# A Randomized Approximation Algorithm for Probabilistic Inference on Bayesian Belief Networks

**R. Martin Chavez and Gregory F. Cooper**
*Section on Medical Informatics, Stanford University School of Medicine, Stanford, California 94305*

Researchers in decision analysis and artificial intelligence (AI) have used Bayesian belief networks to build probabilistic expert systems. Using standard methods drawn from the theory of computational complexity, workers in the field have shown that the problem of probabilistic inference in belief networks is difficult and almost certainly intractable. We have developed a randomized approximation scheme, BN-RAS, for doing probabilistic inference in belief networks. The algorithm can, in many circumstances, perform efficient approximate inference in large and richly interconnected models. Unlike previously described stochastic algorithms for probabilistic inference, the randomized approximation scheme (ras) computes a priori bounds on running time by analyzing the structure and contents of the belief network. In this article, we describe BN-RAS precisely and analyze its performance mathematically.

## 1. INTRODUCTION

Recent work in expert systems and decision analysis has increased interest in Bayesian probabilistic techniques for the representation of knowledge. This article describes a randomized approximation scheme (ras), which we have named BN-RAS, for the Bayesian inference problem in belief networks. BN-RAS computes posterior probabilities, with high likelihood of success, to within prespecified relative or interval errors. A corresponding analysis of running time characterizes the strengths and weaknesses of the method and explicitly characterizes computational resources in terms of known quantities and error tolerances.

Within the discipline of artificial intelligence, many researchers have studied methodologies for encoding the knowledge of experts as computational artifacts. General purpose environments for constructing probabilistic, knowledge-intensive systems based on belief networks have been developed [3]. Such networks serve as partial graphical representations for decision models, in which the knowledge engineer must define clearly the alternatives, states, preferences, and relationships that constitute a decision basis.

NETWORKS, Vol. 20 (1990) 661–685

CCC 0028-3045/90/050661-025$04.00

The problem of Probabilistic Inference in Belief NETworks (PIBNET) is hard for $\#P$, and therefore hard for $\mathcal{NP}$ [7]. In a later section, we describe the complexity classes $\#P$ and $\mathcal{NP}$ in more detail. The classification of PIBNET as $\mathcal{NP}$-hard has prompted a shift in focus away from deterministic algorithms and toward approximate methods (including stochastic simulations), heuristics, and analyses of average-case behavior.

Stochastic simulations compute probabilities by estimating the frequency of a given scenario in a large space of possible outcomes. Belief networks provide frameworks for the generation of hypothetical outcomes, also known as *trials*, in stochastic simulations. In this article, we offer a complexity-theoretic treatment of approximate probabilistic inference. We use methods drawn from the analysis of ergodic Markov chains and randomized complexity theory to build an algorithm that, in many cases, efficiently approximates the solutions of inference problems for belief networks to arbitrary precision.

We shall summarize previous work on belief networks as a paradigm for knowledge representation. We shall then discuss earlier work on stochastic simulation and logic sampling [18]. In particular, we shall examine Pearl's method, known hereafter as *straight simulation*, for performing probabilistic inference in belief networks. Finally, we shall propose and analyze a randomized approximation scheme for probabilistic inference.

### 1.1 Belief Networks

Belief networks provide high-level graphical representations of expert opinion. More precisely, a belief network is a directed acyclic graph wherein nodes represent domain variables and edges represent causal or associational relationships [16]. For each orphan node (a domain variable that lacks incoming arcs of preconditioning influences), the network designer must specify a prior-probability distribution. For all other nodes, the expert must assess a probability mass function (p.m.f.) conditioned on the node's parents. In general, the size of a node's p.m.f. increases exponentially with the number of incoming arcs. We shall restrict our coverage here to belief networks that model random variables with discrete outcomes.

Let $\Sigma$ denote an alphabet of symbols. In formal terms, a belief network $\mathcal{B}$ is a tuple $(V,O,A,\Pi)$, where $V \subseteq \Sigma^*$ is the set of domain variables (the labels of the network), $O \subseteq \Sigma^*$ is a set of outcomes, $A : V \rightarrow 2^V$ is a function that describes the predecessors of each node, and $\Pi : (V \times O) \times 2^{(V \times O)} \rightarrow \mathcal{R}$ specifies conditional probabilities for each node, given its predecessors. A well-formed belief network has no cycles; the relation $\Pi$ must specify exactly the probabilities implied by the topology of the network, and

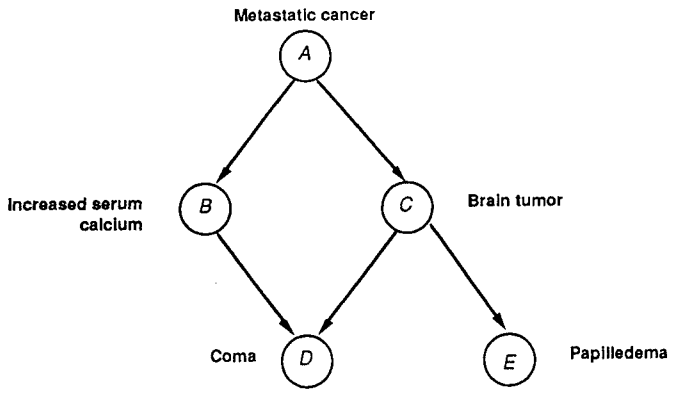
$$\forall X \in V : \sum_{v \in O} \Pi((X,v),s) = 1.0 , \tag{1}$$

where $s$ is a function from $A(X)$ to $O$ such that the cardinality of $s$ is equal to the cardinality of $A(X)$ (i.e., $s$ assigns exactly one outcome to each ancestor of

$X$). The *size* of a belief network $\mathcal{B}$ is the length of an encoding of the tuple $(V,O,A,\Pi)$, where the probabilities in $\Pi$ are represented in unary notation.

Every belief network represents a joint-probability space. Partially connected belief networks also capture intuitive notions of independence; the absence of an arrow implies a precise statement of conditional independence. Those explicitly represented assumptions can greatly reduce the complexity of representing and reasoning with the complete joint p.m.f. over all the variables in the network. For modeling problems that specify singly connected belief networks (i.e., networks that contain at most one undirected path between any pair of nodes), the exact methods compute a probability assignment for all nodes in time proportional to the diameter of the network on parallel machines and in $O(n)$ time on sequential machines, where $n$ denotes the number of nodes. In the worst case, however, PIBNET seems to require exponential time. All known exact algorithms for inference on multiply connected networks operate in exponential time with respect to the number of nodes, in the worst case [19].

Figure 1 contains a small belief network from the domain of medicine; it is intended for illustrative purposes only and not as an accurate medical model. The probability that a patient has a *brain tumor* (i.e., $C = T$) given that he has *metastatic cancer* (i.e., $A = T$) is 20%. The absence of an arc from $B$ to $C$ implies the conditional independence of *increased total serum calcium* and *brain tumor* given *metastatic cancer*. In other words, $P(B = T,C = T|A = T) = P(B =$




$P(A=T) = 0.001$

$P(B=T \mid A=T) = 0.3$
$P(B=T \mid A=F) = 0.001$

$P(C=T \mid A=T) = 0.2$
$P(C=T \mid A=F) = 0.00005$

$P(D=T \mid B=T,C=T) = 0.1$
$P(D=T \mid B=T,C=F) = 0.01$
$P(D=T \mid B=F,C=T) = 0.05$
$P(D=T \mid B=F,C=F) = 0.00001$

$P(E=T \mid C=T) = 0.4$
$P(E=T \mid C=F) = 0.002$

FIG. 1. This belief network captures information about metastatic carcinoma and serum calcium. (It has been greatly simplified for purposes of illustration, and does not convey a completely realistic representation of the causal-associational relationships.)

