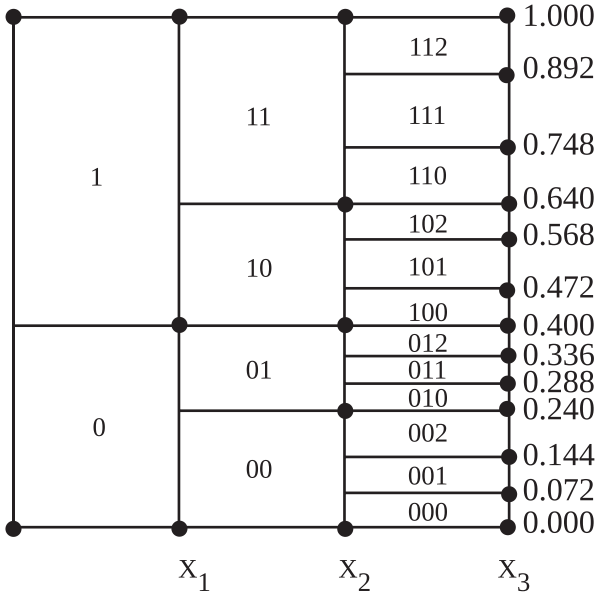


Figure 3: Instantiations and accumulated probabilities.

Note that the resulting sample in this method can be considered as a perfect sample because its empirical distribution function is a perfect straight line as it corresponds to the uniform distribution over the unit interval. In this way, we can consider this method as a numerical procedure more than a true simulation method.

The stratified and modified stratified scheme contribute to the efficient generation of instantiations and can be applied with any sampling distribution except for the Markov blanket sampling distribution. The ordering of the variables may be chosen in such a way that the stratification scheme performs efficiently. Sorting the variables according to their assessment functions as extra criterion for ordering the variables is an option. This prevents backward simulated nodes, which account for a relatively small strata due to their high cardinality of values that are assigned, to be low in the ordering.

3.2 Implementation of the Stratified Simulation Scheme

The method is conceptually very simple but its implementation is complicated. In fact, in general, we cannot generate nor calculate the probabilities of all instantiations because of the associated computational effort (for n binary variables already 2^n instantiations need to be generated). Due to the fact that the method is supposed to be utilized when exact methods cannot be used, we assume that the number of instantiations is much larger than the number of steps m. This implies that many of the instantiations (most of those with lower probabilities) will not appear in the simulated sample. The method must be able to skip these instantiations avoiding unnecessary calculations in an effective way. We can proceed in an ordered way and, using an algorithm to monotonically evolve from the instantiation $(0, 0, \ldots, 0)$ to the instantiation $(r_1 - 1, r_2 - 1, \ldots, r_n - 1)$, take advantage of the order and the deterministic character of the selected sequence to determine the instantiations corresponding to the sequence of values f. The main advantage of this procedure is that for obtaining the new instantiation we only need to update the last k variable values and that we only determine this k value once for each simulated instantiation. This allows a rapid procedure which skip many instantiations at a time. However, we pay the price of determining which variables need to be updated.

The key problem of this procedure is the determination of the instantiation associated with a given value f_i of the sequence f, which is in the interval $[0, 1]$, and the determination of the variable number k above which need to be updated. To this aim, we define an upper $h(i)$ and a lower $l(i) \leq h(i)$ bound for each variable, which indicate the probability values where each variable experiments the next two value changes. For example, in step 4, which corresponds to a value in the interval $(0.240, 0.288)$ (see Figures 3 and 4), the associated instantiation is (010). The next change in the variable X_3 occurs in 0.288 (instantiation (011)) and the following in 0.336 (instantiation (012)). Thus, we change $l(3)$ from 0.240 to 0.288 and $h(3)$ from

0.288 to 0.336. Figure 4 shows how the $h(i)$ and $l(i)$ functions are modified when the actual value in the equally spaced sequence f is in each of the shadowed intervals.

Because we work with a deterministic sequence, once we generate one instantiation associated with a given value of f_i, we can determine how many of the values in the sequence will lead to the same instantiation using the formula (see Figure 5)

$$\delta = \left\lfloor \frac{h(n) - f}{m} \right\rfloor + 1, \tag{7}$$

where $\lfloor . \rfloor$ is the integer part and n is the number of variables. Then we increment the i counter of the f_i sequence in δ units instead of one unit. In this way, we save the work of searching for the same instantiation again and again when the f_i values correspond to the same instantiation. With this technique the simulation time is greatly reduced.

To determine the instantiation associated with the new f_i-value we use a binary searching method, which allows to locate this instantiation in $\ln_2 n$ operations. Given a sequence value f_i, we look for the j index such that $l(j) \leq f \leq h(j)$, which, due to the above definition of the $l(.)$ and $h(.)$ functions is the number of the variable from which we need to update. Once this is done, we can proceed to the updating of the instantiation and the $l(.)$ and $h(.)$ functions, which is not trivial. Figures 6 and 7 give the initialization and the simulation steps of the *Stratified* simulation algorithm which generates the simulated sample and performs the updatings. Figure 8 shows how the modified stratification scheme, that is, the skipping, affects the general sampling framework. $\tilde{P}(i,k)$ is the probability of node i taking value k given its parents as instantiated in the val array and E_j is the random evidence for the node x_j. Note that for deterministic type of evidence this function takes value 1 for one value of k and zero for the rest.

