Approximate Inference in Bayesian Networks: Parameterized Complexity Results

Johan Kwisthout[a]

[a] *Radboud University, Donders Institute for Brain, Cognition and Behaviour, Montessorilaan 3, 6525HR Nijmegen, The Netherlands*

Abstract

Computing posterior and marginal probabilities constitutes the backbone of almost all inferences in Bayesian networks. These computations are known to be intractable in general, both to compute exactly and to approximate (e.g., by sampling algorithms). While it is well known under what constraints *exact* computation can be rendered tractable (viz., bounding tree-width of the moralized network and bounding the cardinality of the variables) it is less known under what constraints *approximate* Bayesian inference can be tractable. Here, we extend the existing formal framework of *fixed-error randomized tractability* (a randomized analogue of fixed-parameter tractability), and use it to address this problem, both by re-interpreting known results from the literature and by providing some additional new results, including results on fixed parameter tractable de-randomization of approximate inference.

Keywords: Approximate Inference, Sampling, Bayesian Networks, Parameterized Complexity, Stochastic Algorithms, De-randomization.

1. Introduction

Computing posterior and marginal probabilities constitutes the backbone of almost all inferences in Bayesian networks. These computations are known to be intractable in general [6]; moreover, it is known that *approximating* these computations is also intractable. To be precise, deterministic approximation is proven to be NP-hard by itself [8]; tractable *randomized* approximation is also ruled out unless $\mathsf{NP} \subseteq \mathsf{BPP}$. To render exact computation tractable, bounding the tree-width of the moralized network is both necessary (under the assumption of the Exponential Time Hypothesis) [23] and (with bounded cardinality) sufficient [24]. For *approximate* inference, the picture is less clear, in part because there are multiple approximation strategies that all have different properties and characteristics. First of all, it matters whether we approximate marginal, respectively conditional probabilities. The approximation error can be measured

Email address: `j.kwisthout@donders.ru.nl` (Johan Kwisthout)

either absolutely (also called additive approximation), i.e., independent of the probability that is to be approximated, or relative (also called multiplicative approximation) to this probability. Finally, the approximation algorithm can be deterministic (always guaranteeing a bound on the error) or randomized (guaranteeing a bounded error with high probability). In this broad array there are a few (somewhat isolated) tractability results [7–9, 14, 29, 30], but an overview of what can and cannot render approximate inference tractable is still lacking.

In this paper we extend and apply *fixed-error randomized tractability analysis* [21], a recent randomized analogue of parameterized complexity analysis [10], to systematically address this issue. We consider both absolute and relative approximation, using both randomized and deterministic algorithms, for the approximation of both marginal and conditional probabilities. We re-interpret old results and provide new results in terms of fixed-parameter or fixed-error tractability and intractability. In addition to identifying a number of corollaries from known results, some particular new contributions in this paper are derandomization results of randomized approximations for fixed degree networks.

The remainder of this paper is structured as follows. After introducing the necessary preliminaries on Bayesian networks, approximation strategies, and parameterized computational complexity in Section 2, we introduce and extend fixed-error randomized tractability analysis in Section 3. We give an overview of results from the literature in Section 4.1 and some new results in Section 4.2. The paper is concluded in Section 5.

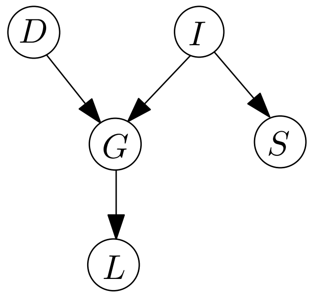
2. Preliminaries

In this section we introduce notation and provide for some preliminaries and our notational conventions in Bayesian networks, approximation algorithms, and complexity theory.

2.1. Bayesian networks

A (discrete) Bayesian network $\mathcal{B}$ is a graphical structure that models a set of discrete random variables, a joint probability distribution over these variables, and the conditional independences in this distribution [27]. $\mathcal{B}$ includes a directed acyclic graph $\mathbf{G}_{\mathcal{B}} = (\mathbf{V}, \mathbf{A})$, modeling the variables and conditional independences in the network, and a set of parameter probabilities Pr in the form of conditional probability tables (CPTs), capturing the strengths of the relationships between the variables. The network thus describes a joint probability distribution $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(V_i \mid \pi(V_i))$ over its variables, where $\pi(V_i)$ denotes the parents of V_i in $\mathbf{G}_{\mathcal{B}}$. We define the *size* $|\mathcal{B}|$ of the network to be the number of bits needed to represent both $\mathbf{G}_{\mathcal{B}}$ and Pr. Our notational convention is to use upper case letters to denote individual nodes in the network, upper case bold letters to denote sets of nodes, lower case letters to denote value assignments to nodes, and lower case bold letters to denote joint value assignments to sets of nodes. The set of values v_i that a variable V can take is denoted as $\Omega(V)$.

Figure 1 presents the running example we will use in this paper, the *Student network* introduced in [18]. Here, we model five random variables: The difficulty



$\Pr(d)$	$= 0.4$
$\Pr(\neg d)$	$= 0.6$
$\Pr(s_h \mid i)$	$= 0.8$
$\Pr(s_l \mid i)$	$= 0.2$
$\Pr(s_h \mid \neg i)$	$= 0.05$
$\Pr(s_l \mid \neg i)$	$= 0.95$
$\Pr(l \mid g_h)$	$= 0.9$
$\Pr(l \mid g_m)$	$= 0.6$
$\Pr(l \mid g_l)$	$= 0.01$
$\Pr(\neg l \mid g_h)$	$= 0.1$
$\Pr(\neg l \mid g_m)$	$= 0.4$
$\Pr(\neg l \mid g_l)$	$= 0.99$

$\Pr(i)$	$= 0.3$
$\Pr(\neg i)$	$= 0.7$
$\Pr(g_h \mid i, d)$	$= 0.5$
$\Pr(g_m \mid i, d)$	$= 0.3$
$\Pr(g_l \mid i, d)$	$= 0.2$
$\Pr(g_h \mid i, \neg d)$	$= 0.9$
$\Pr(g_m \mid i, \neg d)$	$= 0.08$
$\Pr(g_l \mid i, \neg d)$	$= 0.02$
$\Pr(g_h \mid \neg i, d)$	$= 0.05$
$\Pr(g_m \mid \neg i, d)$	$= 0.25$
$\Pr(g_l \mid \neg i, d)$	$= 0.7$
$\Pr(g_h \mid \neg i, \neg d)$	$= 0.3$
$\Pr(g_m \mid \neg i, \neg d)$	$= 0.4$
$\Pr(g_l \mid \neg i, \neg d)$	$= 0.3$

Figure 1: The *Student* network, adapted from Figure 3.3 in [18].

of a course D, with values difficult (d) and easy ($\neg d$); the intelligence of the student I, with values intelligent (i) and unintelligent ($\neg i$); the grade G of the course, where g_h, g_m, and g_l represent high, middle, and low grades, respectively; the SAT score S of the student (s_h = low, s_l = high); and finally whether one is willing to write a strong reference letter ($L = l$ if yes, $L = \neg l$ if no).

In the context of this paper we are particularly interested in the computation of marginal and conditional probabilities from a Bayesian network, defined as computational problems as follows:

MProb
Input: A Bayesian network $\mathcal{B}$ with designated subset of variables $\mathbf{H}$ and a corresponding joint value assignment $\mathbf{h}$ to $\mathbf{H}$.
Output: $\Pr(\mathbf{h})$.

CProb
Input: A Bayesian network $\mathcal{B}$ with designated non-overlapping subsets of variables $\mathbf{H}$ and $\mathbf{E}$ and corresponding joint value assignments $\mathbf{h}$ to $\mathbf{H}$ and $\mathbf{e}$ to $\mathbf{E}$.
Output: $\Pr(\mathbf{h} \mid \mathbf{e})$.

For example, in the student network it can be computed that $\Pr(S = s_h) = 0.275$ (i.e., the prior probability of scoring high at a SAT test is 0.275) and that $\Pr(L = l \mid I = \neg i) = 0.389$ (i.e., the probability that you write a strong recommendation letter for an unintelligent student is 0.389). It is well known that both MProb and CProb are intractable (NP-hard) problems to compute exactly, that is, there is strong evidence that there cannot exist a worst-case polynomial time algorithm for either of them.

