RESEARCH NOTE

Evidential Reasoning Using Stochastic Simulation of Causal Models*

Judea Pearl
Cognitive Systems Laboratory, UCLA Computer Science Department, Los Angeles, CA 90024, U.S.A.

Recommended by A. Bundy and V. Kumar

ABSTRACT

Stochastic simulation is a method of computing probabilities by recording the fraction of time that events occur in a random series of scenarios generated from some causal model. This paper presents an efficient, concurrent method of conducting the simulation which guarantees that all generated scenarios will be consistent with the observed data. It is shown that the simulation can be performed by purely local computations, involving products of parameters given with the initial specification of the model. Thus, the method proposed renders stochastic simulation a powerful technique of coherent inferencing, especially suited for tasks involving complex, nondecomposable models where "ballpark" estimates of probabilities will suffice.

1. Introduction

Stochastic simulation is a method of computing probabilities by counting the fraction of time that events occur in a series of simulation runs. If a causal model of a domain is available, the model can be used to generate random samples of hypothetical scenarios that are likely to develop in the domain. The probability of any event or combination of events can then be computed by recording the fraction of time it registers "true" in the samples generated.

Stochastic simulation shows considerable potential as a probabilistic inference engine that combines evidence correctly and is still computationally tractable. Unlike numerical schemes, the computational effort is unaffected by

* This work was supported in part by the National Science Foundation, Grant #DSR 83-13875.

the presence of dependencies within the causal model; simulating the occurrence of an event under a given set of conditions requires the same computational effort regardless of whether the conditions are correlated or not. Stochastic simulation carries a special appeal to AI researchers in that it develops probabilistic reasoning as a direct extension of deterministic logical inference [2]. It explicitly represents probabilities as "frequencies" in a sample of truth values, and these values, unlike numerical probabilities, can be derived by familiar theorem-proving techniques and combined by standard logical connectives. Neither is the technique foreign to human reasoning; assessing uncertainties by mental sampling of possible scenarios seems a very natural heuristic and is, no doubt, an important component of human judgment.

Another feature offered by simulation techniques is their inherent parallelism. If we associate a processor with each propositional variable in the model, then the simultaneous occurrence of events within each scenario can be generated by concurrently activating the processors responsible for these events. For example, the occurrence of the event "Joe entered the restaurant" could, in one run, trigger the simultaneous events A: "Joe liked the food," B: "Joe hated the noise" and C: "The prices were reasonable," while in a different run, the combination of $(\neg A, B, \neg C)$ may occur. Although parallel techniques have also been developed for numerical computation of probabilities [10], the simulation approach embodies the added advantage of message simplicity. Instead of relaying probability distributions, the messages passing between processors are the actual values assigned to their corresponding variables. This conforms to the connectionist paradigm of reasoning, where processors are presumed to communicate merely by relaying their levels of activity.

Within AI, reasoning by probabilistic sampling has been suggested by Hinton et al. [9] as one of the tasks executable on the Boltzmann machine. Geman and Geman [7] combined probabilistic sampling with simulated annealing to restore noisy images. This work does not take an explicit probabilistic model of the domain as input, but, rather uses probabilistic sampling as an auxiliary computational tool to solve problems specified by nonprobabilistic constraints. Bundy [2] has suggested an approach to probabilistic logic (called *incidence calculus*) based, essentially, on truth-value sampling. Each logical predicate is characterized by a bit string representing a sequence of truth values. The bits in each string are combined by standard logical connectives, and the probability of the predicate is the fraction of bits which are "true" in the predicate's bit string. The model of the domain is specified in terms of logical axioms, together with initial bit strings assigned to selected sentences.

Recently, Henrion [8] applied Bundy's proposal to evidential reasoning tasks, using Bayesian networks as a formalism for representing causal relationships [10]. Henrion's scheme, called *logic sampling*, uses an uninstantiated Bayesian network as a scenario generator which, in each simulation run, assigns random values to all system variables. Belief distributions are calculated by averaging the frequency of events over those cases in which the evidence variables agree with the data observed.