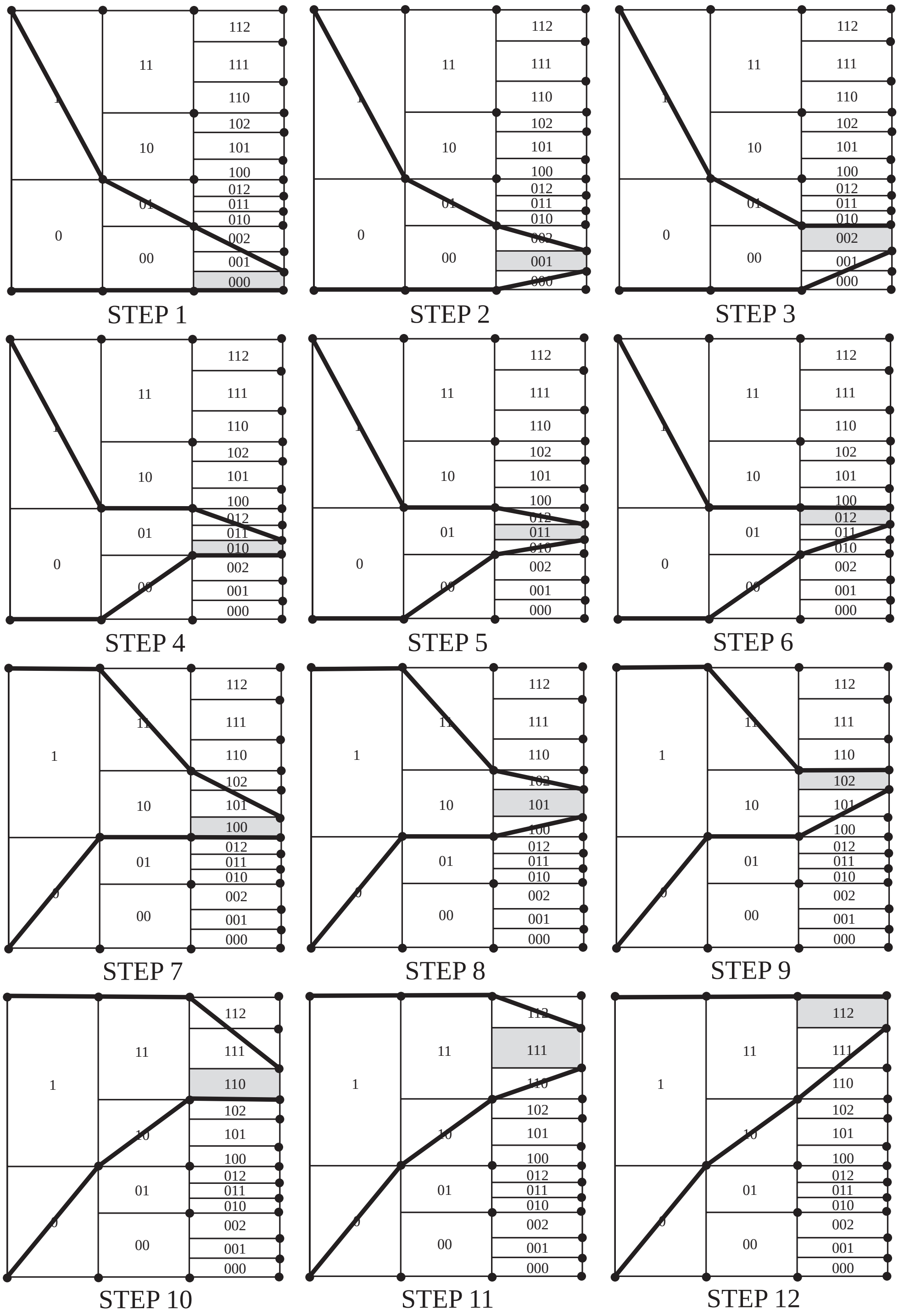
This algorithm has been optimized to reduce the number of multiplications, which is the most costly operation. This is specially useful for variables with high arities.

4 Evaluating Savings

In this section we evaluate the savings obtained by using the stratified simulation methods. We start by a simple case in the following subsection and deal with the general case in the subsequent subsections.

4.1 A Simple Case

Assume a set $(x_1, x_2, \ldots, x_n)$ of n binary identically and independent random variables, where $p < 0.5$ is the probability $P(x_i = 1)$. From a total of 2^n possible instantiations there are $\binom{n}{k}$ such that k of the variables take on the value one and the



STEP 1 STEP 2 STEP 3

STEP 4 STEP 5 STEP 6

STEP 7 STEP 8 STEP 9

STEP 10 STEP 11 STEP 12

Figure 4: Illustration of the $l()$ and $h()$ limits.

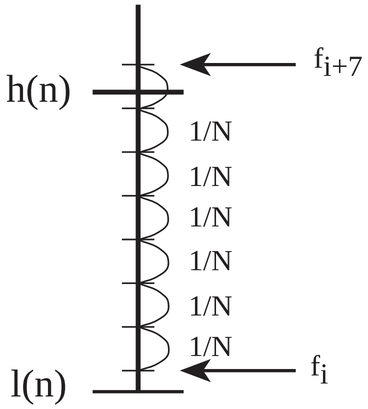


Figure 5: Illustration of how the same values of the f sequence are skipped when they correspond to the same instantiation.

```
$l(0) \leftarrow 0$;  $h(0) \leftarrow 1$
for  $i \leftarrow 1$  to  $n$  do
    $l(i) \leftarrow 0$
    if $x_i \in E$ then
        $val(i) \leftarrow e_i$
        $h(i) \leftarrow h(i-1)$
    else
        $val(i) \leftarrow 0$
        $h(i) \leftarrow h(i-1) * \tilde{P}(i,0)$
```

Figure 6: Initialization step of the stratified simulation algorithm.

```
$f \leftarrow (\text{random}[0:1) + i - 1)/m$
$j \leftarrow$ Binsearch $(f, h)$
while $j <= n$ do
    if $x_j \in E$ then
        $l(j) \leftarrow l(j-1)$
        $h(j) \leftarrow h(j-1)$
    else
        $k \leftarrow 0$
        $l(j) \leftarrow l(j-1)$
        $h(j) \leftarrow l(j) + (h(j-1) - l(j-1)) * \tilde{P}(j,k)$
        while $f > h(j)$ do
          $k \leftarrow k+1$
          $l(j) \leftarrow h(j)$
          $h(j) \leftarrow l(j) + (h(j-1) - l(j-1)) * \tilde{P}(j,k)$
        $val(j) \leftarrow k$
    $j \leftarrow j+1$
return(val)
```

Figure 7: Simulation step of the stratified simulation algorithm.

```
Initialize
for $i \leftarrow 1$ to $m$ do
    $S \leftarrow$ generate instantiation $i$
    $\delta = \lfloor \frac{h(n)-f}{m} \rfloor$
    $p = \delta * P_B(S)/P_S(S)$
    update scores $p$
    $i = i + \delta$
Normalize scores
```

Figure 8: Modified stratified simulation scheme modification of general sampling framework.

remaining $n-k$ variables the value zero. The probability of one such an instantiation is given by

$$p^k(1-p)^{n-k}. \tag{8}$$

Thus, they can be grouped in $n+1$ different groups, where each group includes those instantiations with the same probability and is referred to by number k. We can also order the groups and the instances with respect to their corresponding probabilities of occurrence. We say that instantiation I_1 goes before instantiation I_2 if $P(I_1) > P(I_2)$, which is equivalent to saying that $k_1 < k_2$, that is, the group number of I_1 is smaller that the group number of I_2.

