An Optimal Approximation Algorithm for Bayesian Inference

Paul Dagum* **Michael Luby†**

Abstract

Approximating the inference probability $\mathbf{Pr}[X = x|E = e]$ in *any* sense, even for a single evidence node E, is **NP**-hard. This result holds for belief networks that are allowed to contain *extreme* conditional probabilities—that is, conditional probabilities arbitrarily close to 0. Nevertheless, all previous approximation algorithms have failed to approximate efficiently many inferences, even for belief networks without extreme conditional probabilities.

We prove that we can approximate efficiently probabilistic inference in belief networks without extreme conditional probabilities. We construct a *randomized* approximation algorithm—the *bounded-variance algorithm*—that is a variant of the known likelihood-weighting algorithm. The bounded-variance algorithm is the first algorithm with provably fast inference approximation on all belief networks without extreme conditional probabilities.

From the bounded-variance algorithm, we construct a *deterministic* approximation algorithm using current advances in the theory of pseudorandom generators. In contrast to the exponential worst-case behavior of all previous deterministic approximations, the deterministic bounded-variance algorithm approximates inference probabilities in worst-case time that is subexponential $2^{(\log n)^d}$, for some integer d that is a linear function of the depth of the belief network.

Keywords: Bayesian inference, approximation, belief networks

*Section on Medical Informatics, Stanford University School of Medicine, Stanford, California 94305-5479.

†International Computer Science Institute, Berkeley, CA 94704

1 Approximation Algorithms

Belief networks are powerful graphical representations of probabilistic dependencies among domain variables. Belief networks have been used successfully in many real-world problems in diagnosis, prediction, and forecasting (for example, papers included in [1, 2]). Various exact algorithms exist for probabilistic inference in belief networks [19, 21, 27]. For a few special classes of belief networks, these algorithms can be shown to compute conditional probabilities efficiently.

Cooper [6], however, showed that exact probabilistic inference for general belief networks is **NP**-hard. Cooper's result prompted constructions of approximation algorithms for probabilistic inference that trade off complexity in running time for the accuracy of computation. These algorithms comprise simulation-based and search-based approximations. Simulation-based algorithms use a source of random bits to generate random samples of the solution space. Simulation-based algorithms include straight simulation [25, 26], forward simulation, [15], likelihood weighting [13, 33], and randomized-approximation schemes [3, 4, 7, 8]. Variants of these methods such as backward simulation [14], exist; Neal [23] provides a good overview of the theory of simulation-based algorithms. Search-based algorithms search the space of alternative instantiations to find the most probable instantiation. These methods yield upper and lower bounds on the inference probabilities. Search-based algorithms for probabilistic inference include NESTOR [5], and, more recently, algorithms restricted to two-level (bipartite) noisy-OR belief networks [16, 28, 29], and other more general algorithms [11, 17, 18, 30, 32, 34].

Approximation algorithms are categorized by the nature of the bounds on the estimates that they produce and by the reliability with which the exact answer lies within these bounds. The following inference-problem instance characterizes the two forms of approximation [10]:

INSTANCE: A real value ϵ between 0 and 1, a belief network with binary valued nodes V, arcs A, conditional probabilities $\mathbf{Pr}$, hypothesis node X and set of evidence nodes

$\mathcal{E}$ in V instantiated to x and e, respectively

Absolute and *relative* approximations refer to the type of approximation error and are defined as follows:

ABSOLUTE APPROXIMATION: An estimate $0 \leq Z \leq 1$, such that $\mathbf{Pr}[X = x|\mathcal{E} = e] - \epsilon \leq Z \leq \mathbf{Pr}[X = x|\mathcal{E} = e] + \epsilon$

RELATIVE APPROXIMATION: An estimate $0 \leq Z \leq 1$, such that $\mathbf{Pr}[X = x|\mathcal{E} = e](1 - \epsilon) \leq Z \leq \mathbf{Pr}[X = x|\mathcal{E} = e](1 + \epsilon)$

Deterministic and *randomized approximations* refer to the probability that the approximation Z is within the specified bounds. An approximation algorithm is *deterministic* if it *always* produces an approximation Z within the specified bounds. In contrast, an approximation algorithm is *randomized* if the approximation Z fails to be within the specified bounds with some probability $\delta > 0$.

Let n parametrize the size of the input to the approximation algorithm—that is, the size of the input is bounded by a polynomial function of n. For example, in algorithms designed to approximate inference in belief networks, n may be either the number of belief-network nodes or the size of the largest conditional-probability table. For a deterministic algorithm, the running time of an approximation procedure for $\mathbf{Pr}[X = x|\mathcal{E} = e]$ is said to be *polynomial* if it is polynomial in n and ϵ^{-1}. For a randomized algorithm, the running time is defined as polynomial if it is polynomial in n, ϵ^{-1}, and $\ln \delta^{-1}$.

Simulation-based algorithms are examples of randomized-approximation algorithms; search-based algorithms are examples of deterministic-approximation algorithms that output absolute approximations. Both types of algorithms are known to require exponential time to estimate hard inferences. For example, forward-simulation and

likelihood-weighting algorithms require exponential time to converge to small inference probabilities. Since these algorithms estimate the inference probability $\mathbf{Pr}[X = x|\mathcal{E} = e]$ from the ratio of the probabilities $\mathbf{Pr}[X = x, \mathcal{E} = e]$ and $\mathbf{Pr}[\mathcal{E} = e]$, they require exponential time for rare hypotheses or rare evidence. Most search-based algorithms are heuristic algorithms that also require exponential time to approximate many inference probabilities. For example, we know that, even when $\mathcal{E}$ is a single node E, if we allow some of the other nodes to have *extreme* conditional probabilities with values near 0, then any polynomial-time algorithm cannot generate (1) deterministic approximations of the inference probability $\mathbf{Pr}[X = x|E = e]$ with *absolute* error $\epsilon < 1/2$, unless $\mathbf{NP} \subseteq \mathbf{P}$; and (2) randomized approximations with *absolute* error $\epsilon < 1/2$ and failure probability $\delta < 1/2$, unless $\mathbf{NP} \subseteq \mathbf{RP}$ [10].

The complexity of exact or approximate computation of inference probabilities $\mathbf{Pr}[X = x|E = e]$ in belief networks without extreme conditional probabilities remained enigmatic. Known results did not categorize these problems as **NP**-hard, yet all previous approximate inference algorithms failed to output reliably solutions in polynomial time, and exact algorithms had exponential worst-case run times. We construct the *bounded-variance algorithm* that proves that the complexity of approximating inferences in belief networks without extreme conditional probabilities is polynomial-time solvable.

The bounded-variance algorithm is a simple variant of the known likelihood-weighting algorithm [13, 33], which employs recent results on the design of optimal algorithms for Monte Carlo simulation [9]. We consider an n-node belief network without extreme conditional probabilities and an evidence set $\mathcal{E}$ of constant size. We prove that, with a small failure probability δ, the bounded-variance algorithm approximates any inference $\mathbf{Pr}[X = x|\mathcal{E} = e]$ within *relative* error ϵ in time polynomial in n, ϵ^{-1}, and $\ln \delta^{-1}$. Thus, we prove that, for belief networks without extreme conditional probabilities, probabilistic-inference approximation is polynomial-time solvable; otherwise, it is **NP**-hard.

The bounded-variance algorithm is a randomized algorithm with an associated

failure probability δ. We use current advances in the theory of pseudorandom generation to derandomize this algorithm. The resulting algorithm is a deterministic-approximation algorithm. All previously known deterministic algorithms—for example, search-based methods—output relative approximations that require exponential running time in the worst case. We prove, however, that the deterministic bounded-variance algorithm outputs a *relative* approximation of $\mathbf{Pr}[X = x|\mathcal{E} = e]$ in worst-case subexponential time $2^{(\log n)^d}$ for some integer $d > 1$. The integer d depends on the depth of the belief network—that is, on the longest directed path between a root node and a leaf node. Thus, for small d, the deterministic bounded-variance algorithms offers a substantial speedup over the known exponential worst-case behavior of all previous deterministic algorithms.

We prove that, if a belief network contains extreme conditional probabilities, we can still efficiently approximate certain inferences: Provided the conditional probabilities for nodes X and $\mathcal{E}$ in an inference probability $\mathbf{Pr}[X = x|\mathcal{E} = e]$ are not extreme, we prove that the bounded-variance algorithm and the deterministic algorithm approximate $\mathbf{Pr}[X = x|\mathcal{E} = e]$ efficiently. Thus, we can apply our results even to belief networks with extreme conditional probabilities, provided that the conditional probabilities of nodes X and $\mathcal{E}$ that appear in the inference probability are not extreme.

2 Deterministic Versus Randomized Algorithms

To introduce the difference between deterministic and randomized algorithms, we can contrast the complexity of deterministic algorithms with the complexity of randomized algorithms for the simple case when the algorithms output *absolute* approximations. Randomized algorithms use random bits to generate samples of the solution space. Computer scientists have shown that randomization renders many problems tractable to polynomial-time approximations. These problems constitute the complexity class **RP**. Whether we can also generate deterministic approximations in polynomial time for problems in **RP** is a major open problem. Yao [35] shows that,

if pseudorandom generators exist, then we can generate deterministic approximations for any problem in **RP** in subexponential time $2^{(\log n)^d}$ for some integer $d > 1$. Constructions of deterministic-approximation algorithms for specific problems in **RP** that do not rely on unproved conjectures, such as the existence of pseudorandom generators, have also achieved subexponential time [12, 22]. Thus far, deterministic-approximation algorithms require substantially increased run time, in comparison to a randomized-approximation algorithm for the same problem. Deterministic algorithms, however, have two significant advantages: (1) they do not require random bits, and (2) they do not fail to produce an approximation. Good random bits are computationally expensive, and a poor source of random bits biases the output. Furthermore, although we can make the failure probability of a randomized algorithm small by increasing the run time, we never know when the algorithm fails to output a valid approximation Z.

To approximate the inference probability $\mathbf{Pr}[\mathcal{W} = w]$, randomized algorithms attempt to find a small number of instantiations of the set of all nodes $\mathcal{X} = \mathcal{Z} \cup \mathcal{W}$ that is representative of the probability space $(\Omega, 2^{\Omega}, \mathbf{Pr})$, where Ω denotes the set of all instantiations of $\mathcal{X}$. Let $\Phi \subset \Omega$ denote a subset of instantiations, and, for any instantiation w of $\mathcal{W}$, let $\mathbf{Pr}_{\Phi}[\mathcal{W} = w]$ denote the fraction of the instantiations in Φ that instantiate nodes $\mathcal{W}$ to w. The subset Φ *preserves* the properties of the probability space if, for *any* subset of nodes $\mathcal{W}$ and *any* instantiation of these nodes, the inference probability $\mathbf{Pr}_{\Phi}[\mathcal{W} = w]$ differs from the probability $\mathbf{Pr}[\mathcal{W} = w]$ by an absolute error ϵ. We refer to such a set Φ as a *preserving set*. Monte Carlo theory proves that there exists a preserving set Φ of size $O(1/\epsilon^2)$. This result follows directly from Chebyshev's inequality. Unfortunately, the theory does not provide a method to construct deterministically the set Φ. Nonetheless, we can prove with some nonzero probability, the $O(1/\epsilon^2)$ instantiations generated by a Monte Carlo algorithm provides the set Φ. Thus, for example, if we use a simple randomized algorithm, such as forward simulation, to generate complete instantiations of the belief network, then, with some nonzero probability, the set of instantiations generated

after $O(1/\epsilon^2)$ simulations provides a preserving set. The efficiency of such a randomized approach improves substantially the complexity of deterministic search-based algorithms. Both algorithms output absolute approximations; however, the randomized approach requires in all cases polynomial time, whereas search-based algorithms require exponential worst-case time.