2.2. Approximation

For a particular instance $(\mathcal{B}, \mathbf{H}, \mathbf{h})$ of MProb, respectively $(\mathcal{B}, \mathbf{H}, \mathbf{h}, \mathbf{E}, \mathbf{e})$ of CProb, an *approximate* result is a solution to this instance that is within guaranteed bounds of the optimal solution. We use ϵ to denote such (two-sided) bounds, and define absolute approximation and relative approximation[1] of MProb and CProb as follows:

aa-MProb
Input: As in MProb, in addition, error bound ϵ.
Output: q such that $\Pr(\mathbf{h}) - \epsilon < q < \Pr(\mathbf{h}) + \epsilon$.

ra-MProb
Input: As in MProb, in addition, error bound ϵ.
Output: q such that $\Pr(\mathbf{h}) \times (1 - \epsilon) < q < \Pr(\mathbf{h}) \times (1 + \epsilon)$.

For example in the student network, for $\epsilon = 0.05$, $\mathbf{H} = \{S\}$, and $\mathbf{h} = \{s_h\}$ an algorithm that returns $q = 0.273$ gives both an absolute and a relative approximation of $\Pr(S = s_h)$, since $0.275 - 0.05 < 0.273 < 0.275 + 0.05$ and $0.275 \times (1 - 0.05) \approx 0.262 < 0.273 < 0.275 \times (1 + 0.05) \approx 0.289$. We define aa-CProb and ra-CProb correspondingly for the conditional case.

aa-CProb
Input: As in CProb, in addition, error bound ϵ.
Output: q such that $\Pr(\mathbf{h} \mid \mathbf{e}) - \epsilon < q < \Pr(\mathbf{h} \mid \mathbf{e}) + \epsilon$.

ra-CProb
Input: As in CProb, in addition, error bound ϵ.
Output: q such that $\Pr(\mathbf{h} \mid \mathbf{e}) \times (1 - \epsilon) < q < \Pr(\mathbf{h} \mid \mathbf{e}) \times (1 + \epsilon)$.

The results in the literature typically describe approximation algorithms for the *function* problems introduced above. Since we are interested in analyzing the computational complexity of these approximation strategies under specific input restrictions, we define *decision variants*[2] as follows:

[1] Here, we use the definition by [16] for the relative lower bound, rather than the slightly more constraining definition by [3] that demanded that $\frac{\Pr(\mathbf{h})}{1+\epsilon} < q$. Since the difference between both bounds depends on ϵ only (viz., a factor $1 - \epsilon^2$) we will liberally mix results that depend on either definition.

[2] Note that intractability of these decision problems implies intractability of the functional variants. In practical applications of these problems we are mostly interested in the functional variants; that is why we do not consider r or derivations of r as a parameter of interest in our analyses.

aa-MProb
Input: As in MProb, in addition, error bound ϵ, rational number r.
Question: Is $\Pr(\mathbf{h}) \pm \epsilon > r$?

aa-CProb
Input: As in CProb, in addition, error bound ϵ, rational number r.
Question: Is $\Pr(\mathbf{h} \mid \mathbf{e}) \pm \epsilon > r$?

ra-MProb
Input: As in MProb, in addition, error bound ϵ, rational number r.
Question: Is $\Pr(\mathbf{h}) \times (1 \pm \epsilon) > r$?

ra-CProb
Input: As in CProb, in addition, error bound ϵ, rational number r.
Question: Is $\Pr(\mathbf{h} \mid \mathbf{e}) \times (1 \pm \epsilon) > r$?

Note that, by design, these decision problems include a range of inputs for which either answer is acceptable. For example in the decision variant of the aa-MProb problem we require that an algorithm for this problem answers *yes* if $\Pr(\mathbf{h}) + \epsilon > r$, we require that it answers *no* if $\Pr(\mathbf{h}) - \epsilon \leq r$, and we accept both *yes* and *no* answers if $r \in \langle \Pr(\mathbf{h}) - \epsilon, \Pr(\mathbf{h}) + \epsilon]$. We will denote such decision variants as *approximate decision problems*[3]. An algorithm $\mathcal{A}_f$ for solving a functional variant of an approximation problem can easily be turned into an algorithm $\mathcal{A}_d$ for deciding the corresponding approximate decision problem by comparing the computed approximate value q with the threshold r, answering *yes* if $q > r$ and *no* if $q \leq r$.

2.2.1. Stochastic algorithms

In addition to deterministic approximation algorithms, that are guaranteed to *always* give a solution that is within the error bounds indicated by ϵ, we also consider *stochastic* or randomized approximation algorithms, where this constraint is relaxed. In a stochastic approximation we accept that a solution is *not* within the error bounds with probability $0 < \delta < 1/2$, viz., the approximate result is within the error bounds ϵ with probability at least $1 - \delta > 1/2$. This is formalized as follows for the marginalization problem; the conditional cases are defined analogously.

Definition 2.1 (δ-stochastic absolute approximation)**.** Let $\mathcal{A}_d$ be an algorithm that decides aa-MProb, and let $0 < \delta < 1/2$. We write that $\mathcal{A}_d$ *decides incorrectly* if $\mathcal{A}_d(\mathcal{B}, \mathbf{H}, \mathbf{h}, r, \epsilon) = no$ while $\Pr(\mathbf{h}) + \epsilon > r$, or if $\mathcal{A}_d(\mathcal{B}, \mathbf{H}, \mathbf{h}, r, \epsilon) = yes$ while $\Pr(\mathbf{h}) - \epsilon \leq r$. If $\Pr(\text{`}\mathcal{A}_d$ decides incorrectly') $< \delta$, then $\mathcal{A}_d$ is called a δ-stochastic algorithm for aa-MProb.

[3]A similar construction is used in the literature (e.g., [13]) to define so-called gap-problems. Also related is the notion of a *promise* problem [12] where we promise that the input does not contain instances in the range $\langle \Pr(\mathbf{h}) - \epsilon, \Pr(\mathbf{h}) + \epsilon]$; here the algorithm is allowed to answer anything (or even run forever) if this promise is violated.

Definition 2.2 (δ-stochastic relative approximation)**.** Let $\mathcal{A}_d$ be an algorithm that decides ra-MProb, and let $0 < \delta < 1/2$. We write that $\mathcal{A}_d$ *decides incorrectly* if $\mathcal{A}_d(\mathcal{B}, \mathbf{H}, \mathbf{h}, r, \epsilon) = no$ while $\Pr(\mathbf{h}) \times (1+\epsilon) > r$, or if $\mathcal{A}_d(\mathcal{B}, \mathbf{H}, \mathbf{h}, r, \epsilon) = yes$ while $\Pr(\mathbf{h}) \times (1-\epsilon) \leq r$. If Pr('$\mathcal{A}_d$ decides incorrectly') $< \delta$, then $\mathcal{A}_d$ is called a δ-stochastic algorithm for ra-MProb.

In general, $\mathcal{A}_d$ will use a sampling approach to guarantee these probabilities of answering correctly. The *Hoeffding* (for absolute approximations) and *multiplicative Chernoff* (for relative approximations) bounds can be used to compute δ relative to ϵ, the number of samples M, and (in the latter case) the posterior probability $\mathbf{h}$. The Hoeffding bound is defined as $\delta \leq 2e^{-2M\epsilon^2}$ for a particular number of samples M and error ϵ. In order to guarantee that δ is small enough we can solve for M, which gives us $M \geq \frac{\ln(2/\delta)}{2\epsilon^2}$. Let us assume in our running example that $\mathcal{A}_d$ is a stochastic algorithm that absolutely approximates a decision on $\Pr(S = s_h) > r$ for some r and $\epsilon = 0.05$ by sampling, then the Hoeffding bound learns us that 1160 samples are needed for $\mathcal{A}_d$ to guarantee that it decides incorrectly with probability below $\delta = 0.01$.

Likewise, the multiplicative Chernoff bound is defined (for ra-MProb and ra-CProb) as $\delta \leq 2e^{\frac{-M\Pr(\mathbf{h} \mid \mathbf{e})\epsilon^2}{3}}$. Here, solving for M gives us $M \geq 3\frac{\ln(2/\delta)}{\Pr(\mathbf{h})\epsilon^2}$. Note that for relative approximation M depends on the probability to be approximated. It can be computed that a stochastic algorithm, that relative approximates $\Pr(S = s_h) > r$ with error $\epsilon = 0.05$ and probability of answering incorrectly at most $\delta = 0.01$, needs at least 23120 samples.

