The Computational Complexity of Probabilistic Inference Technical Report

Johan Kwisthout
Model Based Systems Development Group,
Faculty of Science, Radboud University Nijmegen,
P.O. Box 9010, 6500GL Nijmegen, The Netherlands
johank@science.ru.nl

April 13, 2011

Abstract

The INFERENCE problem in probabilistic or Bayesian networks—given a probabilistic network and a joint value assignment to a subset of its variables, compute the posterior probability of that assignment—is arguably the canonical computational problem related to such networks. Since it has been proven NP-hard by Cooper in 1990 [2], a number of complexity results have been obtained for the inference problem. For example, the question whether the posterior probability is non-zero is NP-complete [2] and whether it is larger than a threshold q is PP-complete [10]. Roth [14] established #P-completeness for the functional variant (i.e., where the actual probability is computed). These results are based on different reductions and often lack details (e.g., full membership proofs) due to space constraints. In this paper we discuss a unifying construction that reduces each of these problems from the corresponding SATISFIABILITY variant and give membership proofs in full detail, allowing the reader to get a good understanding of these results and some important aspects of Probabilistic Turing Machines, which are the building blocks of these results. In addition, we show that the threshold variant of probabilistic inference, while fixed-parameter tractable for bounded treewidth, remains para-PP-complete even when the threshold is arbitrarily close to 1.

1 Introduction

A probabilistic or Bayesian network [13, 8] $\mathcal{B}$ is a graphical structure that models a set of stochastic variables, the (in-) dependencies among these variables, and a joint probability distribution over these variables. $\mathcal{B}$ includes a directed acyclic graph $\mathcal{G} = (\mathbf{V}, \mathbf{A})$, modeling the variables and (in-) dependencies in the network, and a set of parameter probabilities Γ in the form of conditional probability tables (CPTs), capturing the strengths of the relationships between the variables. The network models a joint probability

distribution $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(v_i \mid \pi(V_i))$ over its variables, where $\pi(V)$ denotes the parents of V in $\mathcal{G}$. We will 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. We will use $\mathbf{E}$ to denote a set of evidence nodes, i.e., a set of nodes for which a particular joint value assignment is observed, and $\mathbf{e}$ to denote a particular observation. Every (posterior) probability of interest in Bayesian networks can be computed using well known lemmas in probability theory, like Bayes' theorem ($\Pr(H \mid E) = \frac{\Pr(E \mid H)\Pr(H)}{\Pr(E)}$), marginalization ($\Pr(H) = \sum_{g_i} \Pr(H \wedge G = g_i)$), and the property that $\Pr(\mathbf{V}) = \prod_{i=1}^{n} \Pr(v_i \mid \pi(V_i))$.

Arguably the most important computational problem related to probabilistic networks, is determining the posterior probability distribution $\Pr(\mathbf{H} \mid \mathbf{E} = \mathbf{e})$ of some set of variables $\mathbf{H}$ given an observation (or evidence) $\mathbf{e}$ for a set of other variables $\mathbf{E}$ in the network. The probability distribution of any variable or set of variables can in essence be calculated using common techniques in probability theory, namely the chain rule, marginalization, and conditioning. However, this calculation can take a time which is exponential in the size of the network, since $\Pr(\mathbf{H}) = \sum_{\mathbf{g_i}} \Pr(\mathbf{H} \wedge \mathbf{G} = \mathbf{g_i})$. Typical algorithms for probabilistic inference are the *message passing* [13], *variable elimination* [3], and *clique tree propagation* (also called *join tree* propagation) [9] approaches. When the network structure is restricted to be a polytree (or *singly connected* graph), inference can be done in polynomial time, for example using the message passing algorithm [13].

No polynomial-time algorithms are known for the inference problem in general, and indeed various problem variants (discussed below) have been shown to be NP-complete [2], PP-complete [10] and #P-complete [14], respectively. The published proofs from the literature are partially based on other reductions: Cooper's proof [2] uses a reduction from Vertex Cover; Littman [10] asserts PP-completeness of the decision variant of the inference problem as a corollary from Roth's #P-completeness result; Roth's [14] proof uses a reduction from Monotone 2SAT to preserve approximation results. Details of the proofs (e.g., explicit membership proofs in PP or #P) are often omitted due to space constraints. In this paper we will show that the hardness results can all be obtained using a single construction and a reduction from the corresponding canonical Satisfiability-variant. Furthermore, we show that an fixed-parameter intractability result can be derived from this construction: Threshold Inference remains para-PP-complete even if the threshold is arbitrarily close to 1.