This scheme offers an advantage over the Markov-field approaches of Hinton et al. [9] and Geman and Geman [7] in that it takes as input both the structure of a causal model provided by a domain expert and the forward conditional probabilities reflecting the expert's direct experience; thus, the simulation is conducted at a conceptually meaningful level of description. However, since the simulation proceeds only forward in time, there is no way to account in advance for evidence known to have occurred until variables corresponding to these observations are brought into play and get sampled. If they match the observed data, the run is counted; otherwise, it must be discarded. The result is that the scheme requires an excessive number of simulation runs. In cases comprising large numbers of observations (e.g., 20), all but a small fraction (e.g., 10^{-6}) of the simulations may be discarded, especially when a rare combination of data occurs.

A more desirable way of accounting for the evidence would be to permanently "clamp" the evidence variables to the values observed, then conduct stochastic simulation on the clamped network. The question remains, though, how to propagate the random values coherently through the network, now that boundary conditions are imposed on both the top and bottom nodes, i.e., premises as well as consequences.

This paper offers a solution which involves a two-phase cycle: local numerical computation followed by logical sampling. The first step involves computing, for some variable X, its conditional distribution, given the states of all its neighbors' variables. The second phase involves sampling the distribution computed in the first step and instantiating X to the value selected by the sampling. The cycle then repeats itself by sequentially scanning through all the variables in the system.

Section 2 illustrates the proposed scheme using a simple example taken from medical diagnosis. Section 3 proves the correctness of the formula used in these computations, and Section 4 discusses methods for implementing the sampling scheme in parallel.

2. Illustrating the Proposed Scheme

We shall illustrate the operation of the proposed scheme with a simple example borrowed from Spiegelhalter [11], originally given by Cooper [4]:

> Metastatic cancer is a possible cause of a brain tumor and is also an explanation for increased total serum calcium. In turn, either of these could explain a patient falling into a coma. Severe headache is also possibly associated with a brain tumor.

Figure 1 shows the Bayes network representing these relationships. We use capital letters to represent propositional variables (i.e., dichotomies) and lower case letters for their associated propositions. For example, $C \in \{1, 0\}$ represents the dichotomy between having or not having brain tumor. c stands for the assertion $C = 1$ or "brain tumor is present" and $\neg c$ stands for the negation of c, i.e., $C = 0$.

The table below expresses the influences in terms of conditional probability distributions. Each variable is characterized by a distribution, called a *link matrix*, that specifies the probability of that variable, given the state of its parents. The root variable, having no parent, is characterized by its prior distribution.

$P(A)$:	$P(a) = 0.20$	
$P(B \mid A)$:	$P(b \mid a) = 0.80$	$P(b \mid \neg a) = 0.20$
$P(C \mid A)$:	$P(c \mid a) = 0.20$	$P(c \mid \neg a) = 0.05$
$P(D \mid B, C)$:	$P(d \mid b, c) = 0.80$	$P(d \mid \neg b, c) = 0.80$
	$P(d \mid b, \neg c) = 0.80$	$P(d \mid \neg b, \neg c) = 0.05$
$P(E \mid C)$:	$P(e \mid c) = 0.80$	$P(e \mid \neg c) = 0.60$

Given this information, our task is to compute the posterior probability of every proposition in the system, given that a patient is observed to be suffering from severe headaches (e) but is definitely not in coma ($\neg d$), i.e., $E = 1$ & $D = 0$. The first step is to instantiate all the unobserved variables to some arbitrary initial state, say $A = B = C = 1$, and then let each variable, in turn, choose another state in accordance with the conditional probability of that variable, given the current state of the other variables. Thus, for example, if we denote by w_A the state of all variables except A (i.e., $w_A = \{B = 1, C = 1, D = 0, E = 1\}$), then the next value of A will be chosen by tossing a coin that favors 1 to 0 by a ratio of $P(a \mid w_A)$ to $P(\neg a \mid w_A)$.