For any given $\mathcal{A}_d$ that has a probability δ of answering incorrectly, we can repeat $\mathcal{A}_d$ M' times and take a majority decision on the outcomes of the individual runs. The number of repetitions (or samples of a run of $\mathcal{A}_d$) M' needed by the thus constructed algorithm $\mathcal{A}'_d$ to obtain a probability of answering incorrectly of δ' can be computed by the sampling variant of the Chernoff bound. In particular, $M' \geq \frac{\ln(1/\sqrt{\delta'})}{(1/2-\delta)^2}$. Note that if δ is polynomially bounded away from $1/2$ (i.e., $\delta < 1/2 - 1/|x|^c$, where $|x|$ denotes the input size of the problem and c is a constant), and $\mathcal{A}_d$ runs in polynomial time, then we can decrease δ' arbitrarily close to zero at the cost of only a polynomial increase in the running time of $\mathcal{A}'_d$. Hence, in the remainder, we will not consider δ' itself as a parameter of interest, but rather observe the effect of *other* parameters on δ. Typically, as all algorithms we investigate are based on a sampling approach, we will focus on characterizing the number of samples needed to ensure a particular upper bound on δ.

2.3. Complexity theory

We assume that the reader is familiar with basic notions from complexity theory, such as intractability proofs, the computational complexity classes P and NP, and polynomial-time reductions. In this section we shortly review some additional concepts that we use throughout the paper, particularly the complexity classes PP and BPP and some basic principles from parameterized complexity theory. We refer the interested reader to textbooks such as [2] and [10] for a more elaborate treatment of these topics.

The complexity classes PP and BPP are defined as classes of decision problems that are decidable by a probabilistic Turing machine (i.e., a Turing machine that makes stochastic state transitions) in polynomial time with a particular (two-sided) probability of error; the difference between these two classes is in the bound on the error probability. We formally define them as follows:

Definition 2.3 (PP)**.** Let Π be a decision problem. We have that $\Pi \in \mathsf{PP}$ if and only if there exists a probabilistic Turing machine $\mathcal{M}$ that halts after time, polynomial in the size of the input x, with the following acceptance criteria. $\mathcal{M}$ accepts *Yes*-instances of Π with probability strictly larger than $1/2$; *No*-instances are accepted with probability at most $1/2$.

Definition 2.4 (BPP)**.** Let Π be a decision problem. We have that $\Pi \in \mathsf{BPP}$ if and only if there exists a probabilistic Turing machine $\mathcal{M}$ that halts after time, polynomial in the size of the input x, with the following acceptance criteria. $\mathcal{M}$ accepts *Yes*-instances of Π with probability strictly larger than $1/2 + 1/|x|^c$, where $|x|$ denotes the size of the instance and $c > 1$ is a constant; *No*-instances are accepted with probability at most $1/2 - 1/|x|^c$.

Note that the probability of acceptance of a *yes*-instance in PP may depend exponentially on the input size $|x|$ (i.e., the probability of acceptance may be $1/2 + 1/c^{|x|}$ for a constant $c > 1$). PP-complete problems, such as the problem of determining whether the *majority* of truth assignments to a Boolean formula ϕ satisfies ϕ, are considered to be intractable; indeed, it can be shown that $\mathsf{NP} \subseteq \mathsf{PP}$. In contrast, problems in BPP are considered to be tractable. Informally, a decision problem Π is in BPP if there exists an efficient randomized (Monte Carlo) algorithm that decides Π with high probability of correctness. Given that the error is polynomially bounded away from $1/2$, the probability of answering correctly can be boosted to be arbitrarily close to 1 while still requiring only polynomial time as a result of the Hoeffding bound. While obviously $\mathsf{BPP} \subseteq \mathsf{PP}$, the reverse is unlikely; in particular, it is conjectured that $\mathsf{BPP} = \mathsf{P}$ [5].

A related complexity class that is of similar power as PP, but based on *counting* solutions rather than on the probability of acceptance, is the class $\#\mathsf{P}$. This class is populated by function problems, rather than decision problems. A function $f(x)$ is in $\#\mathsf{P}$ if there is a non-deterministic Turing machine on which $f(x)$ computational paths accept an input x, or more informally, a counting problem ("how many solutions...") is in $\#\mathsf{P}$ if its corresponding decision problem ("does a solution exist...") is in NP [32]. The canonical complete problem for $\#\mathsf{P}$ is #SAT. It can be shown that $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$ [4], i.e., $\mathsf{PP} = \#\mathsf{P}$ under Turing reductions.

2.3.1. De-randomization

De-randomization of a stochastic algorithm turns a randomized algorithm into a deterministic algorithm by removing stochastic state transitions. In principle, (decision variants of) randomized algorithms can always be de-randomized by simulating all possible combinations of stochastic transitions and taking a majority vote. In particular if, using some sequence of random decisions, the probability of answering correctly is bigger than $^1/_2$, the majority of all thus simulated computation paths will arrive at this correct solution. If a randomized program takes n random bits in its computation, this will yield an $\mathcal{O}(2^n)$ deterministic algorithm; however, if a polynomial-time randomized algorithm can be shown to use at most $\mathcal{O}(\log |x|)$ *distinct* random bits (i.e., logarithmic in the input size $|x|$) than the de-randomization will yield a polynomial time deterministic (although not very efficient) algorithm. If this were possible for all polynomial-time randomized algorithms (i.e., deciding problems in BPP), then it would follow that $\mathsf{BPP} = \mathsf{P}$.

One can show that a $\log n$ bit random string can be amplified to a k-wise independent random n bit string in time $\mathcal{O}(n^k)$ [1, 25]. *k-wise independence* between $n > k$ bits is a weaker constraint than *mutual independence.* The following example, credited to Bernstein in [15, p. 120], gives an example of k-wise independence in joint probability distributions.

Example 1. *Let X, Y, and Z be three binary variables with the following semantics: X and Y denote the outcome of two (independent) coin flips ($1 =$ heads, 0 is tails); Z denotes the parity of the number of heads showing ($1 =$ odd, $0 =$ even). We have that* $\Pr(X = 0, Y = 0, Z = 0) = 0.25$, $\Pr(X = 0, Y = 1, Z = 1) = 0.25$, $\Pr(X = 1, Y = 0, Z = 1) = 0.25$, *and* $\Pr(X = 1, Y = 1, Z = 0) = 0.25$. *Thus, we have that* $\Pr(X = 1) = \Pr(Y = 1) = \Pr(Z = 1) = 0.5$. *By construction, X and Y are independent, as well as X and Z, respectively Y and Z. However, obviously it does not hold that* $\Pr(X, Y, Z) = \Pr(X) \times \Pr(Y) \times \Pr(Z)$. *Hence, X, Y, and Z are pairwise independent ($k = 2$), but they are not mutually independent.*

In general, showing that the random bits in the randomized algorithm can be constrained to be k-wise independent (rather than fully independent) is sufficient for effective de-randomization. The thus de-randomized stochastic algorithm has a running time which is exponential in k, viz., if the randomized algorithm ran in time $\mathcal{O}(|x|^c)$, the de-randomized algorithm runs in time $\mathcal{O}(|x|^{c^k})$.

2.3.2. Parameterized complexity

Sometimes problems are intractable (i.e., NP-hard) in general, but become tractable if some *parameters* of the problem can be assumed to be small. Informally, a decision problem is called fixed-parameter tractable [10, 11] with respect to parameter k (or a set of parameters $\{k_1, \ldots, k_m\}$) if it can be decided in time, exponential (or even worse) *only* in k and polynomial in the input size $|x|$, that is, in $\mathcal{O}(f(k) \cdot |x|^c)$ for a constant $c > 1$ and an arbitrary computable function f. In practice, this means that problem instances can be solved efficiently, even when the problem is NP-hard in general, if k is known to be small.

For a decision problem Π (formally defined as a subset of strings over $\{0, 1\}^*$), we define a parameterization κ as a mapping $\kappa : \{0, 1\}^* \to \mathbb{N}$. In the remainder, the mapping is often trivial and we will refer to the parameters themselves, implicitly referring to a suitable parameterization. In earlier work it was shown that this mapping can easily be expanded to include monotone rational parameters as well as integer parameters [20]; we will liberally mix integer and rational parameters throughout this paper. Given a parameterization, we define a parameterized problem $\{k\}$-Π as the pair (Π, κ), with $x \in \{0,1\}^*$ as instances of Π and $\kappa(x)$ as the corresponding parameters of these instances[4].