The tradeoff, of course, is that the output of $O(1/\epsilon^2)$ simulations of the randomized algorithm may fail to provide a preserving set Φ, and, therefore, the estimates computed from the output of these simulations are not valid approximations. We do not know how to verify efficiently when the outcome of $O(1/\epsilon^2)$ simulations provides a preserving set.

Because Monte Carlo theory proves that small sets Φ preserving the properties of probability spaces do indeed exist researchers attempted for several decades to derandomize Monte Carlo algorithms through deterministic constructions of these sets. Some of the early work used Latin hypercube sampling for one-dimensional problems, and uniform grids for multidimensional problems. Both these methods led to exponentially large sets Φ. Recent advances in theoretical computer science on pseudorandom generators have shed light on the deterministic construction of small sets Φ. At the heart of these methods lies the ability to stretch a short string of m truly random bits into a long string of $n > m$ pseudorandom bits. If the pseudorandom bits appear random to a specific model of computation, then we can use them as inputs to a randomized algorithm in this model of computation. By stretching all 2^m possible m-bit strings into length n pseudorandom bit strings, we generate deterministically a set of 2^m sample points that we use for Φ. Although, with current methods for stretching short random bit strings into longer pseudorandom bit strings, we can construct sets Φ that are subexponential, further development in this field may ultimately elucidate methods for deterministically constructing sets Φ that approach the $O(1/\epsilon^2)$ bound, suggesting that $\mathbf{RP}$=$\mathbf{P}$.

3 Randomized Approximation

In this section, we present the bounded-variance algorithm. First we formally characterize the class of belief networks without extreme conditional probabilities by the *local variance bound (LVB)* of a belief network. This bound captures both the representational expressiveness and the complexity of inference of belief networks. We prove that the LVB demarcates the boundary between the class of belief networks with intractable approximations and that of those with polynomial approximations. We construct polynomial-approximation algorithms for the latter class.

We define the LVB as follows. For any belief network node X and parents $\pi(X) = \{Y_1, ..., Y_t\}$, let $u(X = x_i)$ denote the maximum, and let $l(X = x_i)$ denote the minimum, of $\mathbf{Pr}[X = x_i | \pi(X) = y]$ over all instantiations $y = \{y_1, ..., y_t\}$ of $\pi(X)$. The LVB is the maximum of the ratio $\frac{u(X=x_i)}{l(X=x_i)}$ over all nodes X and all instantiations x_i of X. For binary-valued belief networks, for example, the LVB reduces to the ratio $\max(\frac{\mathrm{u}}{\mathrm{l}}, \frac{1-\mathrm{l}}{1-\mathrm{u}})$, such that, for every node X, either the interval [l,u] or [1-u,1-l] contains the conditional probability $\mathbf{Pr}[X = 0 | \pi(X) = x]$ for all instantiations x of $\pi(X)$. (Note that, if the interval [l,u] contains $\mathbf{Pr}[X = 0 | \pi(X)]$ for all instantiations of $\pi(X)$, then [1-u, 1-l] contains $\mathbf{Pr}[X = 1 | \pi(X)]$ for all instantiations of $\pi(X)$.)

We make the following assumptions throughout the rest of the paper: (1) all nodes are binary valued, (2) the number of nodes n parametrizes the size of the belief network, and (3) the LVB of the belief network is bounded by some polynomial n^r for some integer r. Assumption 1 simplifies the presentation; however, both the bounded-variance and the derandomized algorithms apply to belief networks with arbitrary m-ary valued nodes with similar running-time results. Assumption 2 also simplifies the presentation. This assumption is valid provided that each conditional-probability table has at most $f(n)$ entries, where f is some polynomial function. For classes of belief networks where f is not a polynomial, we must use $f(n)$ to parametrize the belief-network size. In the latter case, we can also prove convergence times to relative approximations that are polynomial and subexponential in the input size, $f(n)$, for the bounded-variance and the derandomized algorithms, respectively. Those results

apply to belief networks with LVB bounded by a polynomial in $f(n)$. Those cases may be less interesting, however, because both the space requirement of the belief-network encoding, and the computational time for an approximation may be an exponential function of the number of nodes n if $f(n)$ is an exponential function. For large n, both storage and computation become intractable.

3.1 Likelihood-Weighting Algorithm

We want to approximate the inference probability $\mathbf{Pr}[X = x|\mathcal{E} = e]$. If we generate *relative* approximations of the inference probabilities $\mathbf{Pr}[X = x, \mathcal{E} = e]$ and $\mathbf{Pr}[\mathcal{E} = e]$ with relative errors ϵ, then the ratio

$$\frac{\mathbf{Pr}[X = x, \mathcal{E} = e]}{\mathbf{Pr}[\mathcal{E} = e]}$$

also represents a relative approximation of $\mathbf{Pr}[X = x|\mathcal{E} = e]$, with relative error 2ϵ. We cannot, however, construct *absolute* approximations of $\mathbf{Pr}[X = x|\mathcal{E} = e]$ from absolute approximations of $\mathbf{Pr}[X = x, \mathcal{E} = e]$ and $\mathbf{Pr}[\mathcal{E} = e]$. Although Chebyshev's inequality proves that algorithms such as forward simulation generate absolute approximations $\mathbf{Pr}[X = x, \mathcal{E} = e]$ and $\mathbf{Pr}[\mathcal{E} = e]$ in polynomial time, we cannot use these approximations to estimate $\mathbf{Pr}[X = x|\mathcal{E} = e]$ with any type of error.

To generate approximations of inference probabilities $\mathbf{Pr}[X = x|\mathcal{E} = e]$, likelihood-weighting algorithms proceed as follows. Let $\mathcal{Z}$ denote the belief-network nodes not contained by the subset $\mathcal{E}$. Decompose the full joint probability $\mathbf{Pr}[\mathcal{Z} = z, \mathcal{E} = e]$ of the belief network into the *path probability* $\rho(z, e)$ and the *weight distribution* $\omega(z, e)$:

$$\rho(z, e) = \prod_{Z_i \in \mathcal{Z}} \mathbf{Pr}[Z_i|\pi(Z_i)]|_{\mathcal{Z}=z, \mathcal{E}=e},$$

and

$$\omega(z, e) = \prod_{E_i \in \mathcal{E}} \mathbf{Pr}[E_i|\pi(E_i)]|_{\mathcal{Z}=z, \mathcal{E}=e}.$$

(The notation $|_{\mathcal{Z}=z, \mathcal{E}=e}$ appended to the functions $\prod_{Z_i \in \mathcal{Z}} \mathbf{Pr}[Z_i|\pi(Z_i)]$ and $\prod_{E_i \in \mathcal{E}} \mathbf{Pr}[E_i|\pi(E_i)]$ denotes instantiation of their arguments $\mathcal{Z}$ and $\mathcal{E}$ to z and e, respectively.)

The path probability represents a probability distribution over the space Ω of all $2^{|\mathcal{Z}|}$ instantiations of the set of nodes $\mathcal{Z}$. The weight distribution represents a random variable of this probability space with mean $\mathbf{E}\omega = \sum_{z\in\Omega} \rho(z,e)\omega(z,e) = \mathbf{Pr}[\mathcal{E} = e]$. Thus, if we sample the probability space, the mean ϕ_1 of the values $\omega(z_1,e),...,\omega(z_N,e)$ generated from N samples $z_1,...,z_N$ converges to $\mathbf{Pr}[\mathcal{E} = e]$ in the limit of infinite samples N.

We next define a new random variable $\chi(z,e)$ that is equal to 1 if $\mathcal{Z} = z$ instantiates the node X to x, and is equal to 0 otherwise:

$$\chi(z,e) = \begin{cases} 1, & \text{if } X = x \text{ in } z; \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the mean of the random variable $\chi(z,e)\omega(z,e)$ is

$$\mathbf{E}[\chi \cdot \omega] = \sum_{z\in\Omega} \chi(z,e)\rho(z,e)\omega(z,e) = \mathbf{Pr}[X = x, \mathcal{E} = e].$$

If we sample the distribution $\rho(z,e)$, then, in the limit of infinite samples N, the mean ϕ_2 of the values $\chi(z_1,e)\omega(z_1,e),...,\chi(z_N,e)\omega(z_N,e)$ converges to $\mathbf{Pr}[X = x, \mathcal{E} = e]$. To complete the description of the likelihood-weighting algorithm, we must show how to generate samples with distribution $\rho(z,e)$.

To generate a sample z with probability $\rho(z,e)$, we begin with the root nodes of the belief network. If a root node E_i belongs to the set $\mathcal{E}$, then we instantiate it to e_i. Otherwise, for each root node Z_i we choose a number u uniformly from the interval $[0,1]$; we then set $Z_i = 0$ if $u \leq \mathbf{Pr}[Z_i = 0]$, $Z_i = 1$ otherwise. Because u is chosen uniformly from the interval $[0,1]$, the probability that u is less than $\mathbf{Pr}[Z_i = 0]$ is precisely $\mathbf{Pr}[Z_i = 0]$. Thus, with probability $\mathbf{Pr}[Z_i = 0]$, the algorithm instantiates Z_i to 0; with probability $\mathbf{Pr}[Z_i = 1]$ it instantiates Z_i to 1. Once all the root nodes are instantiated, we proceed to the set of those nodes that have all parents instantiated. Again, if any of these nodes belong to the set $\mathcal{E}$, we instantiate them according to e; otherwise, we set them according to the outcome of a random sample from $[0,1]$. We proceed until we instantiate all nodes in $\mathcal{Z}$. This method generates an instantiation $\mathcal{Z} = z$ with the desired probability $\rho(z,e)$.

We discussed how the means ϕ_1 and ϕ_2 of the random variables $\omega(z, e)$ and $\chi(z, e)\omega(z, e)$, respectively, converge to the inference probabilities $\mathbf{Pr}[\mathcal{E} = e]$ and $\mathbf{Pr}[X = x, \mathcal{E} = e]$ in the limit of infinite samples. To generate relative approximations, however, we require only that, with probability at least $1 - \delta$, the estimates ϕ_1 and ϕ_2 approximate $\mathbf{Pr}[\mathcal{E} = e]$ and $\mathbf{Pr}[X = x, \mathcal{E}]$ with relative error ϵ. Since we generate samples $\mathcal{Z} = z$ in polynomial time, the run time of the likelihood-weighting algorithm depends on the number of samples required to guarantee convergence. Thus, for likelihood-weighting algorithms, we are interested in an upper bound on N that guarantees that, for any $\epsilon, \delta > 0$,

$$\mathbf{Pr}[\mu(1 - \epsilon) \leq \phi \leq \mu(1 + \epsilon)] > 1 - \delta, \tag{1}$$

with μ equal to $\mathbf{Pr}[\mathcal{E} = e]$ or $\mu = \mathbf{Pr}[X = x, \mathcal{E} = e]$. The Zero—One Estimator Theorem [20] gives an upper bound on the number N:

$$N = \frac{4}{\mu\epsilon^2} \ln \frac{2}{\delta}.$$

Thus, provided that the probability $\mathbf{Pr}[X = x, \mathcal{E} = e] \leq \mathbf{Pr}[\mathcal{E} = e]$ is not *too* small—for example, it is at least $1/n^{O(1)}$—the number of samples N is polynomial, and the algorithm converges in polynomial time. Unfortunately, if $\mathcal{E}$ consists of several nodes or of a node instantiated to a rare value, then $\mathbf{Pr}[X = x, \mathcal{E} = e]$ does not satisfy this constraint. Furthermore, even when $\mathbf{Pr}[X = x, \mathcal{E} = e]$ does satisfy the constraint, we cannot verify a priori that it does.

3.2 Likelihood-Weighting Algorithm, Revisited

We are interested in an efficient algorithm to approximate inference probabilities. An efficient version of the likelihood-weighting algorithm suggests itself. We refer to this version of likelihood weighting as the *bounded-variance algorithm*, to distinguish it from the algorithm that we described in Section 3.1. Unlike the likelihood-weighting algorithm, the bounded-variance algorithm approximates inference probabilities in polynomial time.