In this paper we explain the membership proofs in more detail, also taking the opportunity to discuss Probabilistic Turing Machines as building blocks for showing complexity results on probabilistic networks. The reader is assumed to be familiar with the traditional machinery of computational complexity, i.e., non-deterministic Turing Machines, many-one and Turing reductions, and the classes P and NP. In Section 2 we discuss Probabilistic Turing Machines, the classes PP and #P and some of their properties. In Section 3 we give completeness results for four problem variants, namely Positive Inference, Certainty Inference, Threshold Inference and Exact Inference. We discuss both membership in NP, co-NP, PP and #P, respectively, and hardness for these classes; we'll show that a single construction elegantly proves hard-

ness for these problems, using reductions from the appropriate SATISFIABILITY variants. A short conclusion is given in Section 4.

2 Probabilistic Turing Machines

In the remainder of this paper, we assume that the reader is familiar with basic concepts of computational complexity theory, such as Turing Machines, the complexity classes P and NP, and NP-completeness proofs. For more background we refer to classical textbooks like [5, 11]. In addition to these basic concepts, to describe the complexity of various problems we will introduce co-NP (the complement class of NP), Probabilistic Turing Machines, the *probabilistic* class PP, and the *counting* class #P. We use the traditional conventions on Turing Machines, i.e., the set of symbols is $\{0, 1, b\}$ (where b denotes the blank symbol), the states include the accept and reject states q_Y and q_N, additional tapes may be used when needed, and the machine either accepts or rejects an input after polynomial time. The latter is accomplished by maintaining a *clock*: at first, the machine computes, based on its input string x, how many moves it may use. It writes that number on an additional clock tape, decrements the number after each move and enters the reject state q_N when the clock reaches zero.

The class co-NP is defined similarly as the class NP, but with "yes" and "no" answers reversed: a language L is in co-NP if *no* computation paths accept an input $x \in L$ in time, polynomial[1] in x; or, using the dual language L^c, if *all* computation paths accept x. The canonical complete problems for co-NP are UNSATISFIABILITY, respectively TAUTOLOGY: given a Boolean formula ϕ, determine whether it is a contradiction (no possible truth instantiation satisfies ϕ) respectively a tautology (every possible truth instantiation satisfies ϕ).

A *Probabilistic Turing Machine* augments the more traditional non-deterministic Turing Machine with a probability distribution associated with each state transition, e.g., by providing the machine with a tape, randomly filled with symbols [6]. At each transition there may be more than two choice points and the probability that a particular transition is taken does not need to be uniformly distributed; however, as will be demonstrated later on each such machine can be translated in polynomial time into an equivalent machine with binary and uniformly distributed transitions. We assume that, when given an input x, a computation path ends in polynomial time with either accepting or rejecting the input; for the machine as a whole, this means that the input is accepted or rejected in polynomial time with a particular probability p, respectively $1 - p$. The class PP now contains languages L accepted with probability strictly larger than $\frac{1}{2}$ in polynomial time by a Probabilistic Turing Machine.

If all choice points are binary and the probability of each transition is $\frac{1}{2}$, then the *majority* of the computation paths accept an input x if and only if $x \in L$. This majority, however, is not fixed and may (exponentially) depend on the input, e.g., a problem in PP may accept 'yes'-instances with size $|x|$ with probability $\frac{1}{2} + \frac{1}{2^{|x|}}$. This makes problems in PP intractable in general. The canonical PP-complete problem is MAJSAT: given a Boolean formula ϕ, does the majority of the truth instantiations satisfy ϕ? Indeed it is easily shown that MAJSAT encodes the NP-complete SATISFIABILITY problem: take a formula ϕ

[1] Here of course the clock is used in a complementary way, entering the *accept* state q_Y when the clock reaches zero.

with n variables and construct $\psi = \phi \vee x_{n+1}$. Now, the majority of truth assignments satisfy ψ if and only if ϕ is satisfiable, thus $\mathsf{NP} \subseteq \mathsf{PP}$.

An alternative formulation of the class PP is based on *threshold machines*: a k-threshold machine $\mathcal{M}_k$ is a non-deterministic Turing Machine where acceptance of an input x is defined using a threshold k on the number of accepting paths: $\mathcal{M}_k$ accepts x if at least k computation paths of $\mathcal{M}_k$ halt in the accept state q_Y. In this notion of PP, an alternative PP-complete problem is typically used, namely K-SAT: given a formula ϕ and an integer k; is the number of accepting truth assignments to ϕ at least k? This problem has MAJSAT as special case (take $k = 2^{(|\phi|-1)} + 1$) and is thus PP-hard; membership in PP was shown in [15].