In Section 3, we shall show that $P(X \mid w_X)$, the distribution of each variable X

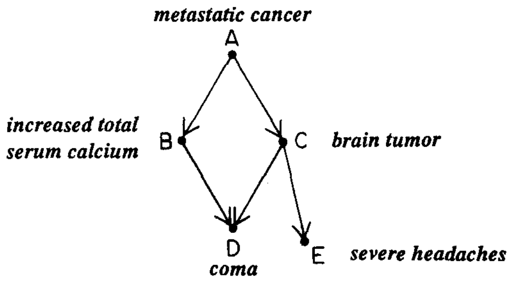

FIG. 1. Bayes' network representing qualitative influences in the example.

conditioned on the values w_X of all other variables in the system, can be calculated by purely local computations. It is given simply as the product of the link matrix of X times the link matrices of its children:

$$P(A|w_A) = P(A|B, C, D, E) = \alpha P(A)P(B|A)P(C|A)\,, \tag{1a}$$

$$P(B|w_B) = P(B|A, C, D, E) = \alpha P(B|A)P(D|B, C)\,, \tag{1b}$$

$$P(C|w_C) = P(C|A, B, D, E) = \alpha P(C|A)P(D|B, C)P(E|C)\,, \tag{1c}$$

where the α are normalizing constants that render the respective probabilities' sum to unity. The probabilities associated with D and E are not needed because these variables are assumed to be fixed at $D = 0$ and $E = 1$. Note that a variable X may determine its transition probability $P(X|w_X)$ by inspecting only its parents, its children and those with which it shares children. (This set of variables is called the *Markov blanket* of X [10].) For example, A needs to inspect only B and C, while B needs to inspect only A, C and D.

For demonstration purposes, we will activate the variables sequentially, in the order A, B, C, acknowledging that any other schedule would be equally adequate.

Activating A

Step 1. Node A inspects its children B and C and, finding both at 1, computes (using equation (1a)):

$$\begin{aligned} P(A=1|w_A) &= P(A=1|B=1, C=1) \\ &= \alpha P(a)P(b|a)P(c|a) \\ &= \alpha \times 0.20 \times 0.80 \times 0.20 \\ &= \alpha \times 0.032\,, \end{aligned}$$

$$\begin{aligned} P(A=0|w_A) &= P(A=0|B=1, C=1) \\ &= \alpha P(\neg\alpha)P(b|\neg a)P(c|\neg a) \\ &= \alpha \times 0.80 \times 0.20 \times 0.05 \\ &= \alpha \times 0.008\,, \end{aligned}$$

$$\alpha = [0.032 + 0.008]^{-1} = 25$$

yielding

$$P(A=1|w_A) = 25 \times 0.032 = 0.80\,,$$
$$P(A=0|w_A) = 25 \times 0.008 = 0.20\,.$$

Step 2. Node A consults a random number generator that issues ones with probability 80% and zeros with 20%. Assuming the value sampled is 1, A adopts this value $A = 1$, and control shifts to node B.

Activating B

Step 1. Node B inspects its neighbors and, finding them with values $A = 1$, $C = 1$, $D = 0$, it computes (using equation (1b)):

$$\frac{P(B=1|w_B)}{P(B=0|w_B)} = \frac{P(B=1|A=1, C=1, D=0)}{P(B=0|A=1, C=1, D=0)}$$
$$= \frac{\alpha P(b|a)P(\neg d|b, c)}{\alpha P(\neg b|a)P(\neg d|\neg b, c)}$$
$$= \frac{0.80 \times (1-0.80)}{(1-0.80)\times(1-0.80)} = \frac{4}{1} .$$

Step 2. As A did in its turn, B samples a random number generator favoring 1 by a 4:1 ratio. Assuming this time that the value sampled is 0, B is set to 0 and gives control to C.