Recall that, to approximate the inference probability $\mathbf{Pr}[X = x|\mathcal{E} = e]$, the likelihood-weighting algorithm outputs relative approximations of the inferences $\mathbf{Pr}[X = x, \mathcal{E} = e]$ and $\mathbf{Pr}[\mathcal{E} = e]$. At a glance, we may find it unusual that likelihood weighting approximates these probabilities by different methods. Clearly, we can also approximate the inference probability $\mathbf{Pr}[X = x, \mathcal{E} = e]$ by simply averaging the likelihood weights for the joint evidence $X = x$ and $\mathcal{E} = e$. This version of the likelihood-weighting algorithm constitutes the basis of the bounded-variance algorithm. By not using the random variable $\chi(z, e)$, we reduce substantially the variance of our estimates. In fact, when the inference probability $\mathbf{Pr}[X = x|\mathcal{E} = e]$ is small, we know, based on a straightforward application of the Generalized Zero—One Estimator Theorem [9], that likelihood weighting requires exponential time to converge to an approximation. In contrast, we prove that the bounded-variance algorithm converges in polynomial time.

We now describe formally the bounded-variance algorithm. Let $\mathcal{W}$ denote a subset of belief-network nodes, and let w denote some instantiation of these nodes. Provided that we can generate relative approximations of inference probabilities $\mathbf{Pr}[\mathcal{W} = w]$ for any subset of nodes $\mathcal{W}$, then we can also generate relative approximations of $\mathbf{Pr}[X = x|\mathcal{E} = e]$ for any set of evidence nodes. Thus, we want to approximate the inference $\mathbf{Pr}[\mathcal{W} = w]$. Suppose that nodes $W_1, ..., W_k$ constitute the set $\mathcal{W}$. The version of the likelihood-weighting algorithm that we described in Section 3.1 would score the random variable $\omega(z, w)$ that is the product of the conditional probabilities $\mathbf{Pr}[W_i = w_i|\pi(W_i)]|_{\mathcal{W}=w,\mathcal{Z}=z}$. To prove rapid convergence, however, we modify the algorithm slightly. Let the intervals $[\mathrm{l}_i, \mathrm{u}_i]$ contain each conditional probability $\mathbf{Pr}[W_i = w_i|\pi(W_i)]|_{\mathcal{W}=w,\mathcal{Z}=z}$ for all instantiations $\mathcal{Z} = z$. We form a new random variable

$$\zeta(z, w) = \frac{\omega(z, w)}{\prod_{i=1}^{k} \mathrm{u}_i},$$

contained in the interval $[\prod_{i=1}^{k} \mathrm{l}_i/\mathrm{u}_i, 1]$. We generate instantiations $z_1, z_2, ...$ of the nodes in $\mathcal{Z}$ according to the prescription that we described for likelihood weighting.

Let S_t denote the sum of the first t sample outcomes,

$$S_t = \zeta(z_1, w) + \cdots + \zeta(z_t, w).$$

We run the algorithm until

$$S_T \geq \frac{4(1+\epsilon)}{\epsilon^2} \ln \frac{2}{\delta},$$

for some number of samples T. We output

$$\phi = \frac{S_T}{T} \prod_{i=1}^{k} \mathrm{u}_i.$$

3.2.1 Proof of Polynomial Runtime

We now prove that the bounded-variance algorithm halts after a polynomial number of samples T, and outputs an estimate ϕ that approximates $\mathbf{Pr}[\mathcal{W} = w]$ with relative error ϵ.

Theorem 1 *Let w instantiate nodes $\mathcal{W} = \{W_1, ..., W_k\}$ and let $\Gamma = \frac{4(1+\epsilon)}{\epsilon^2}$. The bounded-variance algorithm halts after an expected number of samples $\mathbf{E}T$,*

$$\mathbf{E}T \leq \frac{\prod_{i=1}^{k} u_i}{\mu} \Gamma \ln \frac{2}{\delta},$$

and outputs an estimate ϕ that with probability at least $1 - \delta$ approximates $\mu = \mathbf{Pr}[\mathcal{W} = w]$ with relative error ϵ.

PROOF Let $\mathbf{E}\zeta$ denote the mean of the random variable $\zeta(z, w)$ in the bounded-variance algorithm. By the Stopping Theorem (Section A.1 in the appendix), the bounded-variance algorithm halts with expected number of samples $\mathbf{E}T \leq (1/\mathbf{E}\zeta)\Gamma \ln(2/\delta)$. But, by definition of the bounded-variance algorithm, $1/\mathbf{E}\zeta = \mu^{-1} \prod_{i=1}^{k} \mathrm{u}_i$, and therefore $\mathbf{E}T \leq (\prod_{i=1}^{k} \mathrm{u}_i)\mu^{-1}\Gamma \ln(2/\delta)$. By the Stopping Theorem, when the algorithm halts after $S \geq \Gamma \ln(2/\delta)$ successes, the estimate S/T approximates $\mathbf{E}\zeta$ with relative error ϵ. Thus, the estimate $\phi = (S/T) \prod_{i=1}^{k} \mathrm{u}_i$ approximates $(\prod_{i=1}^{k} \mathrm{u}_i)\mathbf{E}\zeta = \mu = \mathbf{Pr}[\mathcal{W} = w]$, also with relative error ϵ. □

Corollary 2 *For belief networks with polynomial LVB σ, the bounded-variance algorithm approximates the inference* $\mathbf{Pr}[\mathcal{W} = w]$ *with relative error* ϵ *in polynomial time* $\sigma^k \Gamma \ln(2/\delta)$.

PROOF Let $\mathbf{E}\zeta$ denote the mean of the random variable $\zeta(z, w)$ in the bounded-variance algorithm. By construction, the interval $[\prod_{i=1}^{k} \mathrm{l}_i/\mathrm{u}_i, 1]$ contains the outcomes of $\zeta(z, w)$. Thus, this interval must also contain the mean $\mathbf{E}\zeta$, and therefore $\mathbf{E}\zeta \geq \sigma^{-k}$. The proof now follows from Theorem 1, because $(\prod_{i=1}^{k} \mathrm{u}_i)/\mu = 1/\mathbf{E}\zeta \leq \sigma^k$ □

3.2.2 Discussion of the Bounded-Variance Algorithm

Corollary 2 proves that, to estimate single-evidence inference probabilities $\mathbf{Pr}[X = x|E = e]$, the bounded-variance algorithm requires *at most* polynomial time $\sigma^2 \Gamma \ln(2/\delta)$. This result is an improvement over the known convergence requirements of all other simulation algorithms. For example, consider a belief network with conditional probabilities $\mathbf{Pr}[E = e|\pi(E)]$ and $\mathbf{Pr}[X = x|\pi(X)]$ contained in the interval $[p, 10 \cdot p]$, for some small $p \leq 0.1$. The bounded-variance algorithm approximates the inference $\mathbf{Pr}[X = x|E = e]$ in time independent of the size of p, whereas forward simulation and likelihood weighting require time proportional to $1/p$. For small p, the latter algorithms are inefficient.

If $p = 0.1$—that is, the interval $[0.1, 1.0]$ contains $\mathbf{Pr}[E = e|\pi(E)]$ and $\mathbf{Pr}[X = x|\pi(X)]$—then, by Corollary 2, the bounded-variance algorithm requires at most $400(1 + \epsilon)\epsilon^{-2} \ln \frac{2}{\delta}$ samples to approximate $\mathbf{Pr}[X = x|E = e]$, *independent* of the size of the belief network. For most inferences, however, the bounded-variance algorithm halts after far fewer samples than predicted by this upper bound. If the inference probability $\mathbf{Pr}[\mathcal{W} = w]$ approaches $\prod_{i=1}^{k} \mathrm{l}_i$, then the number of samples T before halting approaches the upper bound. Otherwise, the algorithm self-corrects to account for the fewer required samples. For example, if the conditional probability $\mathbf{Pr}[X = x|E = e]$ is equal to 0.4, then by Theorem 1, the algorithm halts after at most $1/0.4\Gamma \ln(2/\delta) = 10(1+\epsilon)\epsilon^{-2} \ln(2/\delta)$ simulations and outputs an approximation of $\mathbf{Pr}[X = x|E = e]$ within relative error ϵ.

Input: $0 < \epsilon \leq 2$, $\delta > 0$, and $W_i = \omega_i$, $i = 1, ..., k$ with LVB $[\mathrm{l_i}, \mathrm{u_i}]$

Initialize $t \leftarrow 0$, $\omega \leftarrow 0$, $\Pi \leftarrow \prod_{i=1}^{k} \mathrm{u_i}$, and $S^* \leftarrow 4\ln(2/\delta)(1+\epsilon)/\epsilon^2$

Function Generate_instantiation z:

(Generates random instantiation $z_1, ..., z_{n-k}$ of $Z_1, ..., Z_{n-k}$.)

Initialize $\mathcal{Z}^* \leftarrow \{\}$ and $z^* \leftarrow \{\}$

For $i = 1$ to $n - k$ do

Choose $\alpha_i \in [0, 1]$ uniformly from $[0, 1]$.

If $\alpha_i \leq \mathbf{Pr}[Z_i = 0 | \pi(Z_i)]|_{\mathcal{W}=w, \mathcal{Z}^*=z^*}$ then $z_i \leftarrow 0$ else $z_i \leftarrow 1$

$\mathcal{Z}^* \leftarrow \mathcal{Z}^* \cup \{Z_i\}$

$z^* \leftarrow z^* \cup \{z_i\}$

Return $z \leftarrow z^*$

Algorithm:

S = 0

While $S < S^*$ do

$t \leftarrow t + 1$

Generate_instantiation z

$\omega \leftarrow \prod_{i=1}^{k} \mathbf{Pr}[W_i = \omega_i | \pi(W_i)]|_{\mathcal{W}=w, \mathcal{Z}=z}$

$S \leftarrow S + \omega/\Pi$

Let $T \leftarrow t$ denote the total number of experiments

Output: $\Pi S/T$

Figure 1: The bounded-variance algorithm.

Theorem 1 and Corollary 2 also suggest that polynomial approximation of an inference $\mathbf{Pr}[\mathcal{W} = w]$ is possible even if the LVB is not polynomially bounded. Recall that the intervals $[\mathrm{l}_i, \mathrm{u}_i]$ contain the conditional probabilities $\mathbf{Pr}[W_i = w_i | \pi(W_i)]|_{\mathcal{W}=w, \mathcal{Z}=z}$ for all instantiations $\mathcal{Z} = z$. Thus, to approximate $\mathbf{Pr}[\mathcal{W} = w]$ in polynomial time we require polynomially bounds on only the ratios $\mathrm{u}_i/\mathrm{l}_i$ for the nodes in $\mathcal{W}$, regardless of the bounds on $\mathrm{u}_i/\mathrm{l}_i$ for the nodes in the rest of the belief network.

Corollary 2 shows that the bound on the performance of the bounded-variance algorithm deteriorates as the number of evidence nodes increases. The bounded-variance algorithm guarantees polynomial-time convergence for only those inferences with a constant number of query nodes. Empirical results in real-world applications, where we may observe a large fraction of query nodes and therefore cannot run the bounded-variance algorithm to completion, suggest that the algorithm continues to provide reliable approximations, although we cannot guarantee the error in those approximations [31].

Although we may entertain the possibility that another design of a randomized algorithm might lead to polynomial solutions for inference probabilities regardless of the number of observed nodes, we prove the contrary. In Section 5, we show that even to approximate $\mathbf{Pr}[\mathcal{W} = w]$ with an *absolute* error $\epsilon < 1/2$ is **NP**-hard for large sets $\mathcal{W}$.