Apart from *decision* classes, that are based on some sort of acceptance on a Turing Machine accepting or rejecting inputs, we have *function* classes. Function classes are defined using a Turing Transducer: a Turing Machine with an additional output tape which contains, for each input x, the output $f(x)$; the Turing Transducer is said to *compute* f. For example, FP and FNP are the functional counterparts of P and NP. 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 [16]. The canonical problem in $\#\mathsf{P}$ is #SAT: given a Boolean formula ϕ, give the number of distinct truth instantiations to its variables that satisfies it.

PP and $\#\mathsf{P}$ are closely related; when used as an *oracle* to a deterministic Turing Machine, the complete problems K-SAT and #SAT for the respective classes are equally powerful. An oracle for a particular problem, like #SAT, can be seen as a "black box" magically and in constant time answering problems or computing functions belonging to that problem. A machine $\mathcal{M}$ with oracle access to a problem P, denoted by $\mathcal{M}^P$, is equipped with an additional oracle tape and three additional oracle states q^o_Q, q^o_Y and q^o_N. $\mathcal{M}$ can query the oracle by writing a string x on the oracle tape and enter the oracle query state q^o_Q. The oracle then (immediately) puts $\mathcal{M}$ in either state q^o_Y or q^o_N based on the input x on its tape, and, if the oracle represents a functional problem f, replacing the string x with $f(x)$ on the oracle tape. If P is complete for a class C than we typically use the notation $\mathcal{M}^{\mathsf{C}}$ since we can use (instead of P) any C-complete problem as an oracle without loss of generality. By definition, problems that can be solved with a deterministic Turing machine with oracle access to C are in the class P^{C}; likewise problems that can be solved with a *non*-deterministic Turing machine with oracle access to C are in the class NP^{C}.

The relation between PP and $\#\mathsf{P}$ can be formalized as $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$: any oracle call to a function in $\#\mathsf{P}$ can be simulated with a polynomial number of calls to a decision problem in PP and vice versa[2].

Result 1 ([1]). $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$.

Proof. We prove this theorem using the complete problems #SAT (for $\#\mathsf{P}$) and K-SAT (for PP). Trivially, any K-SAT problem (ϕ, k) can be solved using an algorithm for #SAT as the answer n of the #SAT query represents that n out of $2^{|\phi|}$ paths accept: simply answer "yes" if and only if the answer is greater than k. The other direction requires a little more work. We use binary search to find

[2]And thus, since a polynomial times a polynomial is a polynomial again, P^{PP}=$\mathsf{P}^{\#\mathsf{P}}$.

the lowest value for k in $\{1, 2, \ldots, 2^{|\phi|}\}$ such that K-SAT accepts and output $k-1$. This can be done in polynomial time since we need at most $\log_2(2^{|\phi|}) = |\phi|$ steps in our binary search algorithm. □

3 Complexity results for Inference

In this section we give the known hardness and membership proofs for the following variants of the general INFERENCE problem.

POSITIVE INFERENCE
Instance: Let $\mathcal{B} = (\mathcal{G}, \Gamma)$ be a probabilistic network, and let Pr be its joint probability distribution.Let $\mathbf{H} \subseteq \mathbf{V}$ denote a set of variables with joint value assignment $\mathbf{h}$, and let $\mathbf{E} \subseteq \mathbf{V}$ denote a set of evidence variables with joint value assignment $\mathbf{e}$.
Question: Is $\Pr(\mathbf{h} \mid \mathbf{e}) > 0$?

CERTAINTY INFERENCE
Instance: Let $\mathbf{h}$ and $\mathbf{e}$ be given as in POSITIVE INFERENCE.
Question: Is $\Pr(\mathbf{h} \mid \mathbf{e}) = 1$?

THRESHOLD INFERENCE
Instance: Let $\mathbf{h}$ and $\mathbf{e}$ be given as in POSITIVE INFERENCE. Furthermore, let $0 \leq q < 1$.
Question: Is $\Pr(\mathbf{h} \mid \mathbf{e}) > q$?

EXACT INFERENCE
Instance: Let $\mathbf{h}$ and $\mathbf{e}$ be given as in POSITIVE INFERENCE.
Output: The probability $\Pr(\mathbf{h} \mid \mathbf{e})$.