The complexity class FPT characterizes parameterized problems $\{k\}$-Π that are fixed-parameter tractable with respect to the parameter k. On the other hand, if Π remains NP-hard for all but finitely many values of the parameter k, then $\{k\}$-Π is para-NP-hard: bounding k does not render the problem tractable. Conceptually somewhat in between[5] these extremes one can place a problem $\{k\}$-Π that can be solved in time $\mathcal{O}(|x|^{f(k)})$; in this case, $\{k\}$-Π is in the class XP. A problem which is complete for this class is typically not considered to be tractable [11]; having the parameter in the exponent of the instance size quickly blows up the running time for increasing $|x|$. Below we give formal definitions of FPT, XP, and para-NP.

Definition 2.5 (FPT)**.** Let $\{k\}$-Π be a parameterized decision problem. We have that $\{k\}$-$\Pi \in \mathsf{FPT}$ if and only if there exists a deterministic Turing machine $\mathcal{M}$ that decides every instance x of Π in time $f(k) \times \mathcal{O}(|x|^c)$ for an arbitrary function f and constant $c > 1$.

Definition 2.6 (XP)**.** Let $\{k\}$-Π be a parameterized decision problem. We have that $\{k\}$-$\Pi \in \mathsf{XP}$ if and only if there exists a deterministic Turing machine $\mathcal{M}$ that decides every instance x of Π in time $\mathcal{O}(|x|^{f(k)})$ for an arbitrary function f.

Definition 2.7 (para-NP)**.** Let $\{k\}$-Π be a parameterized decision problem. We have that $\{k\}$-$\Pi \in$ para-NP if and only if there exists a non-deterministic Turing machine $\mathcal{M}$ that decides every instance x of Π in time $f(k) \times \mathcal{O}(|x|^c)$ for an arbitrary function f and constant $c > 1$.

XP and para-NP are known to contain complete problems, FPT is not suspected to contain complete problems. A final observation with respect to fixed-

[4]The informed reader might observe that we take sort of a middle stance between Flum & Grohe's [11] definition of a parameter (being the resultant of a *polynomial-time* computable mapping $\{0,1\}^* \to \mathbb{N}$) and Downey & Fellows' [10] definition of a parameter as a positive integer without requiring that this integer stems from such a polynomial-time mapping. In the remainder we use parameters (such as the posterior probability) which are considered to be natural properties of instances and which can be mapped onto a positive integer, but which are neither known to be *polynomial-time* computable from the input, nor are positive integers themselves. To paraphrase Downey & Fellows [10, p. 15, footnote 1], we believe that our definition is 'certainly appropriate for the practical applications' that we describe in this paper.

[5]Readers may notice the omission of the W-hierarchy in this treatment. We will not touch upon any W-hardness result in this paper. Furthermore, since the relation between XP and para-NP is still unknown, "in between" is not to be interpreted literally as a claim that $\mathsf{FPT} \subseteq \mathsf{XP} \subseteq$ para-NP.

parameter tractability that we want to make is that for each $k \subset k_1$, if k-$\Pi \in \mathsf{FPT}$, then k_1-$\Pi \in \mathsf{FPT}$, and for each $k_2 \subset k$, if k-Π is para-NP-hard, then k_2-Π is para-NP-hard.

3. Fixed-error randomized tractability analysis

A similar notion of parameterized tractability has been proposed for the randomized complexity class BPP [21]. A problem may need an exponential number of samples to be solved by a randomized algorithm (e.g., be PP-complete), but this number of samples may be exponential only in some parameters of the instance, while being polynomial in the input size. A parameterization of the probability of acceptance of a probabilistic Turing machine introduces the notion *fixed-error randomized tractable* for a problem with parameter k that can be decided by a probabilistic Turing Machine in polynomial time, where the probability of error depends polynomially on the input and exponential on k. The class FERT characterizes problems Π that can be efficiently computed with a randomized algorithm (i.e., in polynomial time, with error arbitrarily close to 0) if k is bounded. We formally define[6] FERT as follows:

Definition 3.1 (FERT)**.** Let Π be a decision problem and let k-Π be a parameterization of Π. We have that k-$\Pi \in \mathsf{FERT}$ if and only if there exists a probabilistic Turing machine $\mathcal{M}$ that halts after time, polynomial in the size of the input x, with the following acceptance criteria. $\mathcal{M}$ accepts *Yes*-instances of Π with probability at least $1/2 + \min(f(k), 1/|x|^c)$ for a constant c and arbitrary function $f : \mathbb{R} \to \langle 0, 1/2]$; *No*-instances are accepted with probability at most $1/2 - \min(f(k), 1/|x|^c)$.

Analogous to the parameterized complexity class XP, in addition we now also define its randomized variant XER. Here the probability of error exponentially depends on a function of k:

Definition 3.2 (XER)**.** Let Π be a decision problem and let k-Π be a parameterization of Π. We have that k-$\Pi \in \mathsf{XER}$ if and only if there exists a probabilistic Turing machine $\mathcal{M}$ that halts after time, polynomial in the size of the input x, with the following acceptance criteria. $\mathcal{M}$ accepts *Yes*-instances of Π with probability at least $1/2 + \min(1/|x|^{f(k)}, 1/|x|^c)$ for a constant c and arbitrary function f; *No*-instances are accepted with probability at most $1/2 - \min(1/|x|^{f(k)}, 1/|x|^c)$.

Finally, if Π is PP-hard for all but finitely values of k, then $\{k\}$-Π is para-PP-hard. A somewhat weaker result relies on the assumption that $\mathsf{BPP} \subsetneq \mathsf{NP}$: if Π is NP-hard for bounded k, then $\{k\}$-$\Pi \notin \mathsf{FERT}$, as the opposite would imply that $\mathsf{NP} \subseteq \mathsf{BPP}$. Figure 2 gives an overview of the inclusion relationships between the various complexity classes. Note that $\mathsf{FERT} \subseteq \mathsf{XER} \subseteq$ para-PP (as we demand that each problem in XER must be decidable in polynomial time on a probabilistic Turing machine), while the relationship between XP and para-NP is open.

[6] In [21] a slightly different, but equivalent, definition of FERT was introduced.

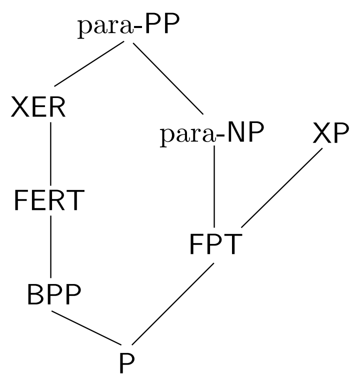


Figure 2: Inclusion properties for the complexity classes P, BPP, FERT, XER, FPT, XP, para-NP, and para-PP.

4. Results

For exact computations, it is well known that the *tree-width* of the network and the maximum *cardinality* of the variables are crucial parameters that allow for fixed-parameter tractable computations of both marginal and posterior probabilities. For approximate computations, we identify the parameters in Table 1. The *dependence value of the evidence* (D_e) is a measure of the cumulative strength of the dependencies in the network, given a particular joint value assignment to the evidence variables [7]. Under the assumption that there are no deterministic parameters in the network, we can define for any node $X_i \notin \mathbf{E}$ its dependence strength (given observed parents $\mathbf{p_e}$) $\lambda_i^{\mathbf{e}}$ as ${}^{u_i}/_{l_i}$, where l_i and u_i are the greatest, resp. smallest numbers such that

$$\forall_{x_i \in \Omega(X_i)} \forall_{\mathbf{p_u} \in \Omega(\pi(X_i) \setminus \mathbf{E})} l_i \leq \Pr(X_i = x_i \mid \mathbf{p_u}, \mathbf{p_e}) \leq u_i$$

The dependence value of the network (given evidence $\mathbf{e}$) is then given as $D_e = \prod_i \lambda_i^{\mathbf{e}}$. Note that this value is dependent on the evidence. The *local variance bound* (B) of a network is a measure of the representational expressiveness and the complexity of inference of the network [9]. It is defined similarly as the dependence value, but is not conditioned on the evidence, and a maximization rather than a product of the local values is computed. Let λ_i be ${}^{u_i}/_{l_i}$, where l_i and u_i are the greatest, resp. smallest numbers such that

$$\forall_{x_j \in \Omega(X_i)} \forall_{\mathbf{p} \in \Omega(\pi(X_i))} l_i \leq \Pr(X_i = x_i \mid \mathbf{p}) \leq u_i$$

The local variance bound of the network is then given as $B = \max_i(\lambda_i)$. Additional parameters that we will consider are the posterior probability $\Pr(\mathbf{h} \mid \mathbf{e})$ (or marginal probability $\Pr(\mathbf{h})$ if there is no evidence), the prior probability of the evidence $\Pr(\mathbf{e})$, the number of hypothesis variables $|\mathbf{H}|$ and the number of hypothesis *plus* the number of evidence variables $|\mathbf{H}\cup\mathbf{E}|$, the maximum indegree d, the maximum path length l, and the maximum connectivity (i.e., the maximal number of directed paths between any pair of nodes) c of the network. Finally, we will also parameterize the actual approximation error ϵ. In Table 1 we include the parameter values for the approximation of $\Pr(L = l \mid I = \neg i) = 0.389$ in the running example.