4 Deterministic Approximation

Deterministic-approximation algorithms, such as search-based algorithms, do not improve on the run time of randomized algorithms. Clearly, since the class of problems **RP** with randomized polynomial-time solutions contains the class of problems **P** with deterministic polynomial-time solutions, a deterministic solution requires as much or more computation than does a randomized solution. The advantages of deterministic algorithms, however, are that (1) they do not require a source of random bits; and (2) they do not have an associated failure probability. Recall that the output of a randomized algorithm fails to approximate the solution with some probability δ. Al-

though we can make this probability small, we do not know when the estimate fails to approximate the solution.

Previous deterministic-approximation algorithms are search-based algorithms that tighten incrementally the bounds on an inference probability. For example, the sum of the probabilities $\mathbf{Pr}[\mathcal{W} = w, \mathcal{Z} = z]$ over all $2^{|\mathcal{Z}|}$ instantiations of $\mathcal{Z}$ yields an exact computation of the inference $\mathbf{Pr}[\mathcal{W} = w]$. If, however, there exists a small number of instantiations $z_1, ..., z_N$ such that the probabilities $\mathbf{Pr}[\mathcal{W} = w, \mathcal{Z} = z_i]$ contribute most of the mass to the inference probability $\mathbf{Pr}[\mathcal{W} = w]$, then summing over these instantiations approximates $\mathbf{Pr}[\mathcal{W} = w]$. Unfortunately, in most cases, there does not exist a small set of instantiations that captures most of the mass of a probability. If there does exist such a small set of instantiations, then, in general, it is **NP**-hard to find [10]. Nonetheless, researchers have developed various heuristic methods that attempt to find these instantiations when possible.

We present a deterministic-approximation algorithm for probabilistic inference. Our approach is to derandomize the randomized bounded-variance algorithm. The methods that we use to derandomize the bounded-variance algorithm are, at present, applicable only to constant-depth belief networks. In contrast to the exponential worst-case behavior of search-based algorithms to output good approximations, subexponential worst-case behavior is demonstrated for the derandomized bounded-variance algorithm.

4.1 Derandomization of the Bounded-Variance Algorithm

To approximate the inference probability $\mathbf{Pr}[\mathcal{W} = w]$, the bounded-variance algorithm generates instantiations of the nodes $\mathcal{Z}$ with distribution $\rho(z, w)$, and scores the random variable $\zeta(z, w)$. Recall that, to generate instantiations with distribution $\rho(z, w)$, we first order the n belief-network nodes such that each node occurs after its parents. We then instantiate the nodes $\mathcal{W}$ to w. Thus, the remaining uninstantiated nodes $\mathcal{Z} = \{Z_1, ..., Z_{n-k}\}$ are ordered such that a parent of any node Z_i either belongs to $\mathcal{Z}$ and therefore occurs before Z_i, or belongs to the set $\mathcal{W}$ and therefore

is instantiated. We begin with the lowest ordered node Z_1 in $\mathcal{Z}$. Either the node Z_1 is a root node, or its parents $\pi(Z_1)$ belong to $\mathcal{W}$ and are instantiated. If Z_1 is a root node, then we choose a number u from the interval $[0,1]$ uniformly, and set $Z_1 = 0$ if $u \leq \mathbf{Pr}[Z_1 = 0]$, and $Z_1 = 1$ otherwise. If Z_1 is not a root node, then we let $\pi(Z_1) = z_1$ denote the instantiation of its parents. We choose a number u from the interval $[0,1]$ uniformly, and set $Z_1 = 0$ if $u \leq \mathbf{Pr}[Z_1 = 0|\pi(Z_1) = z_1]$, and $Z_1 = 1$ otherwise. Once we instantiate Z_1, we instantiate Z_2 similarly, and continue the process until we instantiate all nodes in $\mathcal{Z}$. The order on the nodes $\mathcal{Z}$ guarantees that we instantiate all the parents of a node Z_i before we instantiate Z_i. This process forms the `Generate_instantiation` function for the bounded-variance algorithm shown in Figure 1.

Instead of choosing a number u between $[0,1]$ uniformly, we can choose an m-bit string u uniformly from all m-bit strings. For example, if we let U denote the integer representation of u, then we set $Z_1 = 0$ if $U/2^m \leq \mathbf{Pr}[Z_1 = 0|\pi(Z_1) = z_1]$, and we set $Z_1 = 1$ otherwise. Thus, we instantiate $Z_1 = 0$ with probability $U/2^m$ that approximates $\mathbf{Pr}[Z_1 = 0|\pi(Z_1) = z_1]$ with absolute error $1/2^m$. We assume for now that we choose m sufficiently large to make this error insignificant in the computation of an inference probability. (In Section A.3 in the appendix, we show how to choose m to bound this error.) Thus, to generate an instantiation of the nodes $\mathcal{Z}$ with distribution $\rho(z,w)$, we choose an nm-bit string uniformly from the space of all 2^{nm} strings of length nm. We use the first m bits to instantiate Z_1, the second m bits to instantiate Z_2, and so forth, until we instantiate all nodes in $\mathcal{Z}$. If we score the outcomes of the random variable $\zeta(z,w)$ after several instantiations of $\mathcal{Z}$, then the mean of $\zeta(z,w)$ approximates $\mathbf{Pr}[\mathcal{W} = w] / \prod_{i=1}^{k} \mathrm{u}_i$.

Recall that Monte Carlo theory dictates that, with some nonzero probability, we approximate any inference probability $\mathbf{Pr}[\mathcal{W} = w]$ within absolute error ϵ after only $O(1/\epsilon^2)$ trials. In other words, the theory proves that there exists a subset Φ of nm-bit strings of size $O(1/\epsilon^2)$ such that, if we score the random variable $\zeta(z,w)$ on the instantiations of $\mathcal{Z}$ generated from this subset, then the mean approximates

$\mathbf{Pr}[\mathcal{W} = w] / \prod_{i=1}^{k} \mathrm{u}_i$. Although we do not know how to find deterministically a set of size $O(1/\epsilon^2)$, we show next that we can find a set of subexponential size that approximates $\mathbf{Pr}[\mathcal{W} = w]$ with an *absolute* error $1/n^q$ for any integer q. From this result, we prove that, for constant depth belief networks without extreme conditional probabilities, we can approximate $\mathbf{Pr}[X = x | E = e]$ in subexponential worst-case time within *relative* error $1/n^q$ for any integer q. Thus, the deterministic specification of an input set Φ on which to evaluate the function `Generate_instantiation` provides the key to derandomizing the bounded-variance algorithm.

Observe that we can compute $\mathbf{Pr}[\mathcal{W} = w] / \prod_{i=1}^{k} \mathrm{u}_i$ exactly as follows. We cycle over all 2^{nm} possible instantiations of nm bits and, for each instantiation, we generate an instantiation of $\mathcal{Z}$. We score the random variable $\zeta(z, w)$ for each instantiation of $\mathcal{Z}$. The mean of the 2^{nm} values for $\zeta(z, w)$ yields $\mathbf{Pr}[\mathcal{W} = w] / \prod_{i=1}^{k} \mathrm{u}_i$. Instead of cycling over the set of 2^{nm} instantiations of nm bits, however, we prove that we can cycle over a subset Φ of subexponential size. Let d denote the depth of the belief network, let $\bar{d} = 5(d+1)$, and let $l(n) = (\log(nm))^{2\bar{d}+6}$. We construct the set Φ from the set of $2^{l(n)}$ bit strings of length $l(n)$, stretched into length nm-bit strings by special binary-valued matrices $\mathcal{A}_{nm,l(n)}$ of size $nm \times l(n)$. (Section A.2 in the appendix describes the construction of these matrices.) For each string in Φ, we generate an instantiation of $\mathcal{Z}$, and we score the random variable $\zeta(z, w)$. The mean of $\zeta(z, w)$ evaluated at all $2^{l(n)}$ instantiations of $\mathcal{Z}$ generated from the set Φ is a deterministic approximation of the inference probability $\mathbf{Pr}[\mathcal{W} = w] / \prod_{i=1}^{k} \mathrm{u}_i$. This algorithm defines the *derandomized bounded-variance algorithm*, or simply the *derandomized algorithm*, to approximate inference probabilities. We summarize this algorithm in Figure 2; in Section 4.2.3, we prove that this approximation is within relative error $1/n^q$ of $\mathbf{Pr}[\mathcal{W} = w]$ for any integer q.

4.2 Proof of Subexponential Runtime

We first discuss Boolean circuits as a model of computation and we then prove subexponential runtime using this model.

Input: depth d, LVB σ, and $W_i = \omega_i$, $i = 1, ..., k$

Function Construct_samplespace Φ:

Initialize $\Phi \leftarrow \{\}$, $\bar{d} \leftarrow 5(d+1)$, $m \leftarrow 2\log(n\sigma)$, and $l \leftarrow (\log(nm))^{2\bar{d}+6}$

Construct matrix $\mathcal{A}_{nm,l}$ defined in Section A.2 in the appendix

For $i = 0$ to $2^l - 1$ do

$v \leftarrow$the l-bit binary representation of i

$u \leftarrow \mathcal{A}_{nm,l} v$

$\Phi \leftarrow \Phi \cup \{u\}$

Return Φ

Function Generate_instantiation z from $u \in \Phi$:

(Generates instantiation $z_1, ..., z_{n-k}$ of $Z_1, ..., Z_{n-k}$ from u.)

Initialize $\mathcal{Z}^* \leftarrow \{\}$, $z^* \leftarrow \{\}$, and let $u \leftarrow (u_0^1, ..., u_{m-1}^1, ..., u_0^n, ..., u_{m-1}^n)$

For $i = 1$ to $n - k$ do

$U_i \leftarrow u_0^i 2^0 + \cdots + u_{m-1}^i 2^{m-1}$

$\alpha_i \leftarrow U_i / 2^m$

If $\alpha_i \leq \mathbf{Pr}[Z_i = 0 | \pi(Z_i)]|_{\mathcal{W}=w, \mathcal{Z}^*=z^*}$ then $z_i \leftarrow 0$ else $z_i \leftarrow$ 1

$\mathcal{Z}^* \leftarrow \mathcal{Z}^* \cup \{Z_i\}$

$z^* \leftarrow z^* \cup \{z_i\}$

Return $z \leftarrow z^*$

Algorithm:

Construct_samplespace Φ

For all $u \in \Phi$ do

Generate_instantiation z from u

$\omega \leftarrow \prod_{i=1}^k \mathbf{Pr}[W_i = \omega_i | \pi(W_i)]|_{\mathcal{W}=w, \mathcal{Z}=z}$

$S \leftarrow S + \omega / \Pi$

Output: $\Pi S / |\Phi|$

Figure 2: The deterministic bounded-variance algorithm.

4.2.1 Boolean Circuits

We discuss a model of computation for which Nisan [24] proves that we can stretch a short string of truly random bits into a long string of pseudorandom bits that appears random to this model. These models are *constant-depth, unbounded fan-in Boolean circuits*, and consist of a directed acyclic graph (DAG) on a set of s binary-valued *input nodes* $U_1, ..., U_s$, t binary-valued *gate nodes* $Y_1, ..., Y_t$ where t is polynomial in s, and one binary-valued *output node* O. In the DAG, the input nodes are the only source nodes, and the output node is the only sink node. The number of parents of a gate node and of the output node is unbounded—for example, the output node may have all other nodes as parents. Each gate node and the output node determines its value from the values of its parent nodes by one of three Boolean operations, "and", "or" and "not"; we define three types of nodes, *and*-nodes, *or*-nodes and *not*-nodes. The value of an *and*-node is the "and" of the parent nodes, the value of an *or*-node is the "or" of the parent nodes, and the value of a *not*- node is the "not" of its parent node. Figures 3(a), (b), and (c) depict the Boolean circuits for those three Boolean operations. The *size* of the circuit is the number of nodes in the circuit. The *depth* of the circuit is the longest directed distance in the DAG between an input node and the output node. For *constant-depth* circuits, the depth is not a function of the size of the circuit.