The total probability associated with group k is

$$\left(\begin{array}{c} n \\ k \end{array} \right) p^k(1-p)^{n-k}. \tag{9}$$

Now we evaluate the performance of the stratified simulation methods by comparing the required sample size s for the standard stratified simulation scheme, and the required sample size r for the modified scheme.

Assume that we use the standard stratified scheme and that we perform s runs. Then, the jump occurs in intervals of probability $1/s$. This means that a total of $sp^k(1-p)^{n-k}$ runs fall inside each instantiation of group k. Then, in the modified scheme, we save $\lfloor sp^k(1-p)^{n-k} - 1 \rfloor$ runs for each of those instantiations. Note that we get a saving only in those groups such that

$$sp^k(1-p)^{n-k} \geq 2, \tag{10}$$

which is equivalent to

$$s \geq \frac{2}{p^k(1-p)^{n-k}} \tag{11}$$

Thus, for values of k below the threshold

$$k_{limit} = \frac{-\ln 2 + \ln s + n\ln(1-p)}{\ln(1-p) - \ln(p)} \tag{12}$$

there is no saving. So, the total saving, in terms of runs (required sample size), is

$$s - r = \sum_{k \leq k_{limit}} \left(\begin{array}{c} n \\ k \end{array} \right) \lfloor sp^k(1-p)^{n-k} - 1 \rfloor \tag{13}$$

Figure 9 shows $Log\lfloor r/s \rfloor$ as a function of s for $n = 30$ and different values of p. Two interesting irregularities can be observed in this figure. The sudden jumps are due to the integer part function $\lfloor . \rfloor$ and the increasing segments are due to the fact that for very small decrements of the step no extra saving occurs because no new instantiations are attained. As we can see, the saving percentages are important.

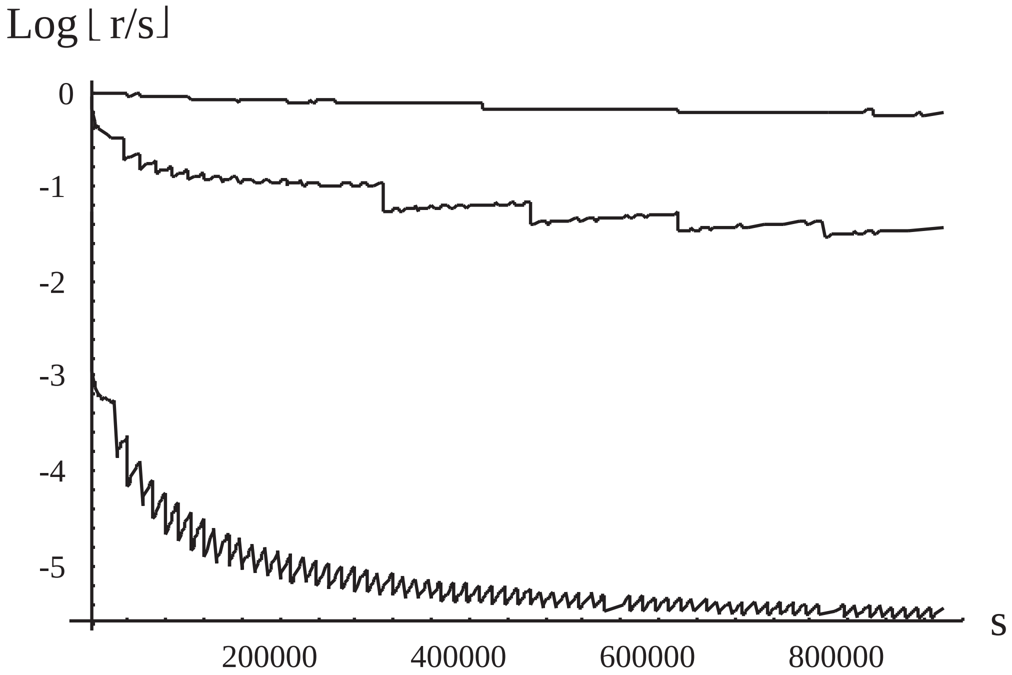


Figure 9: $Log\lfloor r/s \rfloor$ as a function of s for $n = 30$ and $p = 0.2$ (upper curve), 0.1 (intermediate) and 0.01 (lower curve).

When the probability jump is less or equal than the minimum of the probabilities of all instantiations, that is, when all instantiations are attained, we have:

$$\frac{1}{s} \leq p^n \Leftrightarrow s \geq \frac{1}{p^n} > 2^n,$$

and the total saving is

$$\sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) \left(sp^k (1-p)^{n-k} - 1 \right) = s - 2^n, \tag{14}$$

where we have taken into account that

$$\sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) = 2^n. \tag{15}$$

So, after all the 2^n instantiations have been generated there is no increment in the simulation time. Note that when $s > 2^n$ there is no reason for simulating because the exact evaluation of the probabilities of all instantiations can be done in the same computation time.

We can also compute the savings in terms of computer operations. For each binary search we need $\ln_2 n$ comparisons and for each jump we need a multiplication.

Then, the cost difference between r trials with their corresponding jumps and their associated standard stratified simulation method is

$$C(r, s) = s \ln_2 n - r(\alpha + \ln_2 n) = (s - r) \ln_2 n + r\alpha \tag{16}$$

where α is the relative cost of a multiplication with respect to a comparison and asymptotically, the influence of its second term vanishes.