$T|A = T) \cdot P(C = T|A = T)$ or $P(B = T|C = T, A = T) = P(B = T|A = T)$. The designer of a belief network specifies a conditional probability distribution for each node given its parents.

## 1.2. Computational Complexity Theory

For a general introduction to computational complexity, see the bibliography in [14]. We take our model of sequential computation to be a Turing machine. Let $\mathcal{P}$ denote the class of problems that can be solved in an amount of time that is a polynomial function of the size of the problem. Problems are usually represented as an infinite set of strings that characterize problem instances, where each instance represents a "yes–no" question. A polynomial-time decision algorithm for a given problem answers the question posed by an arbitrary problem instance after computing for no more than $O(n^k)$ time steps, where $n$ represents the size of the instance and $k$ denotes a fixed constant. We transform calculation problems into decision problems by asking not what the exact solution is, but rather whether the numerical solution lies below a given bound.

### 1.2.1. *Nondeterministic Polynomial Time*

$\mathcal{NP}$ is the class of problems that can be solved nondeterministically in polynomial time on a machine that can compute along multiple paths simultaneously. Equivalently, $\mathcal{NP}$ contains those problems for which a Turing machine can guess a solution and then verify the solution's correctness in deterministic polynomial time. $\mathcal{PSPACE}$ is the class of all problems that can be decided in computational space that is a polynomial in the size of the problem. Inasmuch as a computation that requires $T(n)$ time can use no more than $T(n)$ space, the class $\mathcal{PSPACE}$ contains both $\mathcal{NP}$ and $\mathcal{P}$. Whether $\mathcal{PSPACE}$ properly contains $\mathcal{NP}$ and $\mathcal{NP}$ properly contains $\mathcal{P}$, however, remain to be determined.

### 1.2.2. *$\mathcal{NP}$-Complete Problems and Reducibility*

Let $A$ be a problem that belongs to a class $\mathcal{C}$ of problems. We say that $A$ is *complete* for $\mathcal{C}$ if there exists a construction that reduces every instance of every problem in $\mathcal{C}$ to an instance of $A$ using space logarithmic in the size of the instance. In some sense, then, $A$ captures the maximal complexity of the class $\mathcal{C}$; $A$ is at least as difficult to decide, for arbitrary problem instances, as is any other problem in the class. If every problem in $\mathcal{C}$ is reducible to $A$, but $A$ is not known to belong to $\mathcal{C}$, we say that $A$ is *hard* for $\mathcal{C}$.

The hardest problems in $\mathcal{NP}$, which are labeled $\mathcal{NP}$-complete, probably do not admit polynomial-time deterministic algorithms [8]. If any of the $\mathcal{NP}$-complete problems admits a polynomial-time deterministic algorithm, then $\mathcal{P} = \mathcal{NP}$ and all problems in $\mathcal{NP}$ can be solved in polynomial time. The vast majority of theoreticians believe, however, that $\mathcal{NP}$ properly contains $\mathcal{P}$ and that polynomial-time algorithms for $\mathcal{NP}$-complete problems do not exist.

#### 1.2.3. *PIBNET and PIBNETD*

An algorithm for the PIBNET decision (PIBNETD) problem determines, for some variable $Y$ in a given belief network $\mathcal{B} = (V,\{T,F\},A,\Pi)$, whether $\Pr[Y = T] > 0$. PIBNETD is known to be complete for the class $\mathcal{NP}$ via a reduction from 3-satisfiability [7,6], where the size of the problem is determined by a natural encoding of the tuple $\mathcal{B}$. An algorithm for the PIBNET problem computes, for a given variable $Y$, the quantity $\Pr[Y = T]$.

Inasmuch as we can reduce PIBNETD to PIBNET in logarithmic space, we know that PIBNET is hard for $\mathcal{NP}$. The problem of solving decision networks, which contain decision nodes as well as chance nodes, is hard for $\mathcal{PSPACE}$ (the class of problems solvable in polynomial space) and therefore is at least as hard as, if not harder than, that for PIBNET [15].

#### 1.2.4. *The Class #P*

$\#P$ is the set comprising integer-valued functions that count the number of accepting computations of some nondeterministic polynomial-time Turing machine [23]. The counting versions of most $\mathcal{NP}$-complete problems, in particular, are complete for $\#P$. The $\#P$-complete problems are at least as hard as the $\mathcal{NP}$-complete problems and perhaps are harder. Given an algorithm that for a $\#P$-complete problem, we can always generate a related algorithm for the naturally corresponding $\mathcal{NP}$-complete problem by answering "yes" when the counting method returns 1 or greater and "no" otherwise. Examples of $\#P$-complete problems include counting the number of Hamiltonian circuits in a graph, evaluating the probability that a probabilistic graph is connected, computing the permanent of a matrix, and counting the perfect matchings in a bipartite graph [8].

PIBNET is hard for $\#P$ [7]. We therefore do not expect efficient polynomial-time solutions for the exact problem. Specific belief networks or classes of belief networks may have efficient polynomial-time algorithms, but the discovery of a general polynomial-time algorithm for PIBNET would imply that $\mathcal{P} = \mathcal{NP}$. We focus instead on efficient randomized algorithms that, with high probability, approximate the desired solution to within some prespecified error for a wide variety of networks.

## 2. ALGORITHMS FOR BELIEF NETWORKS

Current algorithms for probabilistic inference on belief networks can be placed into three categories: the exact methods, which deterministically calculate posterior probabilities; logic sampling and straight simulation, which apply Monte Carlo techniques to the inference problem but do not give a priori bounds on running time; and the PIBNET algorithm BN-RAS.

### 2.1 Exact Methods

Several algorithms have been developed for computing posterior probabilities of nodes in belief networks contingent on a set of evidence, including the

message-passing algorithm of Pearl [16], the clique-triangulation method of Lauritzen and Spiegelhalter [13], and the arc-reversal technique of Shachter [20]. In the worst case, all known deterministic techniques experience exponentially increasing time complexity as the intricacy of the causal model increases.

### 2.2 Logic Sampling and Straight Simulation

The *incidence calculus* [2] represents samples of the joint probability space as bit strings and defines axiomatized procedures for deriving new bit strings. More recently, the scheme of *logic sampling* has applied the incidence calculus to Bayesian belief networks [9]. In logic sampling, belief networks serve as simulated trial generators. The algorithm calculates belief destributions by averaging over those trials that correspond to the observed data. Logic sampling, however, may generate a large fraction of irrelevant trials. Recent modifications of logic sampling reduce the posterior variance with a technique known as *importance sampling* [21]. Straight simulation, which employs local numerical computation followed by logic sampling, effectively clamps observed nodes to the appropriate values and thereby avoids generating useless samples [17].

Straight simulation has two undesirable properties. First, the method does not bound the number of required trials in terms of known quantities. To our knowledge, no previous simulation methods (including logic sampling and straight simulation) offer a priori upper bounds on the amount of computation that will guarantee sufficient convergence properties. Second, the theory of random Markov fields guarantees convergence to the correct stationary distribution in the limit; there is no reason, however, to believe that straight simulation converges rapidly or efficiently to that distribution [5]. Empirical testing of straight simulation reveals a serious difficulty: As the network's probabilities approach the boundaries at 0 and 1, the number of simulated trials needed to obtain convergence increases without bound [4].

We address the first issue with area-estimation techniques inspired by Karp and Luby [12]. We assume, for simplicity, the existence of an ideal trial generator that enumerates states of the network according to their exact probabilities. By modifying the scoring strategy and assuming the existence of an ideal trial generator, we build a randomized algorithm that, with high likelihood, guarantees a fixed relative-error tolerance. Chebyshev's inequality allows us to compute the requisite number of trials as a function of known quantities. A similar analysis sets a lower bound on the amount of computation required to guarantee a prespecified interval error.