Henceforth, we use *circuits* synonymous with constant-depth, unbounded fan-in Boolean circuits. Nisan gives a method that stretches $(\log s)^{2d+6}$ truly random bits into s bits that appear random to any family of circuits of size polynomial in s and depth d. Specifically, Nisan proves the following result.

Theorem 3 *[24] Let $\{C_s\}$ denote a family of circuits of depth d and size polynomial in the number of inputs s, and let $l = (\log s)^{2d+6}$. There exists an easily constructible family of $s \times l$ matrices $\{\mathcal{A}_{s,l}\}$ such that, for any integer q and $\epsilon = 1/s^q$,*

$$\mathbf{Pr}[C_s(y) = 0] - \epsilon \leq \mathbf{Pr}[C_s(\mathcal{A}_{s,l}u) = 0] \leq \mathbf{Pr}[C_s(y) = 0] + \epsilon,$$

where y is chosen uniformly over all s-bit strings, u is chosen uniformly over all l-bit

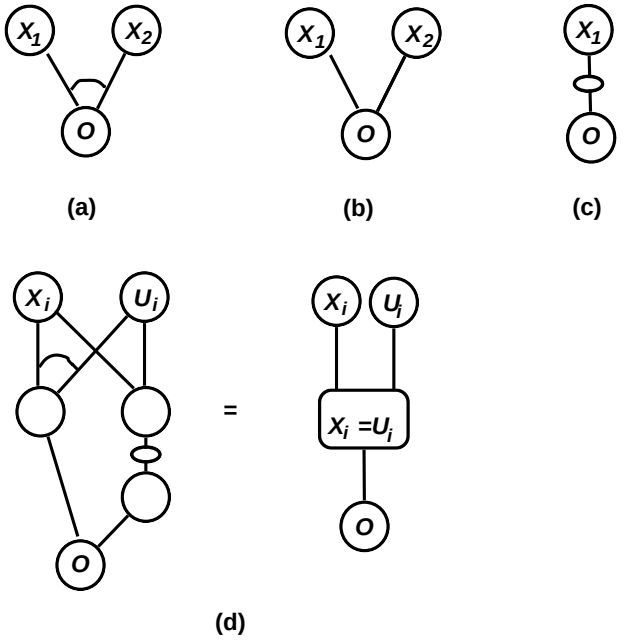

Figure 3: Boolean circuits. (a), (b), and (c) Boolean circuits corresponding to the operations $x_1 \wedge x_2$, $x_1 \vee x_2$, and $\neg x_1$, respectively. (d) Boolean circuit that tests $x_i = u_i$ and a schematic representation of that circuit.

strings, and $\mathcal{A}_{s,l}u$ denotes modulo 2 matrix-vector multiplication of u with $\mathcal{A}_{s,l}$.

In Section A.2 in the appendix, we describe Nisan's design of the matrices $\mathcal{A}_{s,l}$ in sufficient detail to allow their implementation. These matrices effectively stretch $l = (\log s)^{2d+6}$ truly random bits u into s bits $\mathcal{A}_{s,l}u$ that the circuit C_s cannot distinguish from s truly random bits y, to within an error $\epsilon = 1/s^{O(1)}$. We can put this result in the context of the discussion of Section 2. Let Ω denote the set of all 2^s instantiations y, and let $S \subseteq \Omega$ denote the subset of inputs y such that $C_s(y) = 0$. The probability $\mathbf{Pr}[C_s(y) = 0]$ denotes the fraction $|S|/|\Omega|$ of all 2^s inputs Ω on which the circuit C_s outputs 0. Let Φ denote the subset of Ω constructed from the s-bit strings $\mathcal{A}_{s,l}u$ for all $2^{(\log s)^{2d+6}}$ inputs u, and let $S' \subseteq \Phi$ denote the subset of strings $\mathcal{A}_{s,l}u$ on which the circuit C_s outputs 0. The second probability $\mathbf{Pr}[C_s(\mathcal{A}_{s,l}u) = 0]$ denotes the fraction $|S'|/|\Phi|$ of all $2^{(\log s)^{2d+6}}$ inputs Φ on which the circuit C_s outputs 0. Theorem 3 states that $|S'|/|\Phi|$ approximates $|S|/|\Omega|$ with absolute error ϵ. Thus, we construct a deterministic approximation of $|S|/|\Omega|$ with absolute error ϵ by enumerating all $2^{(\log s)^{2d+6}}$ bit strings u, computing the output $C_s(\mathcal{A}_{s,l}u)$, and scoring every output that is equal to 0. If the polynomial $p(s)$ denotes the size of the circuit C_s, then we can evaluate C_s on any input in time $p(s)$; the approximation algorithm requires $O(p(s)2^{(\log s)^{2d+6}})$ computations.

We use Theorem 3 to derandomize the bounded-variance algorithm, and thus to construct a deterministic-approximation algorithm for probabilistic inference. We must overcome several difficulties first, however. First, we must prove that we can implement in a circuit the function `Generate_instantiation` shown in Figure 1. Once we construct this circuit, we use the matrices $\mathcal{A}_{s,l}$ described in Section A.2 in the appendix to generate a set of input strings Φ to the circuit. We must then prove that the approximation of an inference probability based on the set of inputs Φ is within an absolute error ϵ of the exact computation based on all the inputs. Theorem 3 proves this property for only those circuits with a single output bit, whereas circuits that compute inference probabilities must output many bits—for example, we require n bits to express the output probability $1/2^n$.

In Section 4.2.2, we construct a circuit simulation of the function `Generate_instantiation` z `from` $u \in \Phi$ that appears in the derandomized algorithm in Figure 2. In Section 4.2.3, we use this circuit simulation to prove that, if we score the random variable $\zeta(z, w)$ on the output from `Generate_instantiation` z `from` $u \in \Phi$ evaluated on the subexponential number of inputs Φ, then we produce a relative-error deterministic approximation of $\mathbf{Pr}[\mathcal{W} = w]$. We use the circuit simulation only for proof purposes; we use the algorithm presented in Figure 2 for the implementation.

4.2.2 Circuit Implementation

We described how to derandomize the bounded-variance algorithm into a deterministic-approximation algorithm that we refer to as the *derandomized algorithm.* The correctness of the derandomized algorithm relies on the proof that a circuit of constant depth can simulate `Generate_instantiation` z `from` $u \in \Phi$. In this section, we construct a circuit of depth $\bar{d} = 5(d + 1)$ that simulates the `Generate_instantiation` z `from` $u \in \Phi$ for belief networks of depth d.

We first prove that elementary bit-string relations are verifiable by constant-depth circuits.

Lemma 4 *Let x denote an s-bit string. There exists a circuit of depth 4 that, for any s-bit string u, outputs 1 if $u = x$, and outputs 0 otherwise. Similarly, there exists a circuit of depth 5 that outputs 1 if $u > x$, and outputs 0 otherwise.*

PROOF First observe that we can verify whether the ith bits are equal, $u_i = x_i$, in a depth 3 circuit. This result follows because $u_i = x_i$ if and only if $(u_i \wedge x_i) \vee \neg(u_i \vee x_i)$. Thus, the circuit has input nodes x_i and u_i, an *and*-gate node that computes $u_i \wedge x_i$, an *or*-node that computes $u_i \vee x_i$, a *not*-node that negates $u_i \vee x_i$, and an *or*-output node that computes $(u_i \wedge x_i) \vee \neg(u_i \vee x_i)$. Figure 3(d) illustrates that circuit. To verify that all s bits are equal, we observe that $u = x$ if and only if $\forall_{i=1}^{s} u_i = x_i$. Thus, if we verify all s relations $u_i = x_i$ individually by s depth 3 circuits, and we output

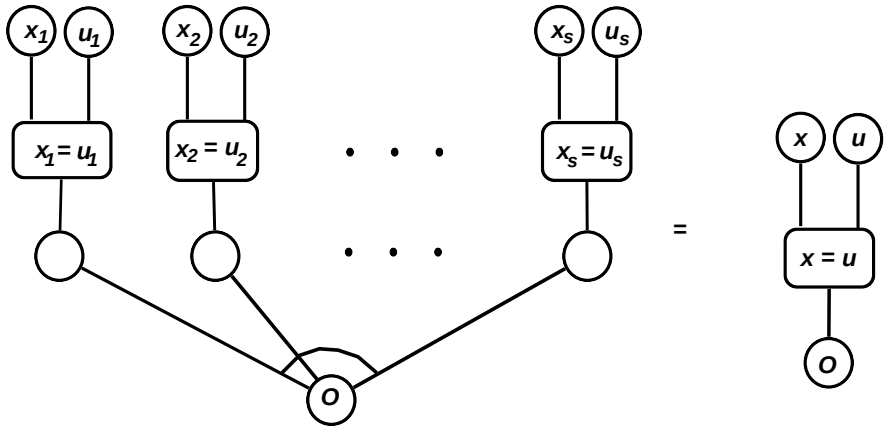

Figure 4: A circuit that tests $x = u$ for length s bit strings $x = \{x_1, ..., x_s\}$ and $u = \{u_1, ..., u_s\}$. We used the schematic representation of Figure 3(d) for the bit-wise comparison circuits.

the "and" of the s outputs from the s circuits, then we output 1 if and only if $u_i = x_i$ for all i. We illustrate that circuit in Figure 4.

The relation $u > x$ is satisfied if there exists some $0 \leq k \leq s-1$ such that u and x agree on the first k bits, and $u_{k+1} = 1$ and $x_{k+1} = 0$. Thus, $u > x$ if and only if

$$\exists_{k \in \{0,...,s-1\}} \forall_{i \leq k} (u_i = x_i) \wedge u_{k+1} \wedge \neg x_{k+1}.$$

For each k, we can verify $\forall_{i \leq k}(u_i = x_i) \wedge (u_{k+1} \wedge \neg x_{k+1})$ in a depth 4 circuit. The "or" of the outputs from s such circuits, one for each k, verifies the relation $u > x$ in a depth 5 circuit. □

We now prove that, for any belief-network node X, we can construct a depth 5 circuit C_X, such that, for any instantiation $y = \{y_1, ..., y_t\}$ of $\pi(X) = \{Y_1, ..., Y_t\}$ and input string u of length m, $C_X(u, y) = 0$ if and only if the integer representation U of u satisfies $U \leq 2^m \mathbf{Pr}[X = 0 | \pi(X) = y]$. Figure 5 shows the circuit C_X. Thus, on input $\pi(X) = y$ and a randomly chosen input u, both the circuit C_X and the derandomized algorithm set $X = 0$ with the same probability.

Lemma 5 *Let X and $\pi(X)$ denote a belief-network node and its parents, respectively. There exists a circuit C_X of depth 5 such that, for any instantiation $\pi(X) = y$ and*

input string u of length m, $C_X(u, y) = 0$ if and only if the integer representation U of u satisfies $U \leq 2^m \mathbf{Pr}[X = 0 | \pi(X) = y]$.

PROOF Let t denote the number of parents $\pi(X)$. Thus, the instantiation y of the parent nodes $\pi(X)$ is a t-bit string. We let $y^1, ..., y^{2^t}$ enumerate all possible strings y. For all $i = 1, ..., 2^t$, let p^i denote the binary expression of the integer $\lfloor 2^m \mathbf{Pr}[X = 0 | \pi(X) = y^i] \rfloor$. We construct a circuit C_X that, on inputs u and y, outputs

$$\exists_{i \in \{1,...,2^t\}} (y = y^i) \wedge (u > p^i). \tag{2}$$

Observe that, by Lemma 4, we can output $y = y^i$ with a circuit of depth 4, and $u > p^i$ with a circuit of depth 5. Thus, we can compute $(y = y^i) \wedge (u > p^i)$ in a circuit of depth 6, and verify that there exists an i that satisfies this equation in a depth 7 circuit. We prove, however, that a circuit of depth 5 suffices to compute Equation 2.