Activating C

Step 1. The neighbors of C are at the state

$$w_C = \{A=1, B=0, D=0, E=1\} .$$

Therefore, from equation (1c):

$$\frac{P(c|w_C)}{P(\neg c|w_C)} = \frac{P(c|a)P(\neg d|\neg b, c)P(e|c)}{P(\neg c|a)P(\neg d|\neg b, \neg c)P(e|\neg c)}$$
$$= \frac{0.20\times(1-0.80)\times 0.80}{(1-0.20)\times(1-0.05)\times 0.60} = \frac{1}{14.25} .$$

Step 2. C samples a random number generator favoring zeros by a 14.25:1 ratio. Assuming 0 is sampled, C adopts the value 0 and gives control to A.

The cycle now repeats itself in the order A, B, C until a query is posted. For example, "What is the posterior distribution of A?" Such a query can be answered in two ways, either by computing the fraction of times A registered the value 1 or by taking the average of the conditional probabilities $P(A = 1|w_A)$ computed by A. The latter normally yields faster convergence.

To illustrate, the value of $P(A = 1|w_A)$ computed in the next activation of A would be

$$\begin{aligned} P(A=1|B=0,\ C=0) &= \alpha P(\alpha)P(\neg b|a)P(\neg c|a) \\ &= \alpha \times 0.20 \times (1-0.80) \times (1-0.20) \\ &= \alpha \times 0.032\,, \end{aligned}$$

$$\begin{aligned} P(A=0|B=0,\ C=0) &= \alpha P(\neg a)P(\neg b|\neg a)P(\neg c|\neg a) \\ &= \alpha \times 0.80 \times (1-0.20) \times (1-0.05) \\ &= \alpha \times 0.608\,, \end{aligned}$$

$$\begin{aligned} &\alpha = (0.032 + 0.608)^{-1} = 1.5625\,, \\ &P(A=1|B=0,\ C=0) = 0.05\,, \\ &P(A=0|B=0,\ C=0) = 0.95\,. \end{aligned}$$

In case a query "$P(a|\neg d, e) = ?$" arrives at this point, A would sample the distribution $P(a) = 0.05$ and, upon selecting a value 0, would provide the estimate

$$\hat{P}(a|\neg d, e) = \tfrac{1}{2}(1+0) = 0.5\,.$$

The second method would give:

$$\hat{P}(a|\neg d, e) = \tfrac{1}{2}(0.80 + 0.05) = 0.425\,.$$

The exact value of $P(a|\neg d, e)$ happens to be 0.097, and it takes over 100 runs to approximate this value to within 1% accuracy. Our choice of initial state $A = B = C = 1$ was an especially bad one; more reasonable starting states can be obtained by simulating the uninstantiated model in the forward direction, i.e., using $P(A)$, draw a value A_1 for A, using $P(B|A_1)$, draw a value for B; and so on.

This simulation scheme can also be used to find the most likely interpretation of the observed data, i.e., a joint assignment $\boldsymbol{w}^*$ of values to all variables in the system that, of all possible assignments, has the highest posterior probability, given the evidence. It is well known, e.g. [10], that the joint posterior probability is proportional to the product

$$P(\boldsymbol{w}|\text{evidence}) = \alpha \prod_i P(x_i|f_i)\,,$$

where i ranges over all variables in the system (including the data) and f_i is the state of X_i's parents, consistent with the assignment $\boldsymbol{w}$. Thus, the probability of any global state $\boldsymbol{w}$ entered by the simulation can be calculated by the product above and, by keeping records of the value that this product achieves with each sampling, the best state reached so far can be easily retrieved.