Note that the first three problems are decision problems and that the last one is a function problem, i.e., a function that can be computed by a Turing Transducer. We will first discuss membership of NP, co-NP, PP, and #P, respectively, for these problems.

3.1 Membership

Lemma 2. POSITIVE INFERENCE *is in* NP.

Proof. To prove membership in NP, it suffices to give a polynomial-time verifiable certificate of membership. Let $\{Y_1, \ldots, Y_n\}$ denote the variables in $\mathbf{V} \setminus \{\mathbf{H} \cup \mathbf{E}\}$. Recall that $\Pr(\mathbf{h}, \mathbf{e}) = \sum_{y_1} \ldots \sum_{y_n} \Pr(Y_1 = y_1, \ldots, Y_n = y_n, \mathbf{h}, \mathbf{e})$ according to the marginalization property. To show that $\Pr(\mathbf{h} \mid \mathbf{e}) > 0$ (and hence that $\Pr(\mathbf{h}, \mathbf{e}) > 0$) we need to show that there exists at least one term $\Pr(y_1, \ldots, y_n, \mathbf{h}, \mathbf{e}) > 0$. A certificate for this purpose consists of the corresponding entries in the CPTs, and can trivially be verified in polynomial time, hence POSITIVE INFERENCE is in NP. □

Lemma 3. CERTAINTY INFERENCE *is in* co-NP.

Proof. To prove membership of co-NP we need to show that any negative instance x can be recognized in polynomial time using a certificate, i.e., we

need to show that we can *falsify* membership in polynomial time. To falsify that $\Pr(\mathbf{h} \mid \mathbf{e}) = 1$ we need to show that there exists at least one term $\Pr(y_1, \ldots, y_n, \mathbf{h}, \mathbf{e}') > 0$ (for a joint value assignment $\mathbf{e}' \neq \mathbf{e}$ to $\mathbf{E}$). Again, such a certificate consists of the corresponding entries in the CPTs, and can be verified in polynomial time, hence CERTAINTY INFERENCE is in co-NP[3]. □

Lemma 4. THRESHOLD INFERENCE *is in* PP.

Proof. To prove membership in PP, we need to show that THRESHOLD INFERENCE can be decided by a Probabilistic Turing Machine $\mathcal{M}$ in polynomial time. To facilitate our proof, we first show how to compute $\Pr(\mathbf{h})$ probabilistically; for brevity we assume no evidence, the proof with evidence goes analogously. $\mathcal{M}$ computes a joint probability $\Pr(y_1, \ldots, y_n)$ by iterating over i using a topological sort of the graph, and choosing a value for each variable Y_i conform the probability distribution in its CPT given the values that are already assigned to the parents of Y_i. Each computation path then corresponds to a specific joint value assignment to the variables in the network, and the probability of arriving in a particular state corresponds with the probability of that assignment. After iteration, we accept with probability $\frac{1}{2}+(1-q)\cdot\epsilon$, if the joint value assignment to $Y_1, \ldots, Y_n$ is consistent with $\mathbf{h}$, and we accept with probability $\frac{1}{2}-q\cdot\epsilon$ if the joint value assignment is *not* consistent with $\mathbf{h}$. The probability of entering an accepting state is hence $\Pr(\mathbf{h})\cdot(\frac{1}{2}+(1-q)\epsilon)+(1-\Pr(\mathbf{h}))\cdot(\frac{1}{2}-q\cdot\epsilon) = \frac{1}{2}+\Pr(\mathbf{h})\cdot\epsilon-q\cdot\epsilon$. Now, the probability of arriving in an accepting state is strictly larger than $\frac{1}{2}$ if and only if $\Pr(\mathbf{h}) > q$. □

For EXACT INFERENCE, showing membership in #P is a bit problematic as #P is defined as the class of counting problems which have a decision variant in NP; a problem is in #P if it computes the number of accepting paths on a particular TM given an input x. Since EXACT INFERENCE is not a counting problem, technically EXACT INFERENCE cannot be in #P; however, we will show that EXACT INFERENCE is in #P modulo a simple normalization. We already showed in the PP-membership proof of THRESHOLD INFERENCE, that we can construct a Probabilistic Turing Machine that accepts with probability q on input $\mathbf{h}$, where $\Pr(\mathbf{h}) = q$. We now proceed to show[4] that there exists a nondeterministic Turing Machine that on input $\mathbf{h}$ accepts on exactly l computation paths, where $\Pr(\mathbf{h}) = \frac{l}{(k!)^{p(|\phi|)}}$ for some number k and polynomial p. The process is illustrated in Figure 1.