We analyze the second problem with techniques developed by Jerrum and Sinclair [11]. Recognizing that a trial generator probably cannot enumerate combinatoric objects according to their precise probabilities in polynomial time [22], we bound the discrepancy between a modified generator's multistep transition probabilities and the true probabilities of the stationary distribution. The accuracy of the generator increases with the number of transitions that we allow for thorough mixing.

Finally, we combine the modified trial generator with stochastic simulation

in order to define an ras for PIBNET. The resulting BN-RAS calculates posterior probabilities, with high likelihood, to within pre-specified relative and interval tolerances. The running time depends on the tolerances, the logarithm of the probability of achieving those tolerances, the smallest joint probability over all nodes in the network, and the smallest Markov transition probability.

The deterministic methods for probabilistic inference typically exhibit exponential behavior as the topological complexity of the network increases. BN-RAS replaces the exact methods' constraints on topological complexity with a dependency on the smallest transition probability, which dominates the running time. Although our methods cannot perform efficient inference on all belief networks, they nevertheless expand the class of problems for which tractable approximations can be considered. To our knowledge, the methods we describe yield the first precisely characterized approximation algorithm for large and richly connected belief networks.

We use techiques developed by Karp and Luby to analyze the time complexity of straight simulation [12]. The analytic treatment confirms previous empirical observations. Then, we present an approximate PIBNET algorithm, known as BN-RAS, based on the bounded convergence of an ergodic Markov chain and on Monte Carlo area-estimation strategies. Finally, we compute the new algorithm's time complexity in terms of problem size and error tolerance.

## 3. THE BN-RAS ALGORITHM

We briefly recapitulate Karp's and Luby's analysis of Monte Carlo area estimation in the Euclidean plane and apply it to the analysis of straight simulation. The area-estimation strategy offers a simple method for bounding the variance and convergence of Monte Carlo algorithms and establishes the overall framework for our analysis.

### 3.1. Monte Carlo Area Estimation

Suppose a region $E$ of known area $A(E)$ encloses the region $U$ of unknown area $A(U)$ in the plane. Suppose, furthermore, that region $E$ contains exactly $b$ blocks, each with area $c_i$. The region $U$ contains some of those blocks, but not others. A block belongs entirely to $U$, or not at all to $U$. Specifically,

$$\alpha_i = \begin{cases} 1 & \text{if block } i \in U; \\ 0 & \text{if block } i \notin U. \end{cases}$$

In other words, $A(E) = \Sigma_{i=1}^{b} c_i$ and $A(U) = \Sigma_{i=1}^{b} \alpha_i \cdot c_i$. We could calculate $A(U)$ by computing the sum directly; for very large $b$, however, we wish to avoid costly summations.

The algorithm of Karp and Luby instead presupposes that we have a method randomly to choose block $i$ with probability $c_i/A(E)$. In each trial of the simulation, randomly select block $i$ and compute an estimator $Y = \alpha_i \cdot A(E)$. Inas-

much as

$$E[Y] = \sum_{i=1}^{b} \frac{c_i}{A(E)} \alpha_i \cdot A(E) = A(U) , \tag{2}$$

the random variable $Y$ estimates $\mu = A(U)$ without bias. We reduce the variance by computing multiple estimators $Y_j$ corresponding to trial $j$ and defining the unbiased estimator over multiple trials,

$$\tilde{Y} = \frac{Y_1 + \cdots + Y_N}{N}.$$

Karp and Luby analyze the variance of the Monte Carlo algorithm repeated $N$ times [12]. They define the *relative error*

$$\left| \frac{\tilde{Y} - \mu}{\mu} \right| ,$$

where $\mu$ represents the value of the quantity to be estimated. Then, they derive an upper bound on the number of trials $N$ that guarantees, with probability greater than a specified $1 - \delta$, a relative error less than a specified $\epsilon$. For example, if we set $\epsilon = 0.05$ and $\delta = 0.1$, we must repeat the trial often enough to guarantee a relative error that exceeds 5% no more than 10% of the time. Any Monte Carlo algorithm that exhibits such convergence properties belongs to the class of $(\epsilon,\delta)$ approximation algorithms, by definition.

We can trivially define a distribution-free upper bound on $N$. Let $\sigma^2$ signify the variance of $Y$, the value obtained in a single Monte Carlo trial. Then, the variance of $\tilde{Y}$ is $\sigma^2/N$. By Chebyshev's inequality,

$$\Pr\left[ \left| \frac{\tilde{Y} - \mu}{\mu} \right| > \epsilon \right] = \Pr\left[ |\tilde{Y} - \mu| > \epsilon\mu \right] \leq \frac{\sigma^2}{N\epsilon^2\mu^2} .$$

Thus, to ensure that $\Pr[|(\tilde{Y} - \mu)/\mu| > \epsilon]$ not exceed $\delta$, the condition

$$N \geq \frac{\sigma^2}{\mu^2} \cdot \frac{1}{\delta\epsilon^2}$$

suffices.

For the area-estimation algorithm, $E[Y] = A(U)$ and

$$E[Y^2] = \sum_{i=1}^{b} \frac{c_i}{A(E)} \alpha_i^2 A(E)^2$$

$$= A(E) \sum_{i=1}^{b} c_i \alpha_i = A(E)A(U) .$$

Therefore,

$$\sigma^2 = E[Y^2] - E[Y]^2 = A(E) \cdot A(U) - A(U)^2 ,$$

and

$$\frac{\sigma^2}{\mu^2} = \frac{A(E)}{A(U)} - 1 .$$

Finally,

$$N \geq \left(\frac{A(E)}{A(U)} - 1\right) \cdot \frac{1}{\delta\epsilon^2} \tag{3}$$

guarantees an $(\epsilon,\delta)$ algorithm for any p.m.f. on $Y$. Clearly, we seek estimation algorithms with ratios $A(E)/A(U)$ close to 1. Ideally, the regions $E$ should not be much larger than the region $U$ of interest.

If we impose less stringent constraints by using interval (rather than relative) error, fewer trials will be required, unless $A(E) = A(U)$. To guarantee that

$$\Pr\left[|\tilde{Y} - \mu| > \alpha\right] < \frac{\sigma^2}{N\alpha^2},$$

we set

$$N \geq \frac{\sigma^2}{\alpha^2\delta} = \frac{A(U)(A(E) - A(U))}{\alpha^2\delta} . \tag{4}$$

An algorithm that bounds interval error $\alpha$ with success probability $1 - \delta$ is known as an $(\alpha,\delta)$ approximation scheme, by definition.

In the next section, we show how to construct regions $U$ and $E$ as collections of blocks that represent the probabilities of interest in a Markov simulation for Bayesian inference. We then apply the analysis of Karp and Luby to probabilistic inference. Finally, we examine the difficult problem of randomly choosing blocks from the joint-probability distribution.

### 3.2. Straight Simulation for Belief Networks

In this section, we describe the crucial properties of belief networks that make Monte Carlo sampling possible. We present the Monte Carlo method in detail and then discuss its convergence properties.

#### 3.2.1. *The Local-Computation Property*

Straight simulation depends crucially on the property of local computation in belief networks. More precisely, the probability distribution of each variable $X$

in the network, conditioned on the state of all other variables, can be written

$$\Pr[X|\mathbf{w}_X] = \alpha \cdot \Pr[X|\mathbf{u}_X] \prod_j \Pr[y_j|\mathbf{f}_j(x)] . \tag{5}$$

Here and elsewhere, upper-case letters refer to nodes with arbitrary values; lower-case letters denote instantiated nodes. The scaling factor $\alpha$ normalizes the p.m.f., the $\mathbf{u}_X$ represents the parents of $X$, the $y_j$ represent the children of $X$, the $\mathbf{f}_j(x)$ include $X$ and the other parents of $X$'s children, and the $\mathbf{w}_X$ represent all variables except $X$ [17].