We substitute the expression of Lemma 4 for the relation $y = y^i$ into Equation 2. Rearranging, we get the following equation:

$$\exists_{i \in \{1,...,2^t\}} \exists_{r \in \{0,...,m-1\}} [\forall_{l \leq r} \forall_{0 \leq j \leq t} (y_j = y_j^i) \wedge (u_l = p_l^i) \wedge \neg p_{l+1}^i \wedge u_{l+1}].$$

Since we can compute $\forall_{l \leq r} \forall_{0 \leq j \leq t} (y_j = y_j^i) \wedge (u_l = p_l^i) \wedge \neg p_{l+1}^i \wedge u_{l+1}$ in a depth 4 circuit, we can compute the former expression in a depth 5 circuit. □

We can easily show that these circuits allow us to simulate the function `Generate_instantiation` z `from` $u \in \Phi$ in constant-depth circuits for constant-depth belief networks. We construct circuits $C_{X_1}, ..., C_{X_n}$ for the n belief-network nodes, and connect them into a circuit $\mathcal{C}$ such that, if $X_i \in \pi(X_j)$, then the output of C_{X_i} is also an input to C_{X_j}. Figure 6 shows an example of a circuit $\mathcal{C}$ for a five-node belief network. Thus, to simulate `Generate_instantiation` z `from` $u \in \Phi$, we proceed as follows. We first set the output node of each circuit C_{W_i} in $\mathcal{C}$ to w_i. Let $C_{Z_1}, ..., C_{Z_{n-k}}$ denote an ordering of the remaining circuits in $\mathcal{C}$ according to the order $Z_1, ..., Z_{n-k}$ imposed by the derandomized algorithm—that is, the parents $\pi(Z_i)$ of any node Z_i occur before that node in the order. We now choose an nm-bit string uniformly from the space of all 2^{nm} strings, and use the first m bits as inputs to the circuit C_{Z_1}, the second

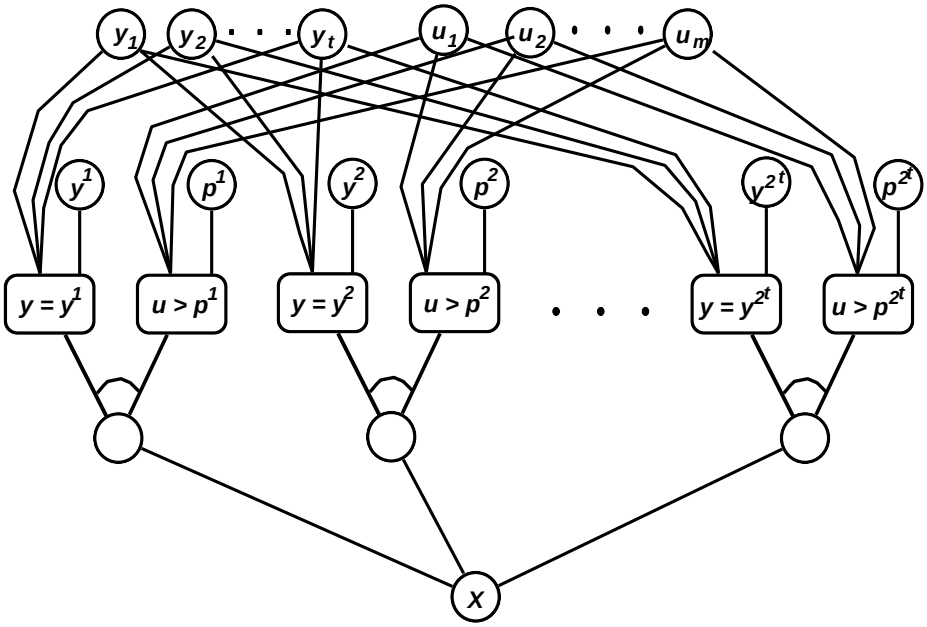

Figure 5: The circuit C_X for node X, parents $\pi(X) = \{Y_1, ..., Y_t\}$ and conditional probabilities $p^1, ..., p^{2^t}$. The parents have 2^t possible instantiations $y^1, ..., y^{2^t}$. Each conditional probability p^i represents the m-bit string $\lfloor 2^m \mathbf{Pr}[X = 0 | \pi(X) = y^i] \rfloor$. On input of an m-bit string $u = \{u_1, ..., u_m\}$ and an instantiation of the parents $y = \{y_1, ..., y_t\}$, the circuit outputs 0 if and only if the integer representation U of u satisfies $U \leq 2^m \mathbf{Pr}[X = 0 | \pi(X) = y]$.

m bits and the output of C_{Z_1} if $Z_1 \in \pi(Z_2)$ as inputs to the circuit C_{Z_2}, the third m bits and the outputs of C_{Z_1} and C_{Z_2} if either Z_1 or Z_2 belongs to $\pi(Z_3)$ as inputs to C_{Z_3}, and so forth. Thus, from the outputs of $C_{Z_1}, ..., C_{Z_{n-k}}$, for a randomly chosen nm-bit string, we generate instantiations z of $\mathcal{Z}$ with the same probability as `Generate_instantiation` z `from` $u \in \Phi$. We use the instantiation z to score the random variable $\zeta(z, w)$.

4.2.3 Proof of Subexponential Convergence

In this section, we prove that the output of the derandomized algorithm approximates inference probabilities within relative error $1/n^q$ for any integer q. The essence of the proof uses the result of Theorem 3.

Let σ denote the LVB of a belief network on n nodes and of depth d. As before, define $\bar{d} = 5(d+1)$, and let $l(n) = (\log(nm)^{2\bar{d}+6}$, where m is chosen according to Section A.3 in the appendix. We use Φ to denote the set of $2^{l(n)}$ binary strings of length nm, constructed according to Section A.2 in the appendix. Let Ω denote the space of all 2^{nm} bit strings of length nm. We let $\mathbf{E}_\Omega \zeta$ denote the mean of the random variable $\zeta(z, w)$ evaluated on the 2^{nm} instantiations $\mathcal{Z} = z$ generated from the different nm-bit strings.

Theorem 6 *For belief networks of depth d and polynomial bounded LVB, on input strings Φ, the output of the derandomized algorithm approximates any inference probability $\mathbf{Pr}[\mathcal{W} = w]$ for $\mathcal{W}$ of constant size within relative error $\epsilon < 1/n^q$ for any integer q.*

PROOF Let Ω denote the set of all nm-bit strings. From Section A.3 in the appendix, if $m = b \log n$ for some sufficiently large constant b, then we can disregard the error that we make in approximating $\mathbf{Pr}[\mathcal{W} = w]$ by $(\prod_{i=1}^{k} u_i)\mathbf{E}_\Omega \zeta$.

Let $\mathcal{Z}^h = \{Z_1, ..., Z_h\}$ denote the nodes in $\mathcal{Z}$ that determine the outcome of the random variable $\zeta(z, w)$. For any instantiation $\mathcal{Z} = z$, let z^h denote z's instantiation of the subset of nodes $\mathcal{Z}^h$. Thus, $\zeta(z^h, w) = \zeta(z, w)$. Let $z_1^h, ..., z_{2^h}^h$ denote all 2^h instan-

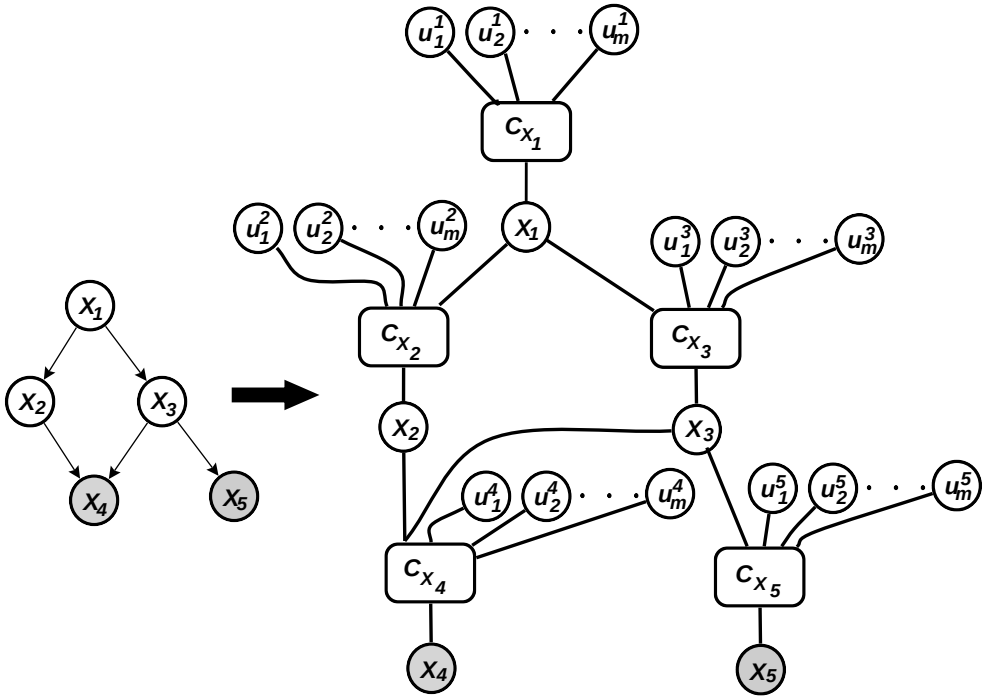

Figure 6: A five-node belief network with evidence nodes X_4 and X_5, and the corresponding circuit $\mathcal{C}$ that simulates `Generate_instantiation` z `from` $u \in \Phi$. In this example, $\mathcal{Z} = \{X_1, X_2, X_3\}$ and $\mathcal{W} = \{X_4, X_5\}$. The outputs of circuits C_{X_4} and C_{X_5} are fixed by the evidence values of X_4 and X_5; therefore, we need to run the circuit $\mathcal{C}$ with only the first $3m$ randomly chosen bits of the input string $u = u_1^1 \cdots u_m^1 u_1^2 \cdots u_m^2 \cdots u_1^5 \cdots u_m^5$. The instantiation z on input u corresponds to the settings of the nodes X_1, X_2, and X_3 for that input.

tiations of z^h. We partition the set Ω into 2^h subsets $\Omega_1, ..., \Omega_{2^h}$ such that Ω_i contains all instantiations $z \in \Omega$ that instantiate $\mathcal{Z}^h$ to z_i^h. We partition Φ similarly into sets $\Phi_1, ..., \Phi_{2^h}$. Thus, $\mathbf{E}_\Omega \zeta = 1/2^{nm} \sum_{i=1}^{2^h} |\Omega_i| \zeta(z_i^h, w)$ and $\mathbf{E}_\Phi \zeta = 1/2^{l(n)} \sum_{i=1}^{2^h} |\Phi_i| \zeta(z_i^h, w)$. We show next that, for all $i = 1, ..., 2^h$, $|\Phi_i|/2^{l(n)}$ approximates $|\Omega_i|/2^{nm}$ within absolute error $1/n^q$ for any integer q.

For each $i = 1, ..., 2^h$ we construct a depth 4 circuit C_i with input nodes $\mathcal{Z}^h$ such that, for any input string $\mathcal{Z}^h = z^h$, $C_i(z^h) = 0$ if and only if $z^h = z_i^h$. Let $\mathcal{C}_i$ denote the circuit $\mathcal{C}$ connected to the circuit C_i. Thus, $\mathbf{Pr}[\mathcal{C}_i(z) = 0] = |\Omega_i|/2^{nm}$, where z is an nm-bit string chosen uniformly from Ω. To choose a string from Φ uniformly, we choose an $l(n)$-bit string u uniformly from all $2^{l(n)}$ such strings, and stretch u into the string $\mathcal{A}_{nm,l(n)}u$ in Φ. Thus, $\mathbf{Pr}[\mathcal{C}_i(\mathcal{A}_{nm,l(n)}u) = 0] = |\Phi_i|/2^{l(n)}$. Since $\mathcal{C}_i$ has depth less than $\bar{d} = 5(d+1)$, from Theorem 3, $\mathbf{Pr}[\mathcal{C}_i(\mathcal{A}_{nm,l(n)}u) = 0]$ approximates $\mathbf{Pr}[\mathcal{C}_i(z) = 0]$ within absolute error $1/n^q$ for any integer q.