Parameter	Meaning	Running example
D_e	*dependence value* of the evidence	39501
B	*local variance bound* of the network	99
$\Pr(\mathbf{h} \mid \mathbf{e})$	posterior probability	0.389
$\|\mathbf{H}\|$	number of hypothesis variables	1
$\|\mathbf{H} \cup \mathbf{E}\|$	number of hypothesis or evidence variables	2
$\Pr(\mathbf{e})$	prior probability of the evidence	0.7
d	maximum indegree of the network	2
l	maximum path length of the network	2
c	maximum connectivity	1
ϵ	absolute or relative error	0.05

Table 1: Overview of parameters used in the results

4.1. Reinterpretation of known results

In this section we review the literature on approximation algorithms with respect to parameterized complexity results. In general, if there exists a (fixed-parameter or fixed-error) tractable *relative* approximation for MProb, respectively CProb, then there also exists a tractable *absolute* approximation; this is due to the probabilities p that are being approximated are between 0 and 1, and thus that if $p \times (1 - \epsilon) < r < p \times (1 + \epsilon)$, then $p - \epsilon < r < p + \epsilon$. The intractability of absolute approximations thus implies the intractability of relative approximations. The converse does not necessarily hold, in particular if p is very small; in a sense, relative approximations are 'harder' then absolute approximations.

For the reader's convenience we will briefly recap the straightforward reductions that show that Non-zero Inference is NP-hard and that Threshold Inference is PP-hard; for a more thorough treatment (including membership proofs for these classes) we refer the reader to [19]. From a Boolean formula ϕ with n variables we construct a Bayesian network $\mathcal{B}_\phi$ as follows. For every variable x_i in ϕ we add a binary, uniformly distributed root variable X_i to $\mathcal{B}_\phi$. For every operator o_j in ϕ we add a binary variable O_j to $\mathcal{B}_\phi$, whose parents are the variables in $\mathcal{B}_\phi$ that represent sub-formulas (or singleton variables) in ϕ that are bound by o_j, and whose CPT encodes the truth table of o_j. The variable corresponding to the top-level operator in ϕ is denoted as V_ϕ. In Figure 3 the thus constructed network is shown for $\phi = \neg(x_1 \vee x_2) \wedge \neg x_3$. Due to the construction of the network, we have that $\Pr(V_\phi = \mathit{true}) = {}^{\#_\phi}/_{2^n}$, where $\#_\phi$ is the number of satisfying truth assignments to $x_1, \ldots, x_n$.

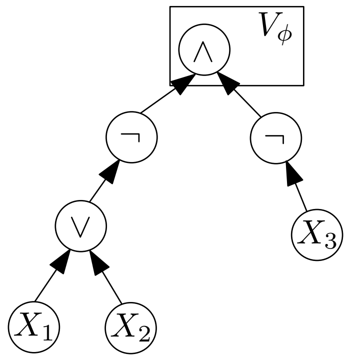


Figure 3: The Bayesian network $\mathcal{B}_\phi$ corresponding to $\phi = \neg(x_1 \vee x_2) \wedge \neg x_3$

Theorem 4.1. Non-zero Inference *is* NP*-hard.*

Proof. We reduce Satisfiability to Non-zero Inference. Let ϕ be an instance of Satisfiability and let $\mathcal{B}_\phi$ be the Bayesian network constructed from ϕ as discussed above. We have that $\Pr(V_\phi = \mathit{true}) > 0$ if and only if there is at least one satisfying truth assignment to ϕ. □

Theorem 4.2. Threshold Inference *is* PP*-hard.*

Proof. We reduce MajSat to Threshold Inference. Let ϕ be a MajSat-instance and let $\mathcal{B}_\phi$ be the Bayesian network constructed from ϕ as discussed above. We have that $\Pr(V_\phi = \mathit{true}) > 1/2$ if and only if the majority of truth assignments to ϕ satisfy ϕ. □

4.1.1. Approximation of marginal probabilities

For any fixed $\epsilon < 1/2^{n+1}$ it is easy to show that absolute approximate inference is PP-hard, as any approximation algorithm that is able to guarantee such a bound is able to solve MajSat simply by rounding the answer to the nearest $1/2^n$; this result even holds for bounded d as the PP-hardness proof is constrained to networks with indegree 2 [19].

Corollary 1 ([19])**.** $\{d\}$-aa-MProb is PP-hard.

For $\epsilon \geq 1/n^c$ we can efficiently approximate the marginal inference problem absolutely by simple forward sampling [14]; this is a corollary of the Hoeffding bound presented earlier. Thus, the number of samples needed to approximate aa-MProb by a randomized algorithm depends (only) on ϵ. To formally prove membership of $\{\epsilon\}$-aa-MProb in the parameterized complexity class FERT, we show the existence of a probabilistic Turing machine $\mathcal{M}$ that accepts *Yes*-instances of aa-MProb in polynomial time with probability strictly more than

${}^1\!/_2 + \min(f(\epsilon), {}^1\!/_{|x|^c})$ and accepts *No*-instances in polynomial time with probability at most ${}^1\!/_2 - \min(f(\epsilon), {}^1\!/_{|x|^c})$.

Let $\mathcal{M}$ be a probabilistic Turing machine that on input $(\mathcal{B}, \mathbf{H}, \mathbf{h}, r, \epsilon)$ computes the joint distribution $\Pr(V_1, \ldots, V_n)$ over the variables in $\mathcal{B}$ by forward sampling (cfm. the proof of Lemma 2.7 in [19]). Each computation path of $\mathcal{M}$ then corresponds with a joint value assignment $(v_1, \ldots, v_n)$ to $(V_1, \ldots, V_n)$; the probability of arriving in this path is exactly $\Pr(v_1, \ldots, v_n)$. We accept the input with probability $(1 - {}^r\!/_2)$ if this joint value assignment is consistent with $\mathbf{h}$ and accept the input with probability $({}^1\!/_2 - {}^r\!/_2)$ if it is inconsistent. Thus, the probability of acceptance is $\Pr(\mathbf{h})(1 - {}^r\!/_2) + (1 - \Pr(\mathbf{h}))({}^1\!/_2 - {}^r\!/_2) = {}^1\!/_2 + {}^{\Pr(\mathbf{h})}\!/_2 - {}^r\!/_2$. Thus, the probability of accepting the input when $\Pr(\mathbf{h}) > r + \epsilon$ is strictly more than ${}^1\!/_2 + {}^\epsilon\!/_2 = {}^1\!/_2 + \min(f(\epsilon), {}^1\!/_{|x|^c})$ and the probability of accepting the input when $\Pr(\mathbf{h}) \leq r - \epsilon$ is at most ${}^1\!/_2 - {}^\epsilon\!/_2 = {}^1\!/_2 - \min(f(\epsilon), {}^1\!/_{|x|^c})$.

Result 2 (based on [14])**.** $\{\epsilon\}$-AA-MProb $\in$ FERT.

In contrast to Result 2, for relative approximation it is impossible to approximate RA-MProb in polynomial time in general with *any* bound ϵ; this is a direct consequence of the proof outlined above. It is easy to show that the decision variant of RA-MProb introduced in the previous section is NP-hard for every value $\epsilon < 1$, as in that case $\Pr(\mathbf{h}) \times (1 \pm \epsilon) > 0$ if and only if $\Pr(\mathbf{h}) > 0$.