Knowing the instantiations for the parents of $X$, the children of $X$, and the other parents of the children of $X$ (which nodes define the *Markov blanket* $M$ of $X$), we can compute the posterior distribution for $X$. The Markov blanket of $X$ probabilistically shields $X$ from the rest of the network; no other information is required to compute a p.m.f. over the values of $X$.

### 3.2.2. *Straight-Simulation Methodology*

Straight simulation operates in the following manner. First, clamp observed nodes to their observed values. Set the unobserved nodes to random values. Choose an arbitrary order $X_1, \ldots, X_n$ in which to enumerate the unobserved nodes. In one trial, begin with the first enumerated node $X = X_1$ and compute a p.m.f. $\Pr[X|\mathbf{w}_X]$ based on the instantiation of $X$'s Markov blanket. Sample from the computed p.m.f. and set $X_1$ to the sampled value. Sequentially repeat the procedure for $X_2, \ldots, X_m$ to complete one trial. Record the sampled values chosen in each trial; repeat the sampling for $N$ trials. The frequency of recorded values after $N$ trials approximates the posterior p.m.f. at each node. Figure 2 gives the Pascal pseudocode that computes individual transition probabilities; Figure 3 presents the pseudocode for straight simulation.

```
PROCEDURE sample(VAR X: node);
{Compute a transition p.m.f.}
VAR
   sum, prod: REAL;
   v, i:      INTEGER;
BEGIN
   sum := 0.0;
   FOR X.value := 1 TO X.nvalues DO
      BEGIN
      prod := cprob(X);   {cprob(X) = Pr[X | u_X]}
      FOR i := 0 TO X.nchildren DO
         prod := prod * cprob(X.children[i]);   {cprob(X.children[i]) = Pr[y_j | f_j(x)]}
      sum := sum + prod;
      X.dist[X.value] := sum;
      END;
    normalize dist;
END;
```

FIG. 2. Pseudocode that computes the Markov transition probabilities from a belief network.

```
PROGRAM straightsim;
BEGIN
VAR
    nodes: ARRAY 1..MAXNODES OF node;
    current_node: INTEGER;

PROCEDURE do_transition(VAR X: node);
{Compute a transition probability, and assign a value to node i}
BEGIN
    sample(X);
    X.value := choose(X.dist);
    {select a value in 1..X.nvalues according to the cumulative p.m.f. in X.dist}
END;

PROCEDURE next_transition;
{Compute a transition}
BEGIN
   {Do not reinitialize the nodes}
   IF current_node = nnodes THEN
      BEGIN
      score outcomes;
      current_node := 1;
      END;
   do_transition(nodes[current_node]);
   current_node := current_node+1;
END;

PROCEDURE estimate;
VAR j, n: INTEGER;
BEGIN
   compute n, the number of transitions;
   current_node := 1;
   FOR j := 1 TO n DO
      next_trial;
END;
END.
```

FIG. 3. Pseudocode for straight simulation.

### 3.2.3. *Straight-Simulation Convergence Properties*

Empiric testing reveals that straight simulation converges very slowly as the computed frequencies approach 0 or 1 [5]. We derive the same result analytically by applying the techniques of Karp and Luby to straight simulation.

Assume for the moment that sequential sampling, based on local computation at each node's Markov blanket, generates network instantiations *cs* (otherwise known as *complete states*) with probability $\Pr[cs|\xi]$ exactly, where $\xi$ represents the observed variables and the set *cs* specifies a value assignment for each domain variable. In actuality, the generator does not immediately produce states according to the exact distribution; only in the limit at infinity does the generator reach the stationary distribution [11]. In the next section, we shall correct that defect and describe a simulator with a stationary distribution that differs from $\Pr[cs|\xi]$ by a bounded relative error. For now, however, the presumed existence of an accurate trial generator greatly simplifies the analysis.

Let $X_1, \ldots, X_m$ denote the unobserved variables; without loss of generality,

let each node have two outcomes, $T$ and $F$. Write $cs$ as a set of $m$ terms $\{(X_1 \leftarrow v_1), \ldots, (X_m \leftarrow v_m)\}$, where each $v_i$ is one of $\{T,F\}$.

In each trial $j$, generate an assignment $cs$ based on prior information $\xi$ and compute estimators

$$Y[X_i,v]_j = \begin{cases} 1 & \text{if } (X_i \leftarrow v) \in cs\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

where $v$ is either $T$ or $F$.

Finally, at the end of $N$ trials, compute the averages

$$\bar{Y}[X_i,v] = \frac{1}{N}\sum_{j=1}^{N} Y[X_i,v]_j\,.$$

Observe that

$$E[Y[X_i,v]] = \sum_{cs:(X_i \rightarrow v) \in cs} \Pr[cs|\xi]$$

$$= \Pr[X_i = v|\xi]\,.$$

By performing many trials, we can reduce the variance of the unbiased estimators $Y[X_i,v]$.

We now analyze the convergence of each estimator $Y[X_i,v]$ separately. By analogy to the area-estimation algorithm, states $s$ consistent with $\xi$ correspond to subdividing blocks. The containing region $E$ represents the set of all states $s$ consistent with $\xi$. By extension,

$$A(E) = \sum_{cs} \Pr[cs|\xi] = 1\,.$$

Similarly, $U_{X_i,v}$ corresponds to the states in which $X_i$ takes on value $v$. The Monte Carlo simulation aims to compute

$$A(U_{X_i,v}) = \Pr[X_i = v|\xi]\,.$$

For fixed $\delta$ and $\epsilon$, the area-estimation algorithm requires a number of trials proportional to $A(E)/A(U)$. Hence, for straight simulation, the number of trials that guarantees convergence in variable $X_i$ satisfies

$$N \geq \frac{1}{\Pr[X_s = v|\xi]} \cdot \frac{1}{\delta\epsilon^2}\,, \tag{6}$$

where $\Pr[X_s = v|\xi]$ denotes the smallest probability in the network conditioned on evidence $\xi$.

By considering the calculation of probabilities in PIBNET, we derive an a priori bound on the number of trials. Let $M$ represent the Markov blanket of a node $X$; $M$ has $I$ instantiations $M_1, \ldots, M_I$. Summing over all instantiations, we obtain

$$\begin{aligned}\Pr[X_i = v|\xi] &= \sum_{k=1}^{I} \Pr[X_i = v|M_k,\xi] \cdot \Pr[M_k|\xi] \\ &\geq \min_{1\leq k\leq I} \{\Pr[X_i = v|M_k]\} \cdot \sum_{k=1}^{I} \Pr[M_k|\xi] \\ &= \min_{1\leq k\leq I} \{\Pr[X_i = v|M_k]\}.\end{aligned}$$

Letting $\tau = \min_{1\leq i\leq m,1\leq k\leq I}\{\Pr[X_i = v|M_k]\}$, we have as a sufficient number of trials

$$N \geq \frac{1}{\tau} \cdot \frac{1}{\delta\epsilon^2} \geq \frac{1}{\Pr[X_s = v|\xi]} \cdot \frac{1}{\delta\epsilon^2}.$$

As $\tau$ approaches 0, the running time to obtain a relative-error bound approaches infinity.

If we require only interval error bounds, we obtain

$$N \geq 1/(4\delta\alpha^2) \geq \frac{\Pr[X_i = v|\xi] \cdot (1 - \Pr[X_i = v|\xi])}{\delta\alpha^2}$$

for the number of trials, inasmuch as the expression $p(1 - p)$ has a maximum of 1/4 when $p = 1/2$.