We have shown that $\mathbf{E}_\Phi \zeta$ approximates $\mathbf{E}_\Omega \zeta$ within absolute error $2^h/n^q$ for any integer q. To convert this approximation to a relative error approximation, we observe that (1) for some integer c, $2^h \leq n^{ck}$, since $|\mathcal{W}| = k$ and each belief-network node can have at most $O(\log n)$ parents (recall that we bound the size of the conditional probabilities by a polynomial in n); and (2) the LVB is at most n^r, and therefore $\mathbf{E}_\Omega \zeta \geq n^{-kr}$. Thus, a n^{-q} absolute-error approximation is a $n^{-q+(c+r)k}$ relative-error approximation. □

5 Complexity of Approximation, Revisited

We have shown that, for belief networks with polynomially bounded LVB—that is, for belief networks without extreme conditional probabilities—we can approximate efficiently any inference probability $\mathbf{Pr}[X = x | \mathcal{E} = e]$, where the size of the set $\mathcal{E}$ is constant. In contrast to these results, we prove that, when the size of $\mathcal{E}$ is a large fraction of the number of nodes in the belief network, we cannot approximate inference probabilities, even for belief networks with LVBs near 1.

Theorem 7 *Consider the class of belief networks with LVB* $< 1 + c$ *for any constant* $c > 0$. *If there exists an algorithm to approximate inferences* $\mathbf{Pr}[X = x | \mathcal{E} = e]$ *for evidence set* $\mathcal{E}$ *of size* n^γ *for any constant* $\gamma > 0$, *then, for any constant* $d > 0$, *(1) if this algorithm is deterministic and the approximation is within absolute error less than* $1/2 - d$, *then* $\mathbf{NP} \subseteq \mathbf{P}$; *and (2) if this algorithm is randomized and the approximation is within absolute error less than* $1/2 - d$ *with probability greater than* $1/2 + d$, *then* $\mathbf{NP} \subseteq \mathbf{RP}$.

PROOF The proof is similar to the proof of the complexity of approximating probabilistic inference with a single evidence node [10]. Let F denote an instance of 3-SAT with variables $V = \{V_1, \ldots, V_n\}$ and clauses $C = \{C_1, \ldots, C_m\}$. The formula F defines the belief network BN with binary-valued nodes $V \cup C^1 \cup \cdots \cup C^M$, where, for each $k = 1, ...M$, the set of nodes $C^k = \{C_1^k, ..., C_m^k\}$ represents a copy of the set of clauses C. Arcs are directed from node V_i to all M nodes C_j^k, $k = 1, ..., M$, if and only if variable V_i appears in clause C_j. For example, Figure 7 shows BN with $M = 2$ for

$$F = (V_1 \vee V_2 \vee V_3) \wedge (\neg V_1 \vee \neg V_2 \vee V_3) \wedge (V_2 \vee \neg V_3 \vee V_4).$$

Each node V_i is given a prior probability $1/2$ of being instantiated to 0 or 1. For any clause C_j, let $j1, j2, j3$ index the three variables in V contained in clause C_j. The conditional probabilities associated with the M nodes C_j^k, $k = 1, ..., M$, each having parent nodes $\{V_{j1}, V_{j2}, V_{j3}\}$ in the belief network, are defined by

$$\mathbf{Pr}[C_j^k = 1 | \{V_{ji} = v_{ji} : i = 1, 2, 3\}] = \begin{cases} \text{u}, & \text{if } \{V_{ji} = v_{ji} : i = 1, 2, 3\} \text{ satisfies } C_j; \\ \text{l}, & \text{otherwise;} \end{cases}$$

where $0 \leq \text{l} < \text{u} \leq 1$.

For $k = 1, ..., M$, let $C^k = 1$ denote the instantiation $C_i^k = 1$ for all $i = 1, ..., m$. Assume that F has at least one satisfying assignment—that is, $\mathbf{Pr}[C^k = 1] > 0$. We determine the truth assignment that satisfies F one variable at a time, starting with finding a value v_1 for V_1. For $a \in \{0, 1\}$, let Z^a approximate $\mathbf{Pr}[V_1 = a | C^1 = 1, ..., C^M = 1]$ with absolute error $1/2 - d$ for any constant $d > 0$. Observe that, since

$\mathbf{Pr}[V_1 = a] = \mathbf{Pr}[V_1 = 1 - a]$,

$$\frac{\mathbf{Pr}[V_1 = a | C^1 = 1, ..., C^M = 1]}{\mathbf{Pr}[V_1 = 1 - a | C^1 = 1, ..., C^M = 1]} = \frac{\mathbf{Pr}[C^1 = 1, ..., C^M = 1 | V_1 = a]}{\mathbf{Pr}[C^1 = 1, ..., C^M = 1 | V_1 = 1 - a]}.$$

Let r denote this ratio, and let $P^x = \mathbf{Pr}[V_1 = x | C^1 = 1, ..., C^M = 1]$. Thus, $P^a = rP^{1-a} = r(1 - P^a)$; therefore, $P^a = 1/(1 + r^{-1})$. We next show that, if F is not satisfiable with $V_1 = 1 - a$, then $r \geq (1 + c)^M / 2^n$, and therefore,

$$P^a \geq \frac{1}{1 + 2^n/(1 + c)^M}.$$

For sufficiently large M—that is, for $M \geq \frac{1}{\log(1+c)} \left[n - \log \frac{d}{1-d} \right]$— we get that, if F is not satisfiable with $V_1 = 1 - a$, then $P^a > 1 - d$. Thus, if Z^a approximates P^a within absolute error $1/2 - d$, then we can determine a truth assignment of V_1 that leads to a satisfying assignment of F: If $Z^a > 1/2$, we set $V_1 = a$; otherwise, $V_1 = 1 - a$.

Let $\mathrm{U} = \mathrm{u}^M$ and $\mathrm{L} = \mathrm{l}^M$. Let Ω_i^a denote the set of assignments of the variables $V_2, ..., V_n$ with $V_1 = a$ that satisfy exactly $m - i$ clauses in C, and let $N_i^a = |\Omega_i^a|$. Thus,

$$\begin{aligned}
\mathbf{Pr}[C^1 = 1, ..., C^M = 1 | V_1 = a] &= \sum_{V_2, ..., V_n} \mathbf{Pr}[C^1 = 1, ..., C^M = 1 | V_1 = a, V_2, ..., V_n] \mathbf{Pr}[V_2, ..., V_n] \\
&= \sum_{i=0}^{m} (\mathbf{Pr}[C^1 = 1 | V_1 = a, \Omega_i^a])^M \mathbf{Pr}[\Omega_i^a] \\
&= \sum_{i=0}^{m} (\mathrm{l}^i \mathrm{u}^{m-i})^M \frac{N_i^a}{2^{n-1}} \\
&= \frac{U^m}{2^{n-1}} \sum_{i=0}^{m} N_i^a (\mathrm{L}/\mathrm{U})^i.
\end{aligned}$$

Similarly, we can show that

$$\mathbf{Pr}[C^1 = 1, ..., C^M = 1 | V_1 = 1 - a] = \frac{U^m}{2^{n-1}} \sum_{i=0}^{m} N_i^{1-a} (\mathrm{L}/\mathrm{U})^i.$$

If $V_1 = 1 - a$ does not lead to a satisfying assignment of F, then $N_0^{1-a} = 0$, and, since F is satisfiable by assumption, $N_0^a \geq 1$. Thus,

$$r = \frac{N_0^a + (\mathrm{L}/\mathrm{U}) N_1^a + \cdots + (\mathrm{L}/\mathrm{U})^m N_m^a}{(\mathrm{L}/\mathrm{U}) N_1^{1-a} + \cdots + (\mathrm{L}/\mathrm{U})^m N_m^{1-a}} \geq \frac{1}{(\mathrm{L}/\mathrm{U}) 2^n} \geq (1 + c)^M / 2^n,$$

where the inequalities follow, because $\sum_{i=1}^{m} N_i^{1-a} = 2^n$ when $N_0^{1-a} = 0$, and because u/l $= 1 + c$.

To find the truth setting for the second variable V_2, we proceed as follows. For every child C_i^k of V_1 in BN, we make the following modification to BN. Let C_i^k have parents V_1, V', V''. We redefine the conditional probabilities for each child C_i^k as

$$\mathbf{Pr}[C_i^k | V', V''] = \mathbf{Pr}[C_i^k | V_1 = v_1, V', V''].$$

We then delete node V_1 from BN, and let BN' denote the resulting belief network, and let $\mathbf{Pr}'$ denote the full joint probability distribution for BN'. Thus, we can find a truth assignment v_2 for V_2 in BN' in exactly the same way as we found a truth assignment v_1 for V_1 in BN. Proceeding in this way, we find a truth assignment for all the variables. This assignment is guaranteed to satisfy the original formula F under the assumption that F is satisfiable. If F is not satisfiable, then the algorithm terminates with an instantiation of the nodes $V_1, \ldots, V_n$ that does not satisfy F. Therefore, we can determine whether or not F is satisfiable by running the algorithm and checking whether or not $v_1, \ldots, v_n$ satisfies F.

We can prove an analogous result with respect to randomized algorithms. The proof applies the same methods used in [10] to the preceding construction.

The size of the evidence set $C^1 \cup \cdots \cup C^M$ in the inference probability $\mathbf{Pr}[V_i = x | C^1 = 1, ..., C^M = 1]$ is $S = mM$, where M is $O(n)$, and the number of nodes in the belief network BN is $N = n + mM$. For any constant $\gamma < 1$, however, we can add to BN $S^{1/\gamma - 1}$ dummy nodes—that is, nodes not connected to BN. These nodes do not change the values of the conditional probabilities $\mathbf{Pr}[V_i | C^1 = 1, ..., C^M = 1]$. If N' denotes the number of nodes in the new belief network, then the size the evidence set S in the new belief network is $O(N'^{\gamma})$. □

6 Conclusions

We proved that, for constant-sized evidence sets $\mathcal{E}$, we can generate relative approximations of inferences $\mathbf{Pr}[X = x | \mathcal{E} = e]$ in polynomial time for belief networks without

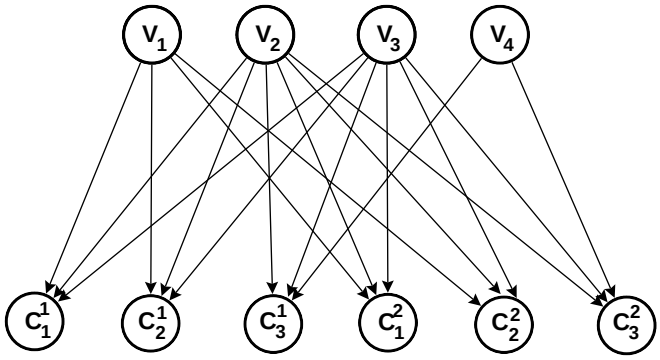

Figure 7: Belief-network structure with $M = 2$ for the 3-SAT instance $F = (V_1 \vee V_2 \vee V_3) \wedge (\neg V_1 \vee \neg V_2 \vee V_3) \wedge (V_2 \vee \neg V_3 \vee V_4)$.

extreme conditional probabilities. We also proved that we can generate these approximations deterministically in subexponential time. We showed that our results also apply to belief networks with extreme conditional probabilities, provided that the conditional probabilities of nodes X and $\mathcal{E}$ of the inference $\mathbf{Pr}[X = x|\mathcal{E} = e]$ are not extreme. Thus, even if all the other conditional probabilities assume 0 or 1 values, we can approximate the inference probability $\mathbf{Pr}[X = x|\mathcal{E} = e]$ with high probability in polynomial time, and deterministically in subexponential time. In addition, we proved that, when the size of the evidence set $\mathcal{E}$ is large, then we cannot approximate $\mathbf{Pr}[X = x|\mathcal{E} = e]$ unless either $\mathbf{P} \subseteq \mathbf{NP}$ or $\mathbf{RP} \subseteq \mathbf{NP}$.