3. Justifying the Computations

Consider a typical neighborhood of variable X in some Bayesian network, as shown in Fig. 2: Define the following set of variables:

(1) X's parents $\boldsymbol{U}_X = \{U_1, \ldots, U_n\}$;
(2) X's children $\boldsymbol{Y}_X = \{Y_1, \ldots, Y_m\}$;
(3) $\boldsymbol{F}_j$, the set of parents of Y_j; and
(4) $\boldsymbol{W}_X = \boldsymbol{W} - X$, the set of all variables except X.

Theorem 3.1. *The probability distribution of each variable X in the network, conditioned on the state of all other variables, is given by the product*

$$P(x|\boldsymbol{w}_X) = \alpha P(x|\boldsymbol{u}_X) \prod_j P[y_j|f_j(x)] , \tag{2}$$

where α is a normalizing constant, independent of x, and x, $\boldsymbol{w}_X$, $\boldsymbol{u}_X$, y_j and $f_j(x)$ denote any consistent instantiations of X, $\boldsymbol{W}_X$, $\boldsymbol{U}_x$, Y_j and $\boldsymbol{F}_j$, respectively.

Thus, $P(x|\boldsymbol{w}_X)$ can be computed by simply taking the product of the instantiated link matrix stored at node X times those stored at X's children. In Fig. 2, for example, we have:

$$P(x|\boldsymbol{w}_X) = \alpha P(x|u_1, u_2) P(y_1|x, m_1) P(y_2|x, u_2, y_1, m_2) .$$

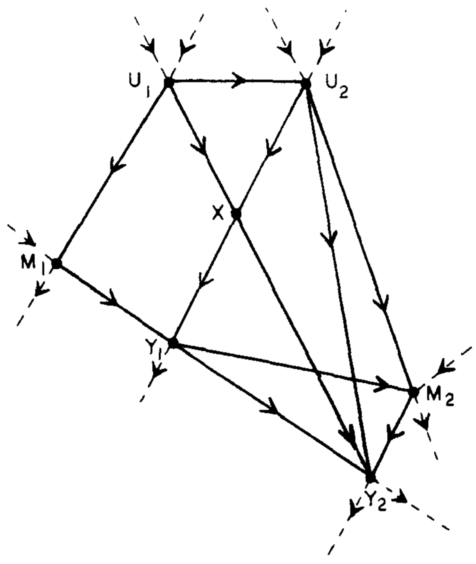

FIG. 2.

Proof. If we index the system's variables $\boldsymbol{W} = \{X_1, X_2, \ldots, X_i, \ldots\}$ by an ordering consistent with the orientation of the Bayes net, then the joint distribution of $\boldsymbol{W}$ can be written as a chain product:

$$\begin{aligned} P(\boldsymbol{w}) &= P(x_1, x_2, \ldots, x_i, \ldots) \\ &= P(x_1)(P(x_2|x_1)P(x_3|x_1, x_2)\cdots \\ &= \prod_i P(x_i|x_1, \ldots, x_{i-1}) \\ &= \prod_i P(x_i|f_i) \end{aligned} \tag{3}$$

where f_i stands for the values attained by X_i's parents. Equation (3) holds because, the parents of a variable X_i are defined as the set of predecessors that, once known, renders X_i independent on all its other predecessors [10]. Now consider a typical variable $X \in \boldsymbol{W}$, having n parents $\boldsymbol{U}_X$ and m children $\boldsymbol{Y}_X = \{Y_1, \ldots, Y_m\}$. X appears in exactly $m + 1$ factors of the product above; once in the factor $P(x|\boldsymbol{u}_X)$ and once in each $P(y_j|f_j)$ factor corresponding to the jth child of X. Thus, we can write

$$\begin{aligned} P(\boldsymbol{w}) &= P(x, \boldsymbol{w}_X) \\ &= P(x|\boldsymbol{u}_X) \prod_{j=1}^{m} P(y_j|f_j(x)) \prod_{k \in K} P(x_k|\sigma_k)\,, \end{aligned}$$

where σ_i stands for the values attained by X_i's parents. Now consider a typical variable

$$K = \{k\colon X_k \in \boldsymbol{W}_X - \boldsymbol{Y}_X\}\,.$$

Since x does not appear in the rightmost product (over k), the latter can be regarded as a constant α' relative to x, and we can write

$$P(x, \boldsymbol{w}_X) = \alpha' P(x|\boldsymbol{u}_X)\prod_j (y_j|f_j(x))\,.$$