We have predicated our analysis on the existence of a trial generator that produces states $s$ given evidence $\xi$ according to the p.m.f. $\Pr[s|\xi]$. Recall that the straight-simulation generator depends on the initial state. Moreover, we know nothing about the generator's convergence properties; finite runs of the generator might not accurately represent the true p.m.f. $\Pr[s|\xi]$. In the next section, we present a new trial generator in the form of an ergodic Markov chain that avoids trapped and periodic states. In addition, we analyze the chain's convergence properties rigorously. Finally, we define the complete BN-RAS and study its performance.

### 3.3 *The BN-RAS Trial Generator*

The method of straight simulation uses a state generator that enumerates nodes in a fixed order. The generator therefore behaves asymmetrically with respect to time. The analysis of convergence for Markov chains that lack time reversibility is extremely difficult [1]. In this section, we propose a time-reversible generator for PIBNET and we analyze its convergence.

Assume, for the moment, that each node $X_i$ has exactly two outcomes, $F$ and $T$. Assume, in addition, that there are no states with 0 probability (i.e., that there are no conditional probabilities equal to 0). The absence of 0 probabilities will aid in the construction of an ergodic chain. We shall later relax that assumption.

### 3.3.1. *Constuction of the Markov Chain*

For a belief network with $m$ nodes, define a Markov chain with $2^{m-1}(m+2)$ states, divided into two categories:

1. *The complete states.* Each complete state $CS$ corresponds to a set $cs$ with an instantiation for each of the $m$ nodes. The $2^m$ complete states are mutually exclusive.
2. *The partial states.* Each partial state $PS$ corresponds to a value assignment $ps$ for exactly $m-1$ of the $m$ nodes. There are $m$ ways to choose $m-1$ nodes, and $2^{m-1}$ states for each such choice. Hence, there are $m \cdot 2^{m-1}$ partial states, for a total of $N = 2^{m-1}(m+2)$ partial and complete states. The partial states are not necessarily mutually exclusive. Denote the state space by $[N]$, the set of integers from 1 to $N$.

Let the symbols $P_{ij}$ represent the transition probability from state $i$ to $j$. Define the transitions as follows:

- *From complete states CS.* With uniform probability $1/m$, randomly select node $X_i$ and move to the partial state in which $X_i$ has no value assignment. The complete state $CS$ corresponds to an instantiation $cs$ of all the variables in the original belief network.
- *From partial states PS.* Let $X_i$ represent the single node that lacks a value assignment in the current partial state. The partial state $PS$ corresponds to a configuration $ps$ of the belief network. Examine state $CS$ corresponding to $cs = ps \cup \{X_i \leftarrow T\}$ and state $\overline{CS}$ corresponding to $\overline{cs} = ps \cup \{X_i \leftarrow F\}$. Move to $CS$ with probability $\Pr[cs|\xi]/Pr[ps|\xi]$. Similarly, move to $\overline{CS}$ with probability $\Pr[\overline{cs}|\xi]/\Pr[ps|\xi]$. Notice, in addition, that $\Pr[ps|\xi] = \Pr[cs|\xi] + \Pr[\overline{cs}|\xi]$. We derive the transition probabilities by calculating a function $f$ (defined in [17]) on the Markov blanket $MB$ and computing the quotient

$$\frac{\Pr[cs|\xi]}{\Pr[ps|\xi]} = \frac{f[MB(cs)]}{f[MB(cs)] + f[MB(\overline{cs})]}.$$

Clearly, all states are reachable from any starting configuration. (The assumed absence of impossible states and functional dependencies guarantees reachability. Later, we shall describe a method that accomodates such dependencies.) For each state, add a self-loop probability of 0.5, and multiply all the transition probabilities defined previously by 0.5. The self-loop probability renders the

chain *aperiodic*, because it ensures that the Markov chain cannot become trapped in the endless repetition of a fixed-length cycle. All the states communicate, so the chain is also *irreducible*. Those two conditions taken together imply *ergodicity* for a finite-state Markov chain. Well-known results in the theory of Markov chains [10] guarantee that there exists a limit distribution $\pi$ over the states $[N]$ such that

$$\lim_{t\to\infty} P_{ij}^{(t)} = \pi_j \qquad \forall i,j \in [N]\,. \tag{7}$$

In other words, the $t$-step transition probabilities in an ergodic Markov chain approach a stationary distribution independent of the starting state. The stationary distribution $\pi$ is the unique left eigenvector of the transition matrix $P$ corresponding to eigenvalue 1; in other words, $\pi P = \pi$.

Adding the self-loop probability 0.5 does not alter the stationary distribution. Let $P$ represent the original transition matrix, with zeroes along the diagonal. If $\hat{P} = 1/2(I + P)$ represents the modified transition matrix, then

$$\pi\hat{P} = \frac{1}{2}(\pi I + \pi P) = \pi\,, \tag{8}$$

so $\pi$ is also the stationary distribution for $\hat{P}$.

Figure 4 demonstrates the ergodic, time-reversible Markov chain for a simple two-node belief network. The transition probabilities from each complete state

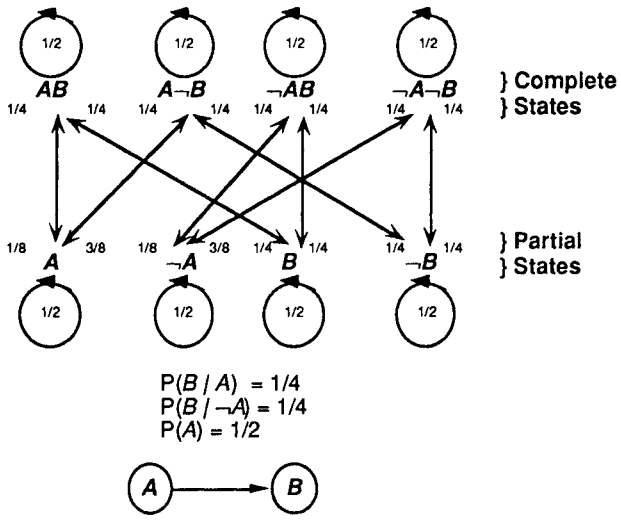



FIG. 4. This Markov chain enumerates the combinatorial space of structures corresponding to the two-node belief network $A \rightarrow B$. Each arrow represents a bidirectional edge. Each fraction denotes the probability of a transition out of the nearby node and into the target node.

to the two compatible partial states are 1/4. In addition, each state has a self-loop probability of 1/2. If we enumerate the states in the order $A$, $\bar{A}$, $B$, $\bar{B}$, $AB$, $A\bar{B}$, $\bar{A}B$, $\overline{AB}$, we obtain the transition matrix

$$\begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & \frac{1}{8} & \frac{3}{8} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{8} & \frac{3}{8} \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

We present the transition matrix here for purposes of explication only; the algorithm never requires calculation or storage of the complete matrix.

Standare matrix calculation yields

$$\tfrac{1}{16} \cdot [2 \quad 2 \quad 1 \quad 3 \quad 1 \quad 3 \quad 1 \quad 3]$$

as the left eigenvector $\pi$ of the transition matrix. Notice, in addition, that the stationary distribution represents each *cs* and *ps* with a weight proportional to the Markov probability. The chain spends half the time in partial states and half the time in complete states. Figure 5 gives the pseudocode for BN-RAS.

To analyze the Markov chain as a trial generator, we require that its stationary distribution correspond to the probabilities of the underlying states, multiplied by some constant. The ergodicity of the Markov chain guarantees a unique stationary distribution $\pi$ and, by extension, a unique solution of the equation $\pi P = \pi$. We guess an expression for the stationary distribution and verify that our formulation satisfies the eigenvector equation.