Appendix

A.1: Stopping Rule

Let $Z_1, Z_2, \ldots$ denote independently and identically distributed random variables with values in the interval $[0, 1]$ and mean μ. Intuitively, Z_t is the outcome of experiment t.

Stopping-Rule Algorithm [9]
Input: $0 < \epsilon \leq 2$, $\delta > 0$

Initialize $t \leftarrow 0$, $X \leftarrow 0$, and $S^* \leftarrow 4\ln(2/\delta)(1+\epsilon)/\epsilon^2$

While $X < S^*$ do

$t \leftarrow t+1$

Generate random variable Z_t

$X \leftarrow X + Z_t$

Let $T^* \leftarrow t$ to be the total number of experiments

Output: S^*/T^*

The Stopping-Rule Theorem proves that the output of the stopping-rule algorithm approximates the mean μ within relative error ϵ with probability at least $1-\delta$. In addition, this theorem gives an upper bound on the expected number of experiments before the algorithm halts.

STOPPING-RULE THEOREM [9]:

1. $\mathbf{Pr}[\mu(1-\epsilon) \leq S^*/T^* \leq \mu(1+\epsilon)] > 1-\delta$.

2. $\mathbf{E}[T^*] \leq 4\ln(2/\delta)(1+\epsilon)/(\mu\epsilon^2)$.

A.2: Pseudorandom Generator

Nisan [24] gives the following construction of the $s \times l$ matrices $\mathcal{A}_{s,l}$ of Theorem 3. Let p denote a prime number of size approximately $(\log s)^{d+3}$, and let $l = p^2$. Let $t = \log s$, and choose s distinct vectors $\mathbf{b}_1, ..., \mathbf{b}_s$, of dimension t from the space $\{0, ..., p-1\}^t$. Each vector $\mathbf{b}_i$ defines the polynomial $f_i(x) = b_i^0 + b_i^1 x + \cdots + b_i^t x^t$, where b_i^j denotes the jth coordinate of $\mathbf{b}_i$. From each polynomial f_i, we construct a set $S_i = \{kp + (f_i(k) \mathrm{mod} p) | k = 0, ..., p-1\}$. The sets S_i are subsets of $\{1, ..., l\}$ that define the matrix $\mathcal{A}_{s,l}$. For $0 \leq i \leq s-1$ and $0 \leq j \leq l-1$, the ijth element a_{ij} of $\mathcal{A}_{s,l}$ is

$$a_{ij} = \begin{cases} 1, & \text{if } j \in S_i; \\ 0, & \text{otherwise.} \end{cases}$$

A.3: Discretization Error

The function `Generate_instantiation` z `from` $u \in \Phi$ in Figure 2 takes as input a string u of length nm chosen uniformly from all nm-bit strings. The function `Generate_instantiation` in Figure 1 takes as input n real numbers u, each chosen uniformly from the interval $[0, 1]$. We determine how the length m of the nm-bit string u affects the error that we make in computing an inference probability $\mathbf{Pr}[\mathcal{W} = w]$ when we use `Generate_instantiation` z `from` $u \in \Phi$, instead of `Generate_instantiation`, to generate instantiations of $\mathcal{Z}$.

As before, let σ denote the LVB of belief network on n nodes. Let $\mathbf{Pr}[\mathcal{W} = w]$ denote any inference probability where $\mathcal{W}$ contains k nodes, and let Ω denote the space of all 2^{nm} bit strings of length nm. We let $\mathbf{E}_\Omega \zeta$ denote the mean of the random variable $\zeta(z, w)$ evaluated on the 2^{nm} instantiations $\mathcal{Z} = z$ generated from the different nm-bit strings.

Lemma 8 *If $m = \log(2n\sigma/\epsilon))$, then $(\prod_{i=1}^{k} u_i)\mathbf{E}_\Omega \zeta$ approximates $\mathbf{Pr}[\mathcal{W} = w]$ within relative error ϵ.*

PROOF Let Δ denote the set of 2^{n-k} instantiations of $\mathcal{Z}$. Observe that

$$\mathbf{Pr}[\mathcal{W} = w] / \prod_{i=1}^{n-k} \mathrm{u}_i = \sum_{z \in \Delta} \rho(z, w)\zeta(z, w).$$

Let $\Omega_z \subseteq \Omega$ denote the subset of nm strings such that, if $u \in \Omega_z$, then the derandomized algorithm outputs the instantiation $\mathcal{Z} = z$. Thus, $\Omega = \cup_{z \in \Delta} \Omega_z$, and

$$\mathbf{E}_\Omega \zeta = \frac{1}{2^{nm}} \sum_{z \in \Delta} |\Omega_z| \zeta(z, w).$$

To complete the proof, we must show that $|\Omega_z|/2^{nm}$ approximates $\rho(z, w)$ within relative error ϵ when $m = \log(2n\sigma/\epsilon)$.

Recall that $\rho(z, w) = \prod_{i=1}^{n-k} \mathbf{Pr}[Z_i | \pi(Z_i)]|_{\mathcal{Z}=z, \mathcal{W}=w}$. For $i \leq n - k$, let $\theta_i = \mathbf{Pr}[Z_i | \pi(Z_i)]|_{\mathcal{Z}=z, \mathcal{W}=w}$; for $i \geq n - k + 1$, let $\theta_i = 1$. Let $U_i = \lfloor 2^m \theta_i \rfloor$. From the design of the derandomized algorithm, we easily verify that $|\Omega_z| = \prod_{i=1}^{n} U_i$. For all i,

$1/\sigma \leq \theta_i$, each $U_i/2^m$ approximates θ_i within relative error $\sigma/2^m$, and thus $|\Omega_z|/2^{nm}$ approximates $\rho(z,w) = \prod_{i=1}^{n} \theta_i$ within relative error $2n\sigma/2^m$. When $m = \log(2n\sigma/\epsilon)$, this error is ϵ. □

Acknowledgments

We thank the anonymous referees for the many insightful and constructive comments that improved this presentation substantially. We also thank Lyn Dupré for her thorough review of the manuscript. Our work was supported by the National Science Foundation under grants IRI-93-11950 and CCR-93-04722, by the Stanford University CAMIS project under grant IP41LM05305 from the National Library of Medicine of the National Institutes of Health, and by Israeli—U.S. NSF Binational Science Foundation grant No. 92-00226.

References

[1] Special issue on probability forecasting. *International Journal of Forecasting*, 11, 1995.

[2] Special issue on real-world applications of uncertain reasoning. *Communications of the ACM*, 38, 1995.

[3] R. Chavez and G. Cooper. An empirical evaluation of a randomized algorithm for probabilistic inference. In M. Henrion, R. Shachter, L. Kanal, and J. Lemmer, editors, *Uncertainty in Artificial Intelligence* **5**, pages 191–207. Elsevier, Amsterdam, The Netherlands, 1990.

[4] R. Chavez and G. Cooper. A randomized approximation algorithm for probabilistic inference on Bayesian belief networks. *Networks*, 20:661–685, 1990.

[5] G. Cooper. NESTOR: *A computer-based medical diganostic aid that integrates causal and probabilistic knowledge.* PhD thesis, Medical Computer Science Group, Stanford University, Stanford, CA, 1984.

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

[7] P. Dagum and R.M. Chavez. Approximating probabilistic inference in Bayesian belief networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:246–255, 1993.

[8] P. Dagum and E. Horvitz. A Bayesian analysis of simulation algorithms for inference in belief networks. *Networks*, 23:499–516, 1993.

[9] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for Monte Carlo estimation. In *Proceedings of the Thirtysixth IEEE Symposium on Foundations of Computer Science*, pages 142–149, Milwaukee, Wisconson, 1995.

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

[11] B. D'Ambrosio. Incremental probabilistic inference. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 301–308, Washington, DC, July 1993. Association for Uncertainty in Artificial Intelligence.

[12] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Velickovic. Approximations of general independent distributions. In *Proceedings of the 24th IEEE Annual Symposium on Theory of Commputing*, 1992.

[13] R. Fung and K.-C. Chang. Weighing and integrating evidence for stochastic simulation in Bayesian networks. In M. Henrion, R. Shachter, L. Kanal, and J. Lemmer, editors, *Uncertainty in Artificial Intelligence* **5**, pages 209–219. Elsevier, Amsterdam, The Netherlands, 1990.

[14] 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, Seatle, Washington, 1994. American Association for Artificial Intelligence.

[15] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. Lemmer and L. Kanal, editors, *Uncertainty in Artificial Intelligence* **2**, pages 149–163. North-Holland, Amsterdam, The Netherlands, 1988.

[16] M. Henrion. Search-based methods to bound diagnostic probabilities in very large belief nets. In *Proceedings of the Seventh Workshop on Uncertainty in Artificial Intelligence*, University of Los Angeles, Los Angeles, California, July 1991.

[17] E. Horvitz, H. J. Suermondt, and G. F. Cooper. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Proceedings of the 1989 Workshop on Uncertainty in Artificial Intelligence*, pages 182–193, Windsor, Ontario, July 1989.

[18] K. Huang and M. Henrion. Efficient search-based inference for noisy-OR belief networks: TopEpsilon. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 325–331, Portland, Oregon, 1996. American Association for Artificial Intelligence.

[19] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.

[20] R. Karp, M. Luby, and N. Madras. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10:429–448, 1989.

[21] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society* **B**, 50, 1988.

[22] M. Luby, B. Velickovic, and A. Wigderson. Deterministic approximate counting of depth-2 circuits. In *Proceedings of the Second Israeli Symposium on Theory of Commputing and Systems*, 1993.

[23] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[24] N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11:63–70, 1991.

[25] J. Pearl. Addendum: Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 33:131, 1987.

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

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

[28] Y. Peng and J.A. Reggia. A probabilistic causal model for diagnostic problem solving—Part 1: Integrating symbolic causal inference with numeric probabilistic inference. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-17(2):146–162, 1987.

[29] Y. Peng and J.A. Reggia. A probabilistic causal model for diagnostic problem solving—Part 2: Diagnostic strategy. *IEEE Trans. on Systems, Man, and Cybernetics: Special issue for diagnosis*, SMC-17(3):395–406, 1987.

[30] D. Poole. Average-case analysis of a search algorithm for estimating prior and posterior probabilities in Bayesian networks with extreme probabilities. In *Proceedings of the Thirteenth IJCAI*, pages 606–612, 1993.

[31] M. Pradhan and P. Dagum. Optimal Monte Carlo estimation of belief network inference. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial*

Intelligence, pages 446–453, Portland, Oregon, 1996. American Association for Artificial Intelligence.

[32] E. Santos and S. Shimony. Belief updating by enumeratiing high-probability independence-based assignments. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 506–513, Seattle, Washington, 1994.

[33] R. Shachter and M. Peot. Simulation approaches to general probabilistic inference on belief networks. In M. Henrion, R. Shachter, L. Kanal, and J. Lemmer, editors, *Uncertainty in Artificial Intelligence* **5**, pages 221–231. Elsevier, Amsterdam, The Netherlands, 1990.

[34] S. E. Shimony and E. Charniak. A new algorithm for finding MAP assignments to belief networks. In *Proceedings of Sixth Conference on Uncertainty in Artificial Intelligence*, pages 98–103, Cambridge, Massachusetts, 1990.

[35] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE Annual Symposium on the Foundations of Computer Science*, pages 1–10, 1985.