Result 3 (based on [6])**.** $\{\epsilon\}$-RA-MProb is para-NP-hard.

Corollary 4. $\{\epsilon\}$-RA-MProb $\notin$ FERT.

We *can* use the Chernoff bound to show that the number of samples needed to approximate RA-MProb depends on both ϵ and on $\Pr(\mathbf{h})$. The proof of membership in FERT is similar to the proof of Result 2; now, the probability of accepting the input when $\Pr(\mathbf{h}) \times (1 + \epsilon) > r$ is strictly more than ${}^1\!/_2 + {}^{(1+\epsilon)\Pr(\mathbf{h})}\!/_2 - {}^{\Pr(\mathbf{h})}\!/_2 = {}^1\!/_2 + {}^{\epsilon\Pr(\mathbf{h})}\!/_2 = {}^1\!/_2 + \min(f(\epsilon, \Pr(\mathbf{h})), {}^1\!/_{|x|^c})$ and the probability of accepting the input when $\Pr(\mathbf{h}) \times (1 - \epsilon) \leq r$ is at most ${}^1\!/_2 + {}^{(1-\epsilon)\Pr(\mathbf{h})}\!/_2 - {}^{\Pr(\mathbf{h})}\!/_2 = {}^1\!/_2 - {}^{\epsilon\Pr(\mathbf{h})}\!/_2 = {}^1\!/_2 - \min(f(\epsilon, \Pr(\mathbf{h})), {}^1\!/_{|x|^c})$.

Result 5 (based on [14])**.** $\{\Pr(\mathbf{h}), \epsilon\}$-RA-MProb $\in$ FERT.

In the remainder, for brevity we will omit such formal proofs of membership in FERT and assume that k-$\Pi \in$ FERT if there exists a randomized algorithm deciding Π, where the number of samples needed is an arbitrary function of k and at most polynomial in the input size[7]. We will also assume that k-$\Pi \in$ XER if the number of samples is of the form $\mathcal{O}(|x|^{f(k)})$, i.e., exponential in (some function of) k.

4.1.2. Approximation of conditional probabilities

The picture becomes more diverse when we consider computing posterior probabilities conditioned on evidence, rather than marginal probabilities. There

[7]A formal proof that this assumption is valid (when the non-parameterized problem $\Pi \in$ PP) will be presented in a forthcoming paper (Nils Donselaar, personal communication).

is a polynomial relation between marginal and conditional relative approximation problems: As $\Pr(\mathbf{h} \mid \mathbf{e}) = \frac{\Pr(\mathbf{h},\mathbf{e})}{\Pr(\mathbf{e})}$, any guaranteed relative approximations $q(\mathbf{h})$ and $q(\mathbf{e})$ will give a relative approximation $q(\mathbf{h} \mid \mathbf{e})$ with error $(1+\epsilon)^2$ [8]. However, conditioning on the evidence increases the complexity of approximation in the absolute case. In [8] it was shown that there cannot be any polynomial time absolute approximation algorithm (unless $\mathsf{P} = \mathsf{NP}$) for any $\epsilon < 1/2$ because such an algorithm would decide 3SAT; this follows as a corollary of Cooper's reduction [6]. This reduction uses a singleton hypothesis variable ($|\mathbf{H}| = 1$) and at most three incoming arcs per variable ($d = 3$). This intractability result implies that there cannot be a tractable relative approximation algorithm either.

Result 6 ([6, 8])**.** $\{\epsilon, d, |\mathbf{H}|\}$-AA-CPROB and $\{\epsilon, d, |\mathbf{H}|\}$-RA-CPROB are para-NP-hard.

Corollary 7. $\{\epsilon, d, |\mathbf{H}|\}$-AA-CPROB $\notin$ FERT; $\{\epsilon, d, |\mathbf{H}|\}$-RA-CPROB $\notin$ FERT.

Furthermore, Shimony and Domshlak showed that one cannot relative-approximate CPROB in polynomial time even for directed-path singly-connected Bayesian networks, that is, networks where there exists at most *one* directed path between every pair of nodes (i.e., $c = 1$). Again this results holds for $d = 3$ and a singleton hypothesis variable.

Result 8 ([30])**.** $\{\epsilon, d, c, |\mathbf{H}|\}$-RA-CPROB is para-NP-hard.

Corollary 9. $\{\epsilon, d, c, |\mathbf{H}|\}$-RA-CPROB $\notin$ FERT.

In *rejection sampling* [14] we basically compute a conditional probability by doing forward sampling and rejecting those samples that are not consistent with the evidence. It follows as a corollary from the Hoeffding bound that the number of samples needed for a guaranteed absolute error bound is proportional to the prior probability of the evidence $\Pr(\mathbf{e})$. Likewise, from the Chernoff bound one can also derive that the number of samples needed for a guaranteed *relative* error bound is proportional to the prior probability of the evidence and the posterior probability $\Pr(\mathbf{h})$. This leads to the following results:

Result 10 ([14])**.** $\{\Pr(\mathbf{e}), \epsilon\}$-AA-CPROB $\in$ FERT.

Result 11 ([14])**.** $\{\Pr(\mathbf{e}), \Pr(\mathbf{h}), \epsilon\}$-RA-CPROB $\in$ FERT.

However, one can improve on Result 11. In so-called randomized approximation schemes one can compute bounds on the number of samples needed for efficient relative approximation of CPROB using the zero-one estimator theorem [17]. As the number of samples for a particular query depends on the error ϵ, the confidence δ, and the *posterior* probability $\Pr(\mathbf{h} \mid \mathbf{e})$ (and not on the prior probability of the evidence), the following result follows directly from this theorem:

Result 12 ([7])**.** $\{\Pr(\mathbf{h} \mid \mathbf{e}), \epsilon\}$-RA-CPROB $\in$ FERT.

Results 10-12 depend on the prior probability of the evidence, respectively the posterior probability conditioned on the evidence. In a series of papers, Paul Dagum and colleagues explored efficient relative approximations of CProb that did not depend on these probabilities, but instead depended on properties of the probability distribution, in particular the dependence and variance bounds as explained in the introduction of this section. Again based on Cooper's proof, Dagum and Chavez [7] proved that CProb for every $D_e > 1$ is #P-hard (by reducing from #3Sat[8]); this implies that we cannot have an absolute approximation for arbitrary ϵ as we could then round the approximate result in order to recover the number of satisfying 3Sat instances. They did introduce a randomized approximation scheme using ϵ, δ, and D_e as parameters, where the number of samples depended linearly on the input size. This gives the following results:

Corollary 13 ([7])**.** aa-CProb is #P-hard for every $D_e > 1$.

Result 14 ([7])**.** $\{D_e, \epsilon\}$-ra-CProb $\in$ FERT.

In a later paper [9], the dependency on the evidence in the network in the definition of the dependence value was removed by the introduction of local bounded variance B. Computing MProb exactly is PP-hard even when parameters are arbitrarily close to $1/2$, that is, when B is bounded; this follows as a corollary from the intractability proof of the MAP problem in [26]. The range of the posterior probabilities in this proof is such that the posterior probability of a particular query of interest in a satisfying instance is bigger then twice the probability of an unsatisfying instance, hence we cannot have a relative approximation for any $\epsilon < 1$ because that would solve SAT. In addition to this result, [9] showed that in general $\{B\}$-aa-CProb is NP-hard for any $\epsilon < 1/2$ and any $B > 1$. However, [9] showed that for bounded B, $|\mathbf{H} + \mathbf{E}|$, *and* ϵ one can obtain efficient likelihood weighting algorithms, even when B is defined only on the query and evidence variables, i.e., $B = \max_{i \in \mathbf{H} \cup \mathbf{E}}(\lambda_i)$, rather than on the network as a whole. This gives the following results:

Result 15 (follows from [26])**.** $\{B\}$-ra-CProb is NP-hard for every $\epsilon < 1$.

Corollary 16. $\{B\}$-ra-CProb $\notin$ FERT for any $\epsilon < 1$.

Result 17 ([9])**.** $\{B, |\mathbf{H} + \mathbf{E}|, \epsilon\}$-ra-CProb $\in$ FERT.