Moreover, since

$$P(\boldsymbol{w}_X) = \sum_x P(x, \boldsymbol{w}_X)$$

is also a constant relative to x, we have

$$P(x|\boldsymbol{w}_X) = \frac{P(x, \boldsymbol{w}_X)}{P(\boldsymbol{w}_X)} = \alpha P(x|\boldsymbol{u}_X)\prod_j P(y_j|f_j(x))\,,$$

which proves the theorem. □

The main significance of Theorem 3.1 is that $P(x|w_X)$ is computed as a product of parameters which are stored with the specification of the model. Thus, the parameters are readily available locally and the computations are extremely simple.

4. Distributed Control of Concurrent Activation

The simulation process can also be executed in parallel but requires some scheduling to keep neighboring processors from operating at the same time. To see why this is necessary, imagine two neighboring processors, X and Y, entering the computation phase at the same time t_1. X observes the value y_1 of Y and calculates $P(x|y_1)$ while, at the same time, Y observes the value x_1 of X and calculates $P(y|x_1)$. At a later time, t_2, they enter the simulation phase with X instantiated to a sample x_2 drawn from $P(x|y_1)$ and Y to a sample y_2 drawn from $P(y|x_1)$. The new values x_2 and y_2 are not compatible with the distribution P. P was consulted to match y_2 with x_1 (and x_2 with y_1), but now that X has changed its value to x_2, y_2 no longer represents a proper probabilistic match to it.

To formalize this notion, note that a prerequisite to coherent relaxation is the stationarity of the distribution of X and Y. In other words, we require that, if at time t_1 X and Y are distributed by $P(x, y)$, then the values of X and Y at time t_2 must also be distributed by $P(x, y)$. This requirement is met when only one variable changes at any given time because then (assuming Y is the changing variable), we can write:

$$\begin{aligned}
&P(X_2 = x, Y_2 = y) \\
&\quad = \sum_{x'y'} P(X_2 = x, Y_2 = y | X_1 = x', Y_1 = y')P(x', y') \\
&\quad = P(Y_1 = y | X_1 = x)P(X_1 = x) \\
&\quad = P(X_1 = x, Y_1 = y) = P(x, y) ,
\end{aligned} \tag{4}$$

which implies stationarity. If, however, X and Y change their values simultaneously, we have

$$\begin{aligned}
&P(X_2 = x, Y_2 = y) \\
&\quad = P(Y_2 = y | X_2 = x)P(X_2 = x) \\
&\quad = \sum_{x'y'} P(X_2 = x, Y_2 = y | X_1 = x', Y' = y')P(x', y') \\
&\quad = \sum_{x'y'} P(X_1 = x | Y_1 = y')P(Y_1 = y | X_1 = x')P(x', y') \\
&\quad = \sum_{x'y'} \frac{P(x, y')}{P(y')} \frac{P(x', y)}{P(x')} P(x', y') ,
\end{aligned} \tag{5}$$

which does not represent stationarity except in the pathological case where X_1 and Y_1 are independent.

This analysis can be extended to the multi-variable case and allows us to determine which variables can be activated simultaneously. Let the set of concurrently activated variables be $\mathbf{Z} = \{Z_1, Z_2, \ldots, Z_n\}$, and assume each Z_i variable chooses a new value z'_i by sampling the distribution $P(z_i|\mathbf{s}_i)$, where $\mathbf{S}_i$ is the subset of variables inspected by Z_i prior to switching. If $\mathbf{W_Z}$ stands for the set of unchanged variables, then stationarity requires

$$P(z', \mathbf{w_Z}) = P(z, \mathbf{w_Z}) \quad \text{or} \quad P(z'|\mathbf{w_Z}) = P(z|\mathbf{w_Z}) \tag{6}$$

because $P(\mathbf{w_Z})$ remains unchanged in the transition.