Expression (16), taking into account (13) and Figure 9, shows that the saving dramatically increases with $\ln_2 n$ and $1-p$. This suggests that the modified stratified simulation scheme is an important improvement compared to the standard stratification scheme.

4.2 General case

In the general case, the variables are neither binary nor equally distributed nor independent random variables. In that case we can apply the theory developed by Druzdzel [7]. The basic idea of this methodology is as follows. Let $P(X)$ be the joint probability distribution defined either by a Bayesian network as in (2) or as the sampling distribution of a Bayesian network as in (4). In $P(X)$, X is an equiprobable distributed variables. Now consider $p = P(X)$ as a random variable and let $f_p(p)$ be the density of the random variable p. Thus, if $f_p(p)$ were known, we could determine a threshold value p_0 such that all instantiations with an associated probability lower than p_0 contribute a given probability to the total probability mass.

Fortunately, the joint probability distribution of the random variables can be written as

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\pi_i) \tag{17}$$

and taking logarithms we get

$$\ln P(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \ln P(x_i|\pi_i), \tag{18}$$

then, because of the central limit theorem, assuming a random selection of the variables and a sufficiently regular joint probability for $(x_1, x_2, \ldots, x_n)$, when n is large enough, the above sum can be approximated by a normal random variable. This is equivalent to assuming that

$$f_{\ln p}(\ln p) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{-(\ln p - \mu)^2}{2\sigma^2}\right\}, \tag{19}$$

where $f_{\ln p}(\ln p)$ is the pdf of $\ln p$, or, equivalently, p is approximately distributed as a log-normal random variable.

Assume we have simulated k instantiations. The elemental contribution of all instantiations with probability p to the total probability mass is $kpf_{\ln p}(\ln p)d\ln p$ (in logarithmic scale) and then, the contribution of all instantiations with probability smaller than p_0 to the total probability mass becomes

$$k\int_{-\infty}^{\ln p_0} pf_{\ln p}(\ln p)d\ln p \tag{20}$$

and, taking into account that all instantiations contribute a probability mass of one, we can calculate the value of k by

$$k = \frac{1}{\int\limits_{-\infty}^{\infty} pf_{\ln p}(\ln p)d\ln p} = \exp\left\{-\mu - \sigma^2/2\right\}, \tag{21}$$

since the integral over the lognormal is $\exp(\mu + \sigma^2/2)$ [12]. Substitution of this value into (20) leads to

$$k\int_{-\infty}^{\ln p_0} pf_{\ln p}(\ln p)d\ln p = \int_{-\infty}^{\ln p_0} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{\left(-\ln p - (\mu + \sigma^2)\right)^2}{2\sigma^2}\right\} d\ln p, \tag{22}$$

which shows that $kpf_{\ln p}(\ln p)$ is normal $N(\mu + \sigma^2, \sigma^2)$.

It is important to mention here that, when we have $\mu >> -\sigma^2 - 2\sigma$, an important part of the probability mass of this distribution can be above 0, a limit value above which no $\ln p$ can be. However, if this is the case the normal approximation is not valid and we need to use the extreme value theory to approximate the tails near zero. In addition, if the normal distribution only approximates the empirical data in the central part, the approximation if good when $\mu + \sigma^2 - \beta\sigma < \mu + \beta\sigma$, that is, when $\sigma < 2\beta$, where β is the number of standard deviations apart we move from the mean (standard values of β are 1.5 or 2).

Following Druzdzel ideas, from the set of all instantiations, we can choose a sample of size m. Then we calculate their associated probabilities and take logarithms. In this way we obtain an approximate normal sample and we can estimate the corresponding mean μ and standard deviation σ by their corresponding sampling values. Then we can determine the probability threshold value p_0 for which all instantiations less likely than p_0 contribute totally less than f to the total probability by solving the following equation

$$\int_{-\infty}^{\ln p_0} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{\left(-\ln p - (\mu + \sigma^2)\right)^2}{2\sigma^2}\right\} d\ln p = f \Leftrightarrow \ln p_0 = \mu + \sigma^2 + \sigma\Phi^{-1}(f) \tag{23}$$

where $\Phi(.)$ is the cdf of the standard normal $N(0,1)$. In the next subsection, we will give a formula for the exact calculation of μ and σ.

Note that p_0 can be used to calculate the step value s to be used in the stratified simulation scheme. In fact if we choose $s = 1/p_0$ we can guarantee an error less than f.

We can calculate the fraction of instantiations ℓ that are less likely than p_0 as

$$\ell = \int_{-\infty}^{\ln p_0} f_{\ln p}(\ln p) d \ln p = \int_{-\infty}^{\ln p_0} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{-(\ln p - \mu)^2}{2\sigma^2}\right\} d \ln p \Leftrightarrow \ln p_0 = \mu + \sigma\Phi^{-1}(\ell). \tag{24}$$

4.3 Calculation of Mean and Variance

It is useful to know on forehand what the parameters of the lognormal distribution are so that the stepsize for the stratified simulation scheme can be determined. The following theorem tells how to calculate them.

Theorem 1 *Let P be a joint probability distribution that can be written as $\prod_{i=1}^{n} P(x_i|\pi_i)$. Let X be a random variable which is uniformly drawn from the instantiations of V. Let $Y = \ln P(X)$ Then, the average μ and variance σ^2 of Y are*

$$\mu = \sum_{i=1}^{n} \mu_i,$$

and

$$\sigma^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} \eta_{ij} - \mu^2,$$

where $\mu_i = \sum_{x_i\pi_i} \ln P(x_i|\pi_i)/|x_i\pi_i|$, $\eta_{ij} = \sum_{x_i\pi_i x_j\pi_j} \ln P(x_i|\pi_i) \cdot \ln P(x_j|\pi_j)/|x_i\pi_i x_j\pi_j|$ and $|X|$ denotes the number of instantiations of X.