In [9] also a de-randomization result was introduced; when, in addition to B and ϵ, also the *depth* of the network (i.e., the length l of the longest path in the graph) is bounded, then CProb can be approximated with a *guaranteed* relative error ϵ in polynomial time. The de-randomized algorithm is not fixed-parameter tractable, however, as the running time is not polynomial in the input size, but sub-exponential (for fixed l and B).

[8]This reduction does *not* suit to prove PP-hardness of the corresponding decision problem, e.g., using the PP-hard Threshold3Sat problem [28]; it is a polynomial-time Turing reduction, rather than a parsimonious reduction, i.e., it uses post-processing after the oracle call to recover the number of solutions. The author is grateful to Fabio Cozman and one of the anonymous reviewers for useful pointers with respect to this topic.

4.2. New results

In addition to the results in the previous sections, that were previously published in the literature, we give a number of new results in this section. We start with some intractability results, after which we will turn our attention to a few new de-randomization results.

4.2.1. Intractability

Some of the intractability results that we list in the overview in Table 4.3 are simple corollaries from well known intractability results. For example, as it is PP-hard to absolutely approximate MProb in general, even with bounded indegree, this implies that $\{d, \Pr(\mathbf{e})\}$-AA-CProb is PP-hard, as we can include an isolated evidence node with arbitrary (non-zero) prior probability without influencing the intractability result.

In this section we will explicitly prove that there cannot be an efficient absolute ϵ-approximation of the MProb problem parameterized on the marginal probability $\Pr(\mathbf{h})$, unless $\mathsf{P} = \mathsf{NP}$. That is, the intractability result presented in Result 1 is independent of the marginal probability; this to contrast to Results 4 and 5 where tractability of RA-MProb *is* conditional on the marginal probability. Our proof strategy is to assume the existence of an algorithm $\mathcal{A}$ for AA-MProb that can, for a given r and ϵ, decide whether $\Pr(\mathbf{h}) > r \pm \epsilon$ whenever $\epsilon' < \Pr(\mathbf{h}) < 1 - \epsilon'$ (for a small value ϵ'), and show that this assumption implies $\mathsf{P} = \mathsf{NP}$. More in particular, our strategy is to construct a Bayesian network from a Satisfiability instance ϕ with n variables, to fix a number $r \in \langle {}^1\!/_{2^{n-1}}, 1 - {}^1\!/_{2^{n-1}} \rangle$, and to define $\mathbf{h}$ such that $\Pr(\mathbf{h}) < r - \epsilon$ if ϕ is unsatisfiable, and $\Pr(\mathbf{h}) > r + \epsilon$ if ϕ is satisfiable.

From an arbitrary Satisfiability instance ϕ (using $\phi = \neg(x_1 \vee x_2) \wedge \neg x_3$ as a running satisfiable example) we construct a Bayesian network $\mathcal{B}_\phi$ as demonstrated in the proof of Theorem 4.1; again, observe that $\Pr(V_\phi = \mathit{true}) = {}^{\#_\phi}\!/_{2^n}$, where $\#_\phi$ is the number of satisfying truth assignments to ϕ; more in particular, $\Pr(V_\phi = \mathit{true}) > 0$ if and only if ϕ is satisfiable. We define $s = r - {}^1\!/_{2^{n^2}}$. In addition to the above described construct, we add a binary variable H to $\mathcal{B}_\phi$, with V_ϕ as only parent, and we set $\Pr(H = \mathit{true} \mid V_\phi = \mathit{true}) = s + {}^1\!/_{2^{n-1}}$ and $\Pr(H = \mathit{true} \mid V_\phi = \mathit{false}) = s$. Figure 4 gives the resulting network for the running example $\phi = \neg(x_1 \vee x_2) \wedge \neg x_3$.

Theorem 4.3. *If an* AA-MProb *instance with* ${}^1\!/_{2^{n-1}} < \Pr(\mathbf{h}) < 1 - {}^1\!/_{2^{n-1}}$ *can be decided in polynomial time, then* $\mathsf{P} = \mathsf{NP}$.

Proof. We reduce Satisfiability to AA-MProb. Let ϕ be a Satisfiability instance with $n \geq 1$ variables and let $\mathcal{B}_\phi$ be the Bayesian network constructed from ϕ as discussed above. Obviously this construction takes polynomial time. Let $r \in \langle {}^1\!/_{2^{n-1}}, 1 - {}^1\!/_{2^{n-1}} \rangle$ and let $s = r - {}^1\!/_{2^{n^2}}$. We then have that $\Pr(H = \mathit{true}) = (s \times 1) + ((s + {}^1\!/_{2^{n-1}}) \times 0) = s$ if ϕ is not satisfiable, and $\Pr(H = \mathit{true}) \geq (s \times (1 - {}^1\!/_{2^n})) + ((s + {}^1\!/_{2^{n-1}}) \times {}^1\!/_{2^n}) = s + {}^2\!/_{2^{n^2}}$ if ϕ is satisfiable. Let $\epsilon < {}^1\!/_{2^{n^2}}$. Obviously, any algorithm that decides in polynomial time whether

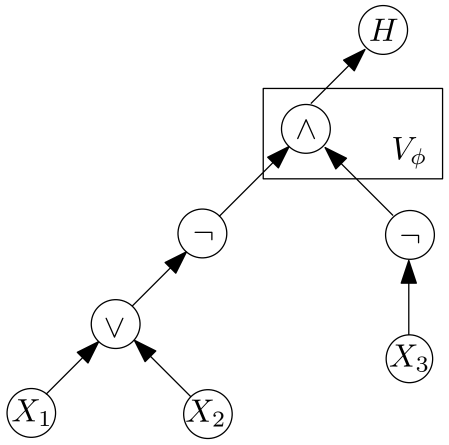


Figure 4: The Bayesian network $\mathcal{B}_\phi$ corresponding to the example SATISFIABILITY instance $\phi = \neg(x_1 \vee x_2) \wedge \neg x_3$.

$\Pr(\mathbf{h}) \pm \epsilon > r$ also decides SATISFIABILITY in polynomial time; from the NP-hardness of SATISFIABILITY we conclude that $\mathsf{P} = \mathsf{NP}$. Note that this result holds for networks with bounded indegree as $d = 2$ in this proof. □

Note that $\mathsf{P} = \mathsf{NP}$ implies that $\mathsf{NP} \subseteq \mathsf{BPP}$, given that BPP is contained in the polynomial hierarchy (cfm. the Sipser-Gács-Lautemann theorem [31]); Theorem 4.3 and the assumption that $\mathsf{BPP} \subsetneq \mathsf{NP}$ implies that $\{d, \Pr(\mathbf{h})\}$-AA-MPROB cannot be fixed-error tractably decidable.

Corollary 18. $\{d, \Pr(\mathbf{h})\}$-AA-MPROB $\notin$ FERT.

4.2.2. De-randomization

We will show that $\{d, \epsilon\}$-AA-MPROB is in XP by de-randomizing Result 2. Let $\{\mathcal{B}, \mathbf{H}, \mathbf{h}, \epsilon\}$ be an instance of AA-MPROB. A simple forward sampling algorithm would compute a single sample for $\mathbf{H}$ using a number of random bits which is linear[9] in the input size $|x|$, i.e., $\mathcal{O}(|x|)$. This algorithm would be run N times to achieve a randomized approximation algorithm, and by Result 2, N would be exponential only in ϵ, and polynomial in the input size. We can de-randomize this algorithm by simulating it for every combination of values of the random bits and making a majority vote, but this would give an exponential time algorithm as the number of random bits is $\mathcal{O}(|x|)$. If $\mathcal{O}(\log |x|)$ *distinct* random bits would suffice, however, we could effectively simulate the randomized algorithm in polynomial time, as the $\mathcal{O}(\log |x|)$ random bits can be amplified

[9]Note that we defined the input size as the total number of bits needed to describe both the graph and the CPTs.

in polynomial time to a k-wise independent $\mathcal{O}(|x|)$-bit random string for every fixed k [1, 25].

In general, we cannot assume k-wise independence in the random bits. As the forward sampling algorithm samples values for variables from their CPTs, given a (previously sampled) joint value assignment to the parents of these variables, the random numbers need to be fully independent for all parents of these variables; that is, if the maximum number of parents any variable has in the network is d (i.e., the maximal indegree of the network) then the random numbers need to be $(d+1)$-wise independent. However, we can effectively de-randomize a simple forward sampling randomized approximation algorithm for AA-MPROB to a polynomial-time deterministic algorithm if d is bounded. A caveat here is that the amplification of the random bits does not run in FPT-time, but in XP-time, i.e., in time $\mathcal{O}(|x|^{c^k})$. This gives us the following result:

Result 19. $\{d, \epsilon\}$-AA-MPROB $\in$ XP.