Since the values z' of the $\mathbf{Z}$ variables are drawn independently from $P(z_i|\mathbf{s}_i)$, equation (6) translates to:

$$\prod_{i=1}^{n} P(Z_i = z_i|\mathbf{s}_i) = P(z_1, z_2, \ldots, z_n|\mathbf{w_Z}) . \tag{7}$$

This requirement is satisfied whenever each $\mathbf{S}_i$ is a Markov blanket of Z_i, i.e.,

$$P(z_i|\mathbf{s}_i) = P(z_i|\mathbf{w}_{Z_i}) \tag{8}$$

and, simultaneously, each $\mathbf{S}_i$ shields Z_i from all other Z,

$$P(z_i|\mathbf{s}_i) = P\left(z_i|\mathbf{s}_i \bigwedge_{j\neq i} z_j\right), \quad i = 1, 2, \ldots, n . \tag{9}$$

To meet both equations (8) and (9), it is clear that, if $\mathbf{S}_i$ contains any of the Z_j, then $\mathbf{S}_i - Z_j$ must also shield Z_i from all other Z. However, if we assume that each $\mathbf{S}_i$ is already the smallest Markov blanket permitted by the network, we must conclude that no Z_j should be a member in any of the $\mathbf{S}_i$. Thus, any set of variables licensed to be activated simultaneously, must not contain a pair belonging to the same Markov blanket.

A convenient way to highlight this requirement is to add dummy links between mates (i.e., nodes sharing a child), taking care that no two adjacent nodes in the thus-augmented network are activated concurrently. The question now arises how to schedule the activation of the processors in such a way that:

(1) no two adjacent processors are activated at the same time;

(2) every processor gets activated sufficiently often;

(3) the activation commands are generated distributedly, with no external supervision.

This problem is a version of the "dining philosophers" dilemma originally posed by Dijkstra [5] and later solved independently by Gafni and Bertsekas [6] and Chandy and Misra [3]. The solution is a distributed control policy called "edge reversal" and involves the following steps:

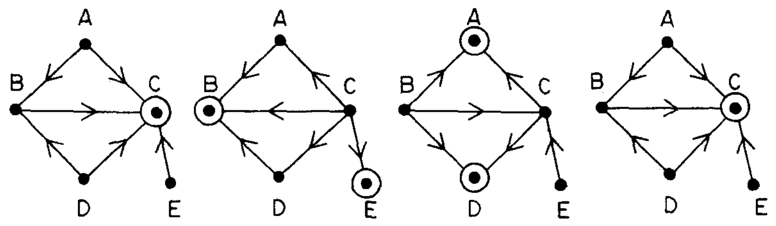

FIG. 3.

(1) Initially, the links of the network are assigned arbitrary acyclic orientation of arrows. (This orientation bears no relation to the causal ordering governing the construction of Bayesian networks.)

(2) Each processor inspects the orientation of the arrows on its incident links and waits until all arrows point inward, i.e., until the processor becomes a *sink*.

(3) Once a processor becomes a sink, it is activated and, when it completes the computation, reverses the direction of all its incident arrows (i.e., it becomes a *source*).

It is easily seen that no two neighbors can be activated at the same time. What is more remarkable about this edge-reversal policy, though, is that no processor ever gets "deprived"; every processor fires at least once before the orientation returns to some previous state and the cycle repeats itself. Moreover, every processor is activated the same number of times in any such cycle [1]. The "nondeprivation" feature is important because it constitutes a necessary condition for the convergence of the entire process [7]. As time progresses, the system is guaranteed to reach a steady state in the sense that, regardless of the initial instantiation, the probability that the system will enter any global state w is given by the joint distribution specified by the link matrices.