Proof: First, we derive the average of Y and then the variance. The average of Y is by definition

$$\mu = E\{Y\} = \frac{1}{|V|}\sum_{V} \log P(V),$$

And by definition of $P(V)$, we have

$$\mu = \frac{1}{|V|}\sum_{V}\sum_{i=1}^{n} \ln P(x_i|\pi_i).$$

By changing order of summation, this is

$$\mu = \frac{1}{|V|}\sum_{i=1}^{n}\sum_{V} \ln P(x_i|\pi_i).$$

Now we can split the last summation in summation over the instantiations of the variables $x_i\pi_i$ and those not in $x_i\pi_i$ and obtain

$$\mu = \frac{1}{|V|}\sum_{i=1}^{n}\sum_{V\backslash x_i\pi_i}\sum_{x_i\pi_i}\ln P(x_i|\pi_i).$$

Because the summed term remains the same for the second summation, we can write this as

$$\mu = \frac{1}{|V|}\sum_{i=1}^{n}|V\backslash x_i\pi_i|\sum_{x_i\pi_i}\ln P(x_i|\pi_i).$$

Bringing the constant $|V|$ whitin the outer summation gives

$$\mu = \sum_{i=1}^{n}\frac{|V\backslash x_i\pi_i|}{|V|}\sum_{x_i\pi_i}\ln P(x_i|\pi_i). \tag{25}$$

Now realizing that $|V\backslash x_i\pi_i| = \prod_{x_j\in V\backslash x_i\pi_i} r_j$ (where r_j is the cardinality of variable x_j) and $|V| = \prod_{x_k\in V} r_k$, we can write $\frac{|V\backslash x_i\pi_i|}{|V|}$ as $1/\prod_{x_k\in x_i\pi_i} r_k$. This is exactly the same as $\frac{1}{|x_i\pi_i|}$. So, we can write (25) as,

$$\mu = \sum_{i=1}^{n}\frac{1}{|x_i\pi_i|}\sum_{x_i\pi_i}\ln P(x_i|\pi_i).$$

By definition of μ_i in the theorem, we get the stated result

$$\mu = \sum_{i=1}^{n}\mu_i.$$

The variance of Y is by definition

$$\sigma^2 = E\{(Y-\mu)^2\} = E\{Y^2\} - \mu^2. \tag{26}$$

Since the last term in the theorem have been already obtained, we concentrate on the first. By definition, we have

$$E\{Y^2\} = \frac{1}{|V|}\sum_{V}\left(\ln P(V)\right)^2,$$

and by definition of $P(V)$ this is

$$E\{Y^2\} = \frac{1}{|V|}\sum_{V}\left(\sum_{i=1}^{n}\ln P(x_i|\pi_i)\right)^2.$$

Splitting the quadratic gives

$$E\{Y^2\} = \frac{1}{|V|}\sum_{V}\left(\sum_{i=1}^{n}\ln P(x_i|\pi_i)\right)\cdot\left(\sum_{j=1}^{n}\ln P(x_j|\pi_j)\right).$$

By reordering terms, we get

$$E\{Y^2\} = \frac{1}{|V|} \sum_V \sum_{i=1}^n \sum_{j=1}^n \ln P(x_i|\pi_i) \cdot \ln P(x_j|\pi_j).$$

By changing sums and bringing the constant $\frac{1}{|V|}$ within the sum just as we did for the calculation of μ, we get

$$E\{Y^2\} = \sum_{i=1}^n \sum_{j=1}^n \sum_V \frac{1}{|V|} \ln P(x_i|\pi_i) \cdot \ln P(x_j|\pi_j).$$

Now by the same observation as above, we have that $\sum_V \frac{1}{|V|} \ln P(x_i|\pi_i) \cdot \ln P(x_j|\pi_j)$ equals $\sum_{x_i\pi_i x_j\pi_j} \frac{1}{|x_i\pi_i x_j\pi_j|} \ln P(x_i|\pi_i) \cdot \ln P(x_j|\pi_j)$. Substitution gives

$$E\{Y^2\} = \sum_{i=1}^n \sum_{j=1}^n \sum_{x_i\pi_i x_j\pi_j} \frac{1}{|x_i\pi_i x_j\pi_j|} \ln P(x_i|\pi_i) \cdot \ln P(x_j|\pi_j).$$

By definition of η_{ij} in the theorem, this can be written as

$$E\{Y^2\} = \sum_{i=1}^n \sum_{j=1}^n \eta_{ij}.$$

Finally, substitution into Formula (26) gives the stated result,

$$\sigma^2 = \sum_{i=1}^n \sum_{j=1}^n \eta_{ij} - \mu^2.$$

■

Theorem 1 gives the formulas that makes that calculation of the parameters of the lognormal distribution possible on forehand. Note that P in the theorem may be either a distribution defined by a Bayesian network or be a sampling distribution. This gives useful insight in the behavior of the stratified algorithm. The complexity of calculating the mean μ is $O(n \cdot k)$ where n is the number of nodes and k the largest probability table. The complexity of calculating the variance σ^2 is $O(n^2 \cdot k^2)$. Note that since $\eta_{ij} = \eta_{ji}$, a lot of terms are the same, and we can write the variance as

$$\sigma^2 = \sum_{i=1}^n \eta_{ii} + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n \eta_{ij} - \mu^2,$$

saving half of the computational effort. Furthermore, when $x_i\pi_i \cap x_j\pi_j = \emptyset$ then we have

$$\begin{aligned} \eta_{ij} &= \textstyle\sum_{x_i\pi_i x_j\pi_j} \ln P(x_i|\pi_i) \cdot \ln P(x_j|\pi_j)/|x_i\pi_i x_j\pi_j| \\ &= \textstyle\sum_{x_i\pi_i} \ln P(x_i|\pi_i)/|x_i\pi_i| \cdot \sum_{x_j\pi_j} \ln P(x_j|\pi_j)/|x_j\pi_j| = \mu_i \cdot \mu_j. \end{aligned}$$

So most of the time, η_{ij} can be calculated simply as the product of the means μ_i and μ_j. The calculation of μ and σ need only be performed once. When evidence is entered and the distribution that we are interested in changes, the new means and variances can be incrementally calculated.