As a corollary from Results 10 and 19, we can also de-randomize $\{\Pr(\mathbf{e}), \epsilon\}$-AA-CPROB in a similar vein to prove the following result:

Result 20. $\{\Pr(\mathbf{e}), d, \epsilon\}$-AA-CPROB $\in$ XP.

Note that in these cases de-randomization turns fixed-error randomized tractable problems into problems in XP (at the cost of an extra parameter). It is open whether there exists FERT-to-FPT de-randomizations that allow for fixed-parameter tractable deterministic variants of fixed-error randomized tractable stochastic algorithms.

4.3. Overview

The results from the previous sub-sections are summarized in Table 4.3; for brevity we did not include every possible combination of parameters but focused on contrasts. The table shows there are a number of combinations of parameters that can render randomized approximation of posterior probabilities (either by a sampling approach or a Monte Carlo Markov Chain approach) tractable, typically bounding both the quality of approximation ϵ, some property of the probability distribution (like the local variance), and/or some property of the particular query (such as the number of hypothesis variables or the prior probability of the evidence).

5. Conclusion

In this paper we investigated the parameterized complexity of approximate Bayesian inference, both by re-interpretation of known results in the formal framework of fixed-error randomized complexity theory, and by adding a few new results. In addition, we extended the framework of fixed-error randomized tractability analysis with the complexity class XER, which can be seen as the randomized counterpart of XP. The overview in Table 4.3 gives an insight in what constraints can, or cannot, render approximation tractable. These

	MProb		CProb	
Parameters	*abs. approx.*	*rel. approx.*	*abs. approx.*	*rel. approx.*
$\{\epsilon\}$	$\in$ FERT (2)	$\not\in$ FERT (4)	$\not\in$ FERT (7)	$\not\in$ FERT (7)
$\{d\}$	PP-hard (1)	PP-hard (1)	PP-hard (1)	PP-hard (1)
$\{d, \epsilon\}$	$\in$ XP (19)	$\not\in$ FERT (4)	$\not\in$ FERT (7)	$\not\in$ FERT (7)
$\{d, \Pr(\mathbf{h})\}$	$\not\in$ FERT (18)	$\not\in$ FERT (18)	$\not\in$ FERT (18)	$\not\in$ FERT (18)
$\{\Pr(\mathbf{h}), \epsilon\}$	$\in$ FERT (2)	$\in$ FERT (5)	$\in$ FERT (12)	$\in$ FERT (12)
$\{d, \Pr(\mathbf{e})\}$	N/A	N/A	PP-hard (1)	PP-hard (1)
$\{\Pr(\mathbf{e}), \epsilon\}$	N/A	N/A	$\in$ FERT (10)	?
$\{d, \Pr(\mathbf{e}), \epsilon\}$	N/A	N/A	$\in$ XP (20)	?
$\{B\}$	N/A	N/A	$\not\in$ FERT (16)	$\not\in$ FERT$^{*)}$ (16)
$\{B, \epsilon, \|\mathbf{H} + \mathbf{E}\|\}$	N/A	N/A	$\in$ FERT (17)	$\in$ FERT (17)
$\{D_e\}$	N/A	N/A	#P-hard (13)	#P-hard (13)
$\{D_e, \epsilon\}$	N/A	N/A	$\in$ FERT (14)	$\in$ FERT (14)

Table 2: Overview of relevant fixed-parameter and fixed-error tractability and intractability results. The number between the parentheses refers to the number of the result. '$\in$ C' denotes that the parameterized problem $\{k\}$-$\Pi \in$ C, '$\not\in$ FERT' denotes that $\{k\}$-$\Pi \not\in$ FERT unless NP $\subseteq$ BPP; and '?' denotes that this problem is open. N/A means that the parameterization is not applicable for the marginal case where we have not evidence variables. *): This result holds for every $\epsilon < 1$. For $\epsilon \geq 1$ we currently have no relative approximation results.

constraints are notably different from the traditional constraints (tree-width and cardinality) that are necessary and sufficient to render exact computation tractable.

In future work we wish to extend this line of research to other problems in Bayesian networks, notably the MAP problem, extending the work in [21]. The de-randomization results might shed some interesting light on the question whether BPP $\overset{?}{=}$ P by looking at structural properties of the parameterized analogs of these complexity classes.

Acknowledgments

This paper is an extended and improved version of [22]. This research was funded by NWO grants 407-11-040 and 612-001-601. The author is grateful to Nils Donselaar for the reformulation of FERT and XER in Definitions 3.1 and 3.2 and the explicit membership proof in FERT in Results 2 and 5. He thanks Cassio P. De Campos, Todd Wareham, and the anonymous reviewers of both this manuscript and the conference version [22] for useful discussions and valuable comments that greatly improved the paper.

References

[1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583,

1986.

[2] S. Arora and B. Barak. *Complexity Theory: A Modern Approach.* Cambridge, UK: Cambridge University Press, 2009.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties.* Berlin: Springer, 1999.

[4] J.L. Balcázar, R.V. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of the ACM*, 33:603617, 1986.

[5] A.E.F. Clementi, J.D.P. Rolim, and L. Trevisan. Recent advances towards proving P=BPP. In Eric Allender, editor, *Bulletin of the EATCS*, volume 64. EATCS, 1998.

[6] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2):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(3):246–255, 1993.

[8] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

[9] P. Dagum and M. Luby. An optimal approximation algorithm for Bayesian inference. *Artificial Intelligence*, 93:1–27, 1997.

[10] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, Berlin, second edition, 2013.

[11] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, Berlin, 2006.

[12] O. Goldreich. On promise problems: A survey. In *Essays in Memory of Shimon Even*, volume 3895, pages 254–290. Springer, 2006.

[13] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[14] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In L. Kanal and J. Lemmer, editors, *Proceedings of the Second International Conference on Uncertainty in Artificial Intelligence*, pages 149–163, 1986.

[15] R.V. Hogg, J.W. McKean, and A. T. Craig. *Introduction to Mathematical Statistics.* Upper Saddle River, NJ: Pearson Prentice Hall, 6th edition, 2005.

[16] V. Kann. *On the approximability of NP-complete optimization problems.* PhD thesis, Royal Institute of Technology Stockholm, 1992.

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

[18] D. Koller and N. Freeman. *Probabilistic Graphical Models: Principles and Techniques.* Cambridge, MA: MIT Press, 2009.

[19] J. Kwisthout. *The Computational Complexity of Probabilistic Networks.* PhD thesis, Faculty of Science, Utrecht University, The Netherlands, 2009.

[20] J. Kwisthout. Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning*, 52(9):1452 – 1469, 2011.

[21] J. Kwisthout. Tree-width and the computational complexity of MAP approximations in Bayesian networks. *Journal of Artificial Intelligence Research*, 53:699–720, 2015.

[22] J. Kwisthout. The parameterized complexity of approximate inference in Bayesian networks. In A. Antonucci, G. Corani, and C.P. de Campos, editors, *Proceedings of Machine Learning Research*, volume 52, pages 264–274, 2016.

[23] J. Kwisthout, H. L. Bodlaender, and L. C. van der Gaag. The necessity of bounded treewidth for efficient inference in Bayesian networks. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*, pages 237–242. IOS Press, 2010.

[24] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

[25] M. Luby. Removing randomness in parallel computation without a processor penalty. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science (FOCS'88)*, pages 162–173, 1988.

[26] J. D. Park and A. Darwiche. Complexity results and approximation settings for MAP explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.

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

[28] A.E. Porreca, A. Leporati, G. Mauri, and C. Zandron. Elementary active membranes have the power of counting. *International Journal of Natural Computing Research*, 2(3):35–48, 2011.

[29] M. Pradhan and P. Dagum. Optimal Monte Carlo estimation of belief network inference. In E. Horvitz and F. Jensen, editors, *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, pages 446–453. Morgan Kaufmann Publishers Inc., 1996.

[30] S.E. Shimony and C. Domshlak. Complexity of probabilistic reasoning in directed-path singly-connected Bayes networks. *Artificial Intelligence*, 151(1):213–225, 2003.

[31] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[32] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189201, 1979.