For arbitrary complete states $cs_i$, $\pi_i = 1/2\Pr[cs_i]$; and for arbitrary partial states $ps_j$, $\pi_j = (1/2m)\Pr[ps_j]$. To derive those equivalences, we observe that the eigenvector condition $\pi P = \pi$ requires

$$\pi_i = \sum_{ps_j} \pi_j P_{ji} + \frac{1}{2}\pi_i = 2\sum_{ps_j} \pi_j P_{ji}\,, \tag{9}$$

where the sum is taken over all partial states compatible with $cs_i$. Similarly, the eigenvector condition on $\pi_j$ requires

$$\pi_j = \sum_{cs_i} \pi_i P_{ij} + \frac{1}{2}\pi_j = 2\sum_{cs_i} \pi_i P_{ij}\,, \tag{10}$$

where the sum is taken over all complete states compatible with $ps_j$.

The transition probabilities are given by $P_{ij} = 1/2m$ from complete to partial

```
PROCEDURE do_transition;
{Perform one transition}
BEGIN
    p := rand;  {With probability 1/2, stay put (guarantees aperiodicity)}
    IF rand <= 0.5 THEN
      BEGIN
      {Move from a complete state to a partial state}
      uniformly choose a node X;
      {Compute the transition p.m.f., and move from a partial state to a complete state}
      sample(X);
      X.value := choose(X.dist);
      END;
END;

PROCEDURE next_trial(t: INTEGER);
{Compute a trial}
VAR
   i: INTEGER;
BEGIN
   set all the nodes to uniform  random values;
   FOR i := 1 TO t DO
      do_transition;
END;

PROCEDURE estimate;
VAR
   j, n, t: INTEGER;
BEGIN
   compute n, the number of trials, and t, the number of transitions per trial;
   FOR j := 1 TO n DO
      BEGIN
      next_trial(t);
      score outcomes;
      END;
END;
```

FIG. 5. Pseudocode for BN-RAS.

states, $P_{ji} = 1/2(\Pr[cs_i]/\Pr[ps_j])$ from partial to complete states, $P_{jj} = 1/2$, and $P_{ii} = 1/2$. We can therefore rewrite Eqs. (9) and (10) as

$$\pi_i = \sum_{ps_j} \pi_j \frac{\Pr[cs_i]}{\Pr[ps_j]} \tag{11}$$

and

$$\pi_j = \sum_{cs_i} \frac{\pi_i}{m}. \tag{12}$$

In addition, we have the normalization requirement

$$\sum_{cs_i} \pi_i + \sum_{ps_j} \pi_j = 1. \tag{13}$$

[Observe that Eqs. (11)–(13) also yield $2^m + m2^{m-1}$ equations in $2^m + m2^{m-1}$

unknowns, for which the ergodicity of the chain with transition matrix $P$ guarantees a unique solution.]

Substituting our expressions for $\pi_i$ and $\pi_j$ into Eqs. (11)–(13), we obtain

$$\pi_i = \sum_{ps_j} \frac{1}{2m} \cdot \Pr[ps_j] \frac{\Pr[cs_i]}{\Pr[ps_j]}$$

$$= \sum_{ps_j} \frac{1}{2m} \cdot \Pr[cs_i]$$

$$= \frac{1}{2} \Pr[cs_i]$$

and

$$\pi_j = \frac{1}{m} \cdot \sum_{cs_i} \frac{1}{2} \Pr[cs_i] = \frac{1}{2m} \Pr[ps_j] ,$$

because the sum over all complete states $cs_i$ compatible with $ps_j$ is equal to $\Pr[ps_j]$.

Finally, we verify that the normalization constraint holds. The probability of being in some Markov state is exactly 1. The sum of probabilities of partial states is $m$, inasmuch as each complete state corresponds to $m$ partial states; recall that the partial states are not mutually exclusive. Hence,

$$\sum_{cs_i} \pi_i + \sum_{ps_j} \pi_j = \sum_{cs_i} \frac{1}{2} \Pr[cs_i] +$$

$$\sum_{ps_j} \frac{1}{2m} \Pr[ps_j]$$

$$= 1 ,$$

as required.

The ergodicity of the chain implies that the Markov process has a unique stationary distribution. In other words, we know that $\pi$ is unique. Our interpretation of $\pi$ in terms of $\Pr[cs_i]$ and $\Pr[ps_j]$ satisfies all of the system's constraints on $\pi$ and is therefore unique.

### 3.4. Analysis of Convergence

The stationary distribution of the revised Markov chain reflects the underlying probabilities. By ignoring partial output states, we can construct a filtered Markov chain that produces complete assignments according to the correct probabilities.

To analyze the convergence of the chain with the techniques of Jerrum and Sinclair, we require time reversibility [11]. Examine the *detailed balance equation*, a necessary and sufficient condition for reversibility:

$$\pi_i P_{ij} = \pi_j P_{ji} \quad \forall i,j \in [N] . \tag{14}$$

Writing the equation in terms of probabilities yields

$$\frac{1}{2}\Pr[cs_i] \cdot \frac{1}{2m} = \frac{1}{2m}\Pr[ps_j] \cdot \frac{\Pr[cs_i]}{2\Pr[ps_j]},$$

which holds for all states.

For any time-reversible chain, introduce a weighted graph $H$ with vertices that correspond to states of the chain and edges $(i,j)$ with weight $\pi_i P_{ij}$. Furthermore, define the conductance

$$\Phi(H) = \min_S \left\{ \sum_{i \in S, j \notin S} \pi_i P_{ij} / \sum_{i \in S} \pi_i \right\}, \tag{15}$$

taking minimization over all subsets $S$ of states. Intuitively, the conductance measures the chain's tendency to flow around the state space. For the two-node example we have considered previously, an exhaustive calculation yields a conductance of approximately 0.035714.

By relating the conductance to the second eigenvalue of the chain's transition matrix, which governs the transient behavior of the chain, Jerrum and Sinclair have proved the following theorem [11].

**Theorem 1.** *Let $H$ (with edges $E(H)$ and vertices $V(H)$) represent the underlying graph of a time-reversible Markov chain with conductance $\Phi(H)$, stationary distribution $\pi$, and minimum self-loop probability* 1/2. *Let $P_{ij}^{(t)}$ denote the t-step transition probability from state i to j—that is, the probability of being in state j if one starts in state i and performs t transitions. Let $\Pi = \min_{i \in [N]} \pi_i$. Define the* relative pointwise distance (*r.p.d.*) *after t transitions,*

$$\Delta(t) = \max_{i,j \in [N]} \left\{ \frac{|P_{ij}^{(t)} - \pi_j|}{\pi_j} \right\}.$$

*Then, the r.p.d. is bounded by*

$$\Delta(t) \leq \frac{(1 - \Phi(H)^2/2)^t}{\Pi}.$$

Theorem 1 guarantees that rapid mixing of the Markov chain follows from a suitable lower bound on the graph-theoretic quantity $\Phi(H)$. For our two-node

example, the r.p.d. reaches a value of 2.0 after five transitions, at which time the bound $(1 - \Phi(H)^2/2)^t/\Pi$ is approximately equal to 16.0.

Let $p_0$ represent the smallest nonzero transition probability in the chain. Compute $p_0$ by examining each node's Markov blanket, multiplying the smallest elements in the corresponding link matrices, and normalizing. Then, multiply the result by 0.5, the adjustment factor for self-loop probabilities of 0.5. Following the exposition in [11], we bound the conductance by

$$\Phi(H) \geq p_0 \min_{0<|S|\leq N/2} \frac{(\Sigma_{i\in S}\pi_i) \cdot (\Sigma_{j\notin S,(i,j)\in E(H)}1)}{(\Sigma_{i\in S}\pi_i)} \geq p_0 \min_{0<|S|\leq N/2} (|C(S)|). \quad (16)$$

where for each subset $S$, $C(S)$ denotes the edges that cross from vertices in $S$ to vertices outside of $S$. Note, from the symmetric definition of conductance, that we need to consider only sets $S$ that contain half the vertices of $[N]$ or less. We bound the conductance by replacing the terms $P_{ij}$ with the minimum transition probability, $p_0$. The sums of $\pi_i$ terms cancel out in the numerator and the denominator.