Let V be a set of variables, P be a distribution over V and μ and σ as in Theorem 1. Let P' be equal to $P(V|x_k = e)$, where x_k is some variable in V. Let μ' and σ'^2 be the mean and variance for P' as defined in Theorem 1. Then μ' can be calculated using μ with the following formula,

$$\mu' = \mu - \mu_k + \sum_{x_i \in K_e} (\mu_i + \mu_i') \,. \tag{27}$$

where K_e are the children in e that are not evidence nodes. And, σ'^2 can be calculated using σ^2 by

$$\sigma'^2 = \sigma^2 - \sum_{i=1, x_i \notin E}^{n} \eta_{ik} + \sum_{i=1, x_i \in K_k}^{n} (-\eta_{ik} + \eta_{ik}) + \mu^2 - \mu'^2. \tag{28}$$

By storing μ_i and σ_{ij} in memory, Formula (27) and (28) may be calculated efficiently.

5 Experimental Results

We have performed some experiments to compare the stratified simulation and the modified simulation scheme with the likelihood weighing and Markov boundary scheme. Ten Bayesian networks over twenty binary variables were used. First ten network structures were generated. Initially an ordering on the variables is made, two nodes x_i and x_j are randomly selected, and an arc $x_j \rightarrow x_i$ is added if $i > j$ and $x_i \rightarrow x_j$ otherwise. Then, randomly one of the variables is selected that is connected to at least one arc and one of the variables that is not connected to any arc. An arc is placed between these nodes in the direction that satisfies the ordering. The process is repeated until all nodes are connected. This method generates networks with a bias towards networks with some nodes having a high number of arcs connected as opposed to networks with long strings of nodes. Realistic networks seem to have the same kind of bias.

For these ten networks structures, assessment functions were generated for binary variables by selecting a random number. In the first experiment, the random number was selected from the unit interval and in the second experiment the number was uniformly selected from $[0, 0.1] \cup [0.9, 1]$. The experiments were performed by generating 100 up to 1000 instantiations, increasing by 100 in each test and 1000 up to 10000 increasing by 1000 in each test. The performance was measured in time to execute an algorithm and the error in the approximation according to $\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{1} P(x_i = k) \ln \left[P(x_i = k)/\hat{P}(x_i = k)\right]$ where $P(x_i = k)$ is the exact probability that x_i takes value k and $\hat{P}(x_i = k)$ its approximated probability.

Figure 10 shows the $\ln(T)$ versus $\ln(E)$ plot of the networks described above for likelihood weighing, Markov sampling, the stratified scheme, and the modified stratified scheme. As reported in other papers [5, 16], likelihood weighing is more efficient than Markov sampling. Also, the outperformance of likelihood weighing by the stratified scheme was reported before [1].

The experiments show that, for a reduced number of simulations, the modified stratified scheme performs equal to the stratified scheme, when the probabilities in the Bayesian network are chosen from the unit interval. This behavior does not appear when the probabilities are extreme, that is, chosen from $[0, 0.1] \cup [0.9, 1]$. The error is exactly the same for both methods. This is because the calculated score is for both algorithms the same. Therefore, the error is also the same for both algorithms. However, the modified stratified scheme performs better than the stratified scheme regarding computation time. The reason for this behavior is that networks with extreme probabilities result in large strata where skipping saves a lot of calculations. These large strata are less frequent in networks with non-extreme probabilities, so that skipping does not really help there. So, the modified stratified scheme performs better when large strata can be expected.

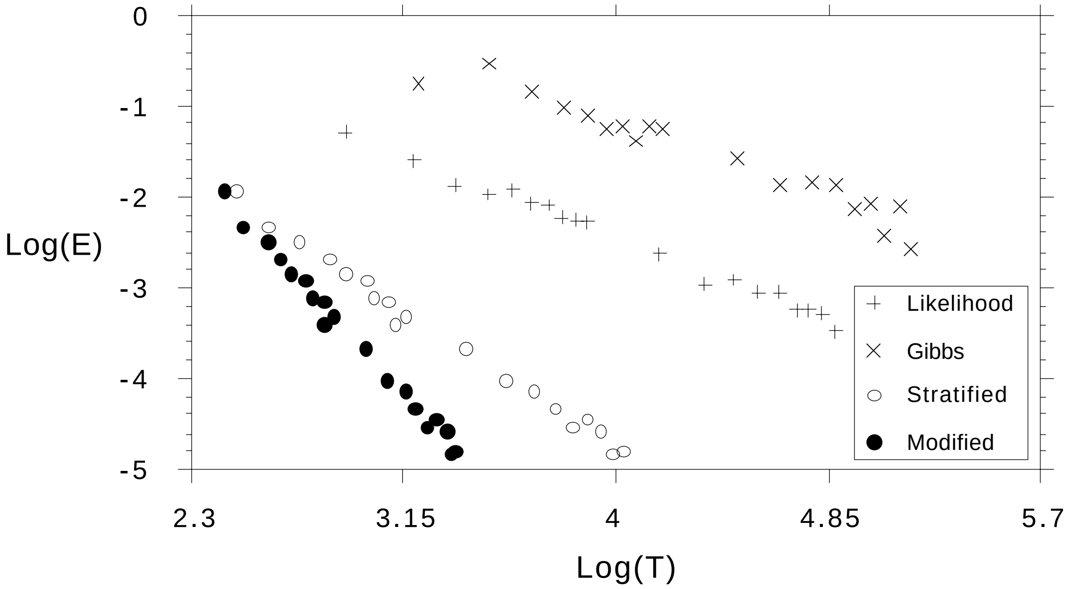


Figure 10: Logarithm of error vs. logarithm of computation time for different algorithms.