Figure 3 illustrates this policy on the Bayesian net of Fig. 2 by marking (with circles) the nodes activated at each step of the process. Initially, the dummy edge BC is added to designate these mates as neighbors, and the orientation of Fig. 3(a) is assigned, where C is the only sink. Once C is activated, the arrows pointing to C are reversed (by C), B and E become sinks and fire. After three steps, Fig. 3(d), the orientation is back where it started, and the cycle repeats. Note that every processor fires once during the cycle and that twice, depicted in Figs. 3(b) and 3(c), we had two processors firing simultaneously. The problem of achieving maximum concurrency with edge reversal was analyzed by Barbosa [1].

5. Conclusions

The local and concurrent scheme presented in this paper renders stochastic simulation a viable inferencing technique for evidential reasoning tasks. Although dozens of runs are necessary for achieving reasonable levels of accuracy, each run requires only $|V| + |E|$ computational steps, where $|V|$ is the number of vertices in the model and $|E|$ is the number of edges. Unlike purely numerical techniques, which sometimes require exponential complexity, the length of computation is determined mainly by the required degree of accuracy, not by the dependencies embodied in the model. It is postulated, therefore, that stochastic simulation will be found practical in applications involving complex models with highly interdependent variables and where "ballpark" estimates of probabilities will suffice.

ACKNOWLEDGMENT

I am indebted to Max Henrion for rekindling my interest in stochastic simulation and helping me regain confidence in its viability as a practical inferencing technique. Also appreciated are the discussions with Eli Gafni and Valmir Barbosa who, during the latter's oral examination, pointed out both the danger of concurrently activating neighboring variables and the remarkable features of the edge-reversal policy.

REFERENCES

1. Barbosa, V.C., Concurrency in systems with neighborhood constraints, Ph.D. Dissertation, Computer Science Department, UCLA, Los Angeles, CA, 1986.
2. Bundy, A., Incidence calculus: A mechanism for probabilistic reasoning, in: *Proceedings Workshop on Uncertainty in Artificial Intelligence*, UCLA, Los Angeles, CA (1985) 177–184.
3. Chandy, K.M. and Misra, J., The drinking philosophers problem, *ACM Trans. Program. Lang. Syst.* **G** (4) (1984) 632–646.
4. Cooper, G.F., NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge, Ph.D. Dissertation, Department of Computer Science, Stanford University, Stanford, CA, 1984.
5. Dijkstra, E.W., Hierarchical ordering of sequential processes, in: C.A.R. Hoare and R.H. Perrott (Eds.), *Operating Systems Techniques* (Academic Press, New York, 1972).
6. Gafni, E.M. and Bertsekas, D.P., Distributed algorithms for generating loop-free routes in networks with frequently changing topology, *IEEE Trans. Commun.* **29** (1) (1981) 11–18.
7. Geman, S. and Geman, D., Stochastic relaxations, Gibbs distributions and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.* **6** (6) (1984) 721–742.
8. Henrion, M., Propagating uncertainty by logic sampling in Bayes' networks, in: *Proceedings Workshop on Uncertainty in AI*, Philadelphia, PA (1986).
9. Hinton, G.E., Sejnowski, T.J. and Ackley, D.H., Boltzman machines: Constraint satisfaction networks that learn, Tech. Rept. CMU-CS-84-119, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1984.
10. Pearl, J., Fusion, propagation and structuring in belief networks, *Artificial Intelligence* **29** (3) (1986) 241–288.
11. Spiegelhalter, D.J., Probabilistic reasoning in predictive expert systems, in: L.N. Kanal and J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence* (North-Holland, Amsterdam, 1986) 47–68.

Received September 1986; revised version received January 1987