First, define a *canonical directed simple path* (cdsp) between each pair of states in the underlying graph. To move from one complete state $cs_1$ to another state $cs_2$, fix an enumeration of the node variables $X_1, \ldots, X_m$. For $i$ from 1 to $m$, if $cs_1$ and $cs_2$ differ in their assignments to $X_i$, move to a partial state $ps_i$ and thence to an intermediate complete state with assignment to $X_i$ specified by $cs_2$. (The partial state $ps_i$ contains the assignments of $cs_2$ for variables 1 to $i - 1$ and the assignments of $cs_1$ for variables $i + 1$ to $m$.)

To move from any state $s_1$ to a target state $s_2$, define the following path:

- *Initial segment.* If $s_1$ is a partial state, move to the nearest compatible complete state $cs_1$ that specifies a value for the unassigned node. Otherwise, the initial segment is empty.
- *Main path.* Move from $cs_1$ to $cs_2$, the complete state nearest to $s_2$.
- *Final segment.* If $s_2$ is a partial state, move from $cs_2$ to $s_2$. Otherwise, the final segment is empty.

Recall that there is a cdsp between any pair of states, and so there are $|S|(N - |S|)$ cdsps connecting vertices in $S$ to vertices outside $S$. Because we have restricted $S$ to contain no more than half the vertices in the graph, the number of cdsps that cross the cut from $S$ to vertices not in $S$ is

$$|S|(N - |S|) \geq \frac{N|S|}{2}. \quad (17)$$

Suppose that no more than $b$ cdsps traverse any edge of $H$. Then, for any $S$, the number of edges in the set $C(S)$ must be at least $N|S|/2$ divided by $b$. Hence,

$$\Phi(H) \geq \frac{Np_0}{2b}.$$

Now find an upper bound on $b$. Examine a hypothetical edge that connects complete state $cs_k$ to partial state $ps_k(\alpha)$, where $ps_k(\alpha)$ is a partial state corresponding to $cs_k$ (but with the node $X_\alpha$ deassigned). Now consider an arbitrary pair of complete states $cs_0$ and $cs_f$. By the definition of a cdsp, the cdsp between $cs_0$ and $cs_f$ (called a $c$–$c$ cdsp, because it connects two complete states) will contain the edge that connects $cs_k$ and $ps_k(\alpha)$ if and only if $cs_0$ and $cs_f$ disagree in their assignment to node $X_\alpha$. Given any fixed complete state $cs_0$ in a network of binary-valued nodes, exactly $N/2$ of the other complete states differ from $cs_0$ in their assignment to variable $X_\alpha$. Therefore, no more than $(N/2)$ $c$–$c$ cdsps can cross the edge connecting $cs_k$ and $ps_k(\alpha)$, for arbitrary $k$ and $\alpha$.

We observe that, for a cdsp between partial states (known as a $p$–$p$ cdsp), the construction of initial and final segments guarantees a unique correspondence between the partial states at the termini of the cdsp and the complete states that demarcate the main segment of the cdsp. A similar argument holds for cdsps of type $c$–$p$ and $p$–$c$. For a given edge between $cs_k$ and $ps_k(\alpha)$, therefore, we have at most $(N/2)$ $c$–$c$ cdsps, $(N/2)$ $c$–$p$ cdsps, $(N/2)$ $p$–$c$ cdsps, and $(N/2)$ $p$–$p$ cdsps. We conclude that at most $2N$ cdsps cross a given edge. Thus,

$$\Phi(H) \geq p_0/4 \,. \tag{18}$$

For our two-node example, we see that $p_0/4 = 1/32$, and the bound indeed applies. Finally,

$$\Delta(t) \leq \frac{(1 - p_0^2/32)^t}{\Pi} \,.$$

To achieve a maximum relative error of $\gamma$ in approximating the stationary distribution, we perform the following number of transitions:

$$t \geq \frac{\log\gamma + \log\Pi}{\log(1 - p_0^2/32)} \,. \tag{19}$$

The preceding analysis assumes the absence of deterministically dependent nodes, in which case the Markov chain is aperiodic and irreducible. Modify the generator to accommodate such nodes by treating a group of deterministic nodes as one for the purposes of simulation. An influence on any of the nodes in a group becomes an influence on the group as a whole. With such a construction, no state has a 0 probability, and all states communicate. BNRAS calculates the posterior probabilities of individual deterministic nodes by examining the distribution of the synthetic group node. The method performs poorly in the presence of large deterministic clusters, in which case the synthetic group node acquires an excessively large number of possible outcomes.

### 3.5. Combining Generation with Estimation

We now recapitulate our earlier area-estimation analysis, but this time we account for the relative error $\gamma$ of the trial generator. Let $e$ represent the event

$X_i = T$ for some node $X_i$ in the network, let $Y$ be the estimator of $\Pr[X_i = T|\xi]$ (where $\xi$ denotes the background information), and let $\bar{Y}$ represent the average over $N$ trials. As usual, $U$ represents the region of unknown area $A(U) = \Pr[e|\xi]$. $E$ represents the sampling universe, with known area $A(E) = 1$.

We know that

$$A(U) = \Pr[e|\xi] \tag{20}$$

$$= \sum_{cs:(X_i \leftarrow T)} \Pr[cs|\xi] \tag{21}$$

$$= \sum_{k:(X_i \leftarrow T) \in cs_k} 2\pi_k \,. \tag{22}$$

Inasmuch as the Markov chain approximates the $\pi_k$ to within a relative error of no more than $\gamma$, the expectation of the summation $Y$ is bounded as

$$\frac{A(U)}{(1+\gamma)} \leq E[Y] \leq (1+\gamma)A(U) \,.$$

A similar argument shows that

$$\frac{1}{1+\gamma} A(E)A(U) \leq E[Y^2] \leq (1+\gamma)A(E)A(U) \,.$$

Combining terms, we observe that

$$\sigma^2 = E[Y^2] - E[Y]^2$$

$$\leq (1+\gamma)A(E)A(U) - [A(U)^2/(1+\gamma)^2]$$

and

$$\mu^2 \geq \frac{A(U)^2}{(1+\gamma)^2} \,.$$

Then

$$\frac{\sigma^2}{\mu^2} \leq \frac{(1+\gamma)A(E)A(U) - [A(U)^2/(1+\gamma)^2]}{A(U)^2/(1+\gamma)^2}$$

$$= (1+\gamma)^3(A(E)/A(U)) - 1 \,.$$

With

$$N \geq \frac{(1+\gamma)^3}{\tau\delta\epsilon^2} \tag{23}$$

trials, where $\tau = \min_{1 \le k \le I}\{\Pr[X = v|M_k]\}$, we can (with probability greater than $1 - \delta$) estimate each probability by $\tilde{Y}$ to within a compounded relative error of $\epsilon$ and $\gamma$.