Example 1 *For two of the generated Bayesian networks, the set of* $\ln p$ *values associated with all instantiations have been generated. They are shown in a normal probability paper (a graph which has been scaled such that normal samples appear as straight lines). Figure 11 shows the result for a network with assessment functions* *chosen from* $[0, 1]$ *and Figure 12 for assessment functions taken from* $[0, 0.1]$ *or* $[0.9, 1]$*. As we can see the normal approximation is very good in the central part but not in the tails.*

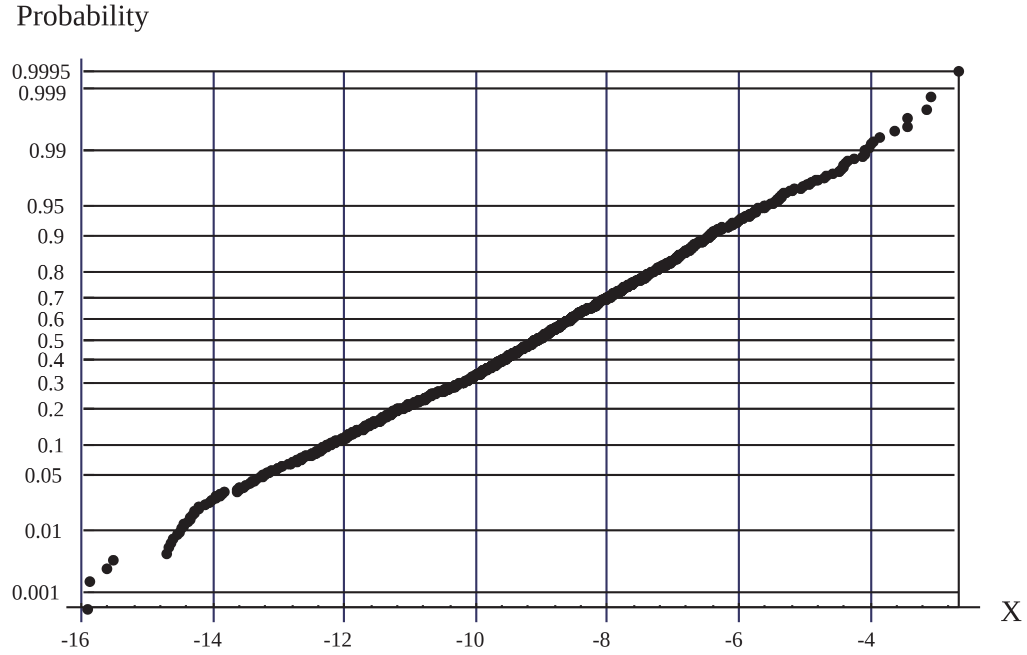


Figure 11: Set of log values for an example of a 10 nodes network with assessment functions chosen from $[0, 1]$ in probability paper.

Figures 13 and 14 show the densities of the logarithm of the instantiation probabilities and the contributions to the total probability mass as derived from the previously described method. For the example in Figure 11 we get $\mu = -9.18329$ *and* $\sigma = 2.2629$ *(note that* $\sigma < 2\beta$ *for* $\beta = 2$ *or* 3*) and for the example in Figure 12 we get* $\mu = -15.8833$ *and* $\sigma = 5.24777$ *(note that* $\sigma > 2\beta$ *for* $\beta = 2$ *or* 3*). Evaluation of the savings is correct for the case of Figure 13 because we are using the central part of the normal distribution. Thus, we can say that* 50% *of the instantiations contribute* 98.8% *of the total probability mass. On the contrary, evaluation of savings is completely wrong for the case of Figure 14 where we need to approximate the right tail. Thus, extreme value theory should be applied here, instead.*

6 Conclusions and Recommendations

A modified version of the stratified simulation scheme for inference in Bayesian networks has been presented and analysed theoretically and experimentally. The performance is better than previous simulation methods, not only in simulation time but also in approximation errors. In addition, theoretical results allow obtaining

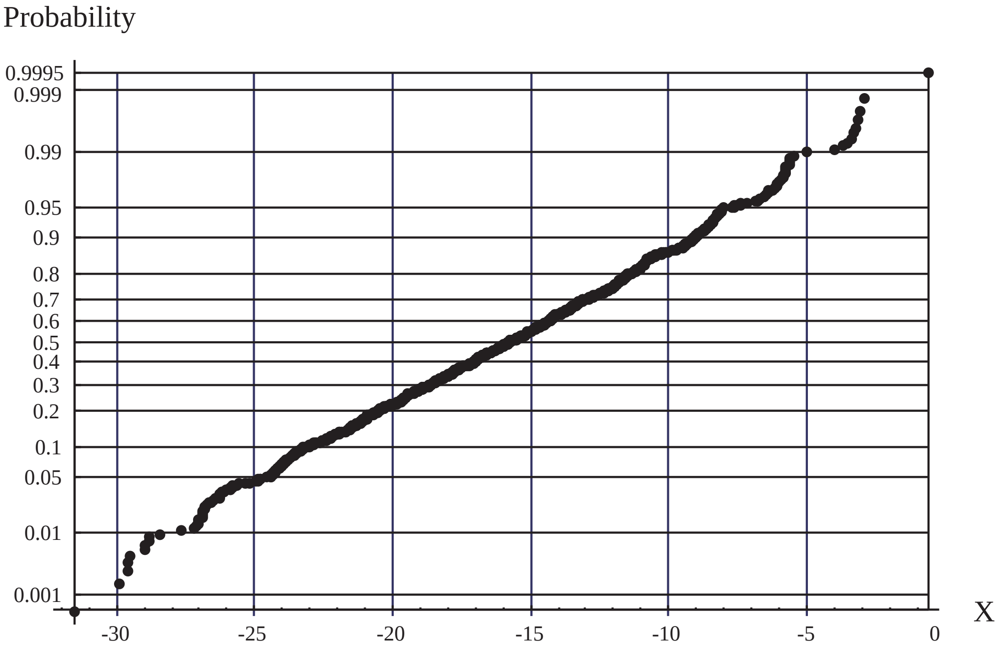


Figure 12: Set of log values for an example of a 10 nodes network with assessment functions chosen from $[0, 0.1]$ in probability paper.

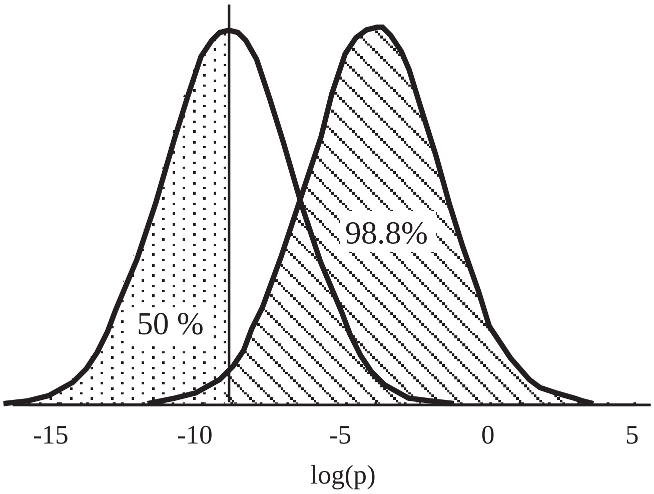


Figure 13: Pdf of the logarithms of the instantiations probabilities and density of the contributions to the total probability mass for an example of a 10 nodes network with assessment functions chosen from $[0, 1]$.

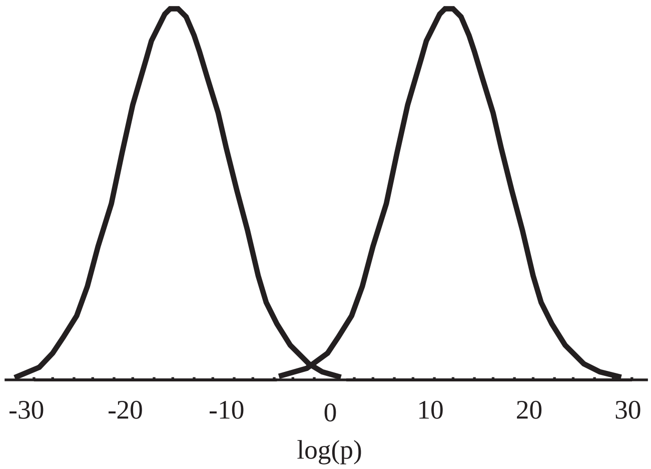


Figure 14: Pdf of the logarithms of the instantiations probabilities and density of the contributions to the total probability mass for an example of a 10 nodes network with assessment functions chosen from $[0, 0.1]$.

probability bounds for the instantiations such that neglecting all instantiations with probability less than a given threshold value leads to controlled errors in marginal probabilities. Calculation of these bounds is based on the use of the central limit theorem to approximate the logarithm of the probability of the different instantiations (Druzdzel [7]). It is shown that the parameters μ and σ of this normal distribution can be efficiently calculated from the assessment functions associated with the Bayesian. In cases where $\sigma < 2\beta$, the normal approximation seems to give good results; otherwise, the tail must be approximated using extreme value theory. A detailed analysis of this case is out of the scope of this paper and will be analysed in a future work.

The stratified simulation scheme is inherently based on discrete variables. For the stratified scheme some adoptions have to be made to be applicable. By choosing an appropriate sampling distribution, the stratified simulation scheme can be applied to the discrete variables in the distribution and the forward sampling scheme to the continuous variables. The discrete variables get assigned a value first with sampling distribution $P(x_i|\pi_i)$ if they have no continuous variable in their parent-sets and $P(x_i|\pi_i, \mu_i)$ where μ_i is some average derived from the distributions of the continuous variables in the parent-set of x_i. Of course the score has to be adapted appropriately.

When there are too many continuous variables, this scheme will not be more efficient than forward sampling since more samples are necessary to give a representative sample. Experimental results will have to give insight in how many continuous variables may appear in the network such that the stratified scheme is more appropriate than forward simulation.

7 Acknowledgments

The first author thanks Enrique Castillo for his invititation to visit the University of Cantabria, which made the cooperation a successfull enterprise. The authors are grateful to the Dirección General de Investigación Científica y Técnica (DGICYT) (project PB92-0504), for partial support of this work.

References

[1] R. Bouckaert. Properties of bayesian belief networks learning algorithms. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109, Seattle, 1994. Morgan Kaufmann Publishers, San Francisco, California.

[2] R. Bouckaert. A stratified simulation scheme for inference in bayesian belief netwoks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 110–117, Seattle, 1994. Morgan Kaufmann Publishers, San Francisco, California.

[3] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[4] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–348, 1992.

[5] S.B. Cousins, W. Chen, and N.E. Frisse. Caben: A collection of algorithms for belief networks. Technical Report WUCS-91-25, Medical Informatics Laboratory, Washington University, St Louis, MO, 1991. obtainable by ftp from wuarchive.wustl.edu:/ /caben.

[6] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993.

[7] M.J. Druzdzel. Some properties of joint probability distributions. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 187–194, Seattle, 1994. Morgan Kaufmann Publishers, San Francisco, California.

[8] R. Fung and K.C. Chang. Weighing and integrating evidence for stochastic simulation in bayesian networks. In L.N. Kanal M. Henrion, R.D. Shachter and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence, 5.* North Holland, Amsterdam, 1990.

[9] R. Fung and B. Del Favero. Backward simulation in bayesian networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 227–234, Seattle, 1994. Morgan Kaufmann Publishers, San Francisco, California.

[10] M. Henrion. Propagation of uncertainty by logic sampling in Bayes' networks. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 149–164. Elsevier Science Publishers, Amsterdam, 1988.

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

[12] G.P. Patil, M.T. Boswell, and M.V. Ratnaparkhi. Statistical distributions in scientific work series. In G. P. Patil, editor, *Continuous univariate models.*, page 594. International Co-operative Publishing House, Burtonsville, MD, 1984.

[13] J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32:245–257, 1987.

[14] J. Pearl. *Probabilistic Reasoning in Intellient Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[15] M. A. Peot and R. D. Shachter. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence*, 48:299–318, 1991.

[16] R.D. Shachter and M.A. Peot. Simulation approaches to general probabilistic inference on belief networks. In L.N. Kanal M. Henrion, R.D. Shachter and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence, 5.* North Holland, Amsterdam, 1990.