Recall that each trial takes $t$ transitions and that each transition requires $O(\kappa \cdot B \cdot V)$ calculations, where $\kappa$ is the largest outdegree in the network, $B$ is the number of nodes in the largest Markov blanket, and $V$ is the number of values of the node with the most outcomes. To ensure that

$$\frac{\Pr[e|\xi]}{(1+\gamma)(1+\epsilon)} \le \tilde{Y} \le (1+\gamma)(1+\epsilon)\Pr[e|\xi]$$

with probability greater than $1 - \delta$, we must therefore perform

$$\left\lceil \frac{(1+\gamma)^3}{\tau\delta\epsilon^2} \right\rceil \cdot \frac{\log\gamma + \log\Pi}{\log(1 - p_0^2/32)} \tag{24}$$

transitions. In addition, we must score $N$ trials and we must perform $O(m \cdot V)$ arithmetic operations at the end. The tabulation of outcomes therefore requires $O(m \cdot (N + V))$ calculations, where $m$ is the number of nodes in the network.

A powering lemma for randomized approximation schemes offers a substantial improvement in running time [22]. In one simulation run, we perform

$$N \ge \frac{4(1+\gamma)^3}{3\tau\epsilon^2} \tag{25}$$

trials. Each trial requires $t$ transitions of the generator. Now compute $12\lceil -\log \delta \rceil + 1$ simulation runs, and for each node $X = v$ record a median of the probability estimates produced by the runs. According to Chernoff's bound, the median estimates will differ from the correct probabilities by a relative error less than $\epsilon$ with probability greater than $1 - \delta$. In other words, the powering lemma transforms an $O(\delta^{-1})$ approximation scheme into an $O(\log \delta^{-1})$ method.

The complete BN-RAS algorithm, which includes a powering scheme, a time-reversible generator, and area estimation, requires

$$\left\lceil \frac{16(1+\gamma)^3}{\tau\epsilon^2} \right\rceil \cdot (\lceil -\log \delta \rceil + 1) \cdot \frac{\log\gamma + \log\Pi}{\log(1 - p_0^2/32)} \tag{26}$$

transitions to achieve an approximation of the posterior probability to within a compounded relative error of $\gamma$ and $\epsilon$, with likelihood greater than $1 - \delta$.

The term $\log(1 - p_0^2/32)$ dominates the computation time; as $p_0$ approaches 0, the denominator goes to 0. Networks with conditional probabilities close to 0 or 1 can therefore require many transitions in order to guarantee the theoretical bounds on $\epsilon$ and $\delta$.

Finally, if we need to guarantee an interval error $\alpha$ rather than a relative

error $\delta$, the $\tau$ term disappears from the denominator. To ensure

$$\frac{\Pr[\epsilon|\xi]}{(1+\gamma)} - \alpha \leq \tilde{Y} \leq (1+\gamma)\Pr[e|\xi] + \alpha \tag{27}$$

with probability greater than $1 - \delta$, perform

$$\left\lceil \frac{4(1+\gamma)^3}{\alpha^2} \right\rceil \cdot (\lceil -\log \delta \rceil + 1) \cdot \frac{\log \gamma + \log \Pi}{\log(1 - p_0^2/32)} \tag{28}$$

transitions. Notice that we also require $N$ tallies of the generated trials, where each tally takes $O(m \cdot (N + V))$ time.

## 4. CONCLUSIONS

We have shown how to compute a priori bounds on the randomized computation of marginal posterior probabilities in belief networks. The BN-RAS algorithm offers, with high probability of success, an upper bound on both relative and interval error. In addition, it guarantees that the simulated Markov chain converges to a reasonable approximation of its stationary distribution within a prespecified number of transitions.

The analysis of running time suggests that BN-RAS performs well on networks of arbitrary size and topological complexity, at the expense of a strict dependence on the probabilities within the network. In networks with conditional probabilities that approach 1 and 0, our theoretical analysis suggests that we explore alternatives to stochastic simulation. For large enough conditional probabilities, however, we expect BN-RAS to require time that is linear in the size of the network.

## ACKNOWLEDGMENTS

Bruce Buchanan, Lyn Dupré, David Heckerman, Eric Horvitz, Harry Lewis, Ross Shachter, Edward Shortliffe, and Les Valiant provided important feedback. This work has been supported by grant IRI-8703710 from the National Science Foundation, grant P-25514-EL from the U.S. Army Research Office, Medical Scientist Training Program grant GM07365 from the National Institutes of Health, and grant LM-07033 from the National Library of Medicine. Computer facilities were provided by the SUMEX-AIM resource under grant RR-00785 from the National Institutes of Health.

## References

[1] A. Z. Broder, How hard is it to marry at random? (On the approximation of the permanent). *Proceedings of the Eighteenth ACM Symposium on Theory of Computing*, (1986) 50–58.

[2] A. Bundy, Incidence calculus: A mechanism for probabilistic reasoning. *Proceedings of the First Workshop on Uncertainty in Artificial Intelligence*, UCLA, Los Angeles, CA, 1985, American Association for Artificial Intelligence, 177–184.

[3] R. M. Chavez, *Randomized algorithms for probabilistic expert systems*. PhD thesis, Knowledge Systems Laboratory, Stanford University, Stanford, CA (to appear).
[4] H. L. Chin and G. F. Cooper, Stochastic simulation of Bayesian belief networks. *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence*, Seattle, WA, 1987, American Association for Artificial Intelligence, 106–113.
[5] H. L. Chin and G. F. Cooper, Bayesian belief network inference using simulation. *Uncertainty in Artificial Intelligence* 3. North-Holland, Amsterdam (1989).
[6] G. F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intell.* in press.
[7] G. F. Cooper, Probabilistic inference using belief networks is NP-hard. Technical Report KSL-87-27, Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University, Stanford, CA, May (1987).
[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, (1979).
[9] M. Henrion, Propagating uncertainty in Bayesian networks by probabilistic logic sampling. *Uncertainty in Artificial Intelligence* 2, North-Holland, Amsterdam (1988) 149–163.
[10] R. A. Howard, *Dynamic Programming and Markov Processes*, Technology Press of the Massachusetts Institute of Technology, Cambridge, MA (1960).
[11] M. Jerrum and A. Sinclair, Conductance and the rapid mixing property for Markov chains: The approximation of the permanent resolved. *Proceedings of the Twentieth ACM Symposium on Theory of Computing*, Chicago, IL, (1988) 235–244.
[12] R. M. Karp and M. Luby, Monte-Carlo algorithms for enumeration and reliability problems. *Proceedings of the Twenty-fourth IEEE Symposium on Foundations of Computer Science* (1983).
[13] S. L. Lauritzen and D. J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Stat. Soc. B* **50**(2) (1988) 157–226.
[14] C. H. Papadimitriou, Computational complexity. *Combinatorial Optimization: Annotated Bibliographies*, Prentice-Hall, Englewood Cliffs, NJ (1985) 39–51.
[15] C. H. Papadimitriou, Games against nature. *J. Comput. Syst. Sci.* **31**(2) (1985) 288–301.
[16] J. Pearl, Fusion, propagation, and structuring in belief networks. *Artificial Intell.* **29** (1986) 241–288.
[17] J. Pearl, Evidential reasoning using stochastic simulation of causal models. *Artificial Intell.* **32** (1987) 245–257.
[18] J. Pearl, Evidential reasoning using stochastic simulation of causal models (addendum). *Artificial Intell.* **33** (1987) 131.
[19] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA (1988).
[20] R. D. Shachter, Probabilistic inference and influence diagrams. *Operations Res.* **36** (1988) 589–604.
[21] R. D. Shachter and M. A. Peot, Simulation approaches to general probabilistic inference on belief networks. *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, Windsor, Ontario, Canada, August 1989. American Association for Artificial Intelligence.
[22] A. J. Sinclair and M. R. Jerrum, Approximate counting, uniform generation and rapidly mixing Markov chains. Technical Report CSR-241-87, Department of Computer Science, University of Edinburgh, Edinburgh, Scotland, UK (1987).
[23] L. G. Valiant, The complexity of computing the permanent. *Theor. Comput. Sci.* **8** (1979) 189–201.

Received February 1990
Accepted March 1990