Lemma 5. EXACT INFERENCE *is in* #P *modulo normalization.*

Proof. Assume we have a Probabilistic Turing Machine $\mathcal{M}$ whose branches may be non-binary and non-uniform. First we observe that we can translate every j-branch to a uniformly distributed $j!$-branch. Assume for example that at any banch point the probability of the transition from t_i to $\{t_{j_1}, t_{j_2}, t_{j_3}\}$ is given as $\frac{1}{7}$ for t_{j_1}, $\frac{1}{5}$ for t_{j_2}, and $1-(\frac{1}{7}+\frac{1}{5})$ for t_{j_3}. We can replace this transition with a uniform 35-way branch, where five branches end up in t_{j_1}, seven branches end up in t_{j_2} and 23 branches end up in t_{j_3}. Assume the maximum number of branches in the original machine $\mathcal{M}$ was k. After this translation step, we might

[3]Observe that we may also define CERTAINTY INFERENCE in the opposite way, i.e., determine whether $\Pr(\mathbf{h} \mid \mathbf{e}) = 0$, with trivial adjustments in the proof construction; the same holds for the corresponding hardness proof.

[4]Lane A. Hemaspaandra, personal communications, 2011.

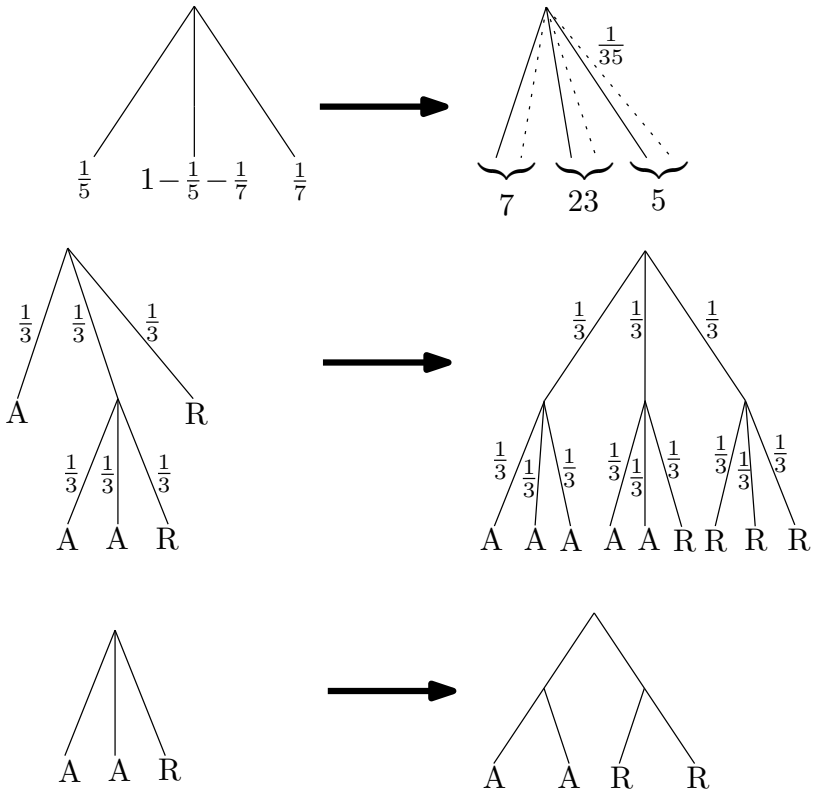

Figure 1: Uniformation, fixing path length and making branch points binary.

up with some branches that are 2-way, some that are 3-way, ..., and some that are k-way. We again rework the machine to obtain only $k!$-branches.

Still, some computation paths may be deeper than others. We remedy this using an normalization approach as in [7] by extending each path to a fixed length, so that each path has the same number of branching points, polynomial in the input size (i.e., $p(|x|)$). Each extended path accepts if and only if the original path accepts and the proportion of accepting and rejecting paths remains the same. We thus have amplified the number of accepting paths to $(k!)^{p(|x|)}$. Lastly, we observe that we can translate each branch (which is a $k!$-way branch) to a sequence of binary branches by taking $z = 2^i$ as the smallest power of 2 larger than $k!$ and constructing a z-way branch (but implemented as i consecutive 2-way branches), where the first $k!$ branches mimic the original behavior, and the remaining $z - k!$ branches all reject. We now have that the number of accepting paths is $(k!)^{p(|x|)}$ times the probability of acceptance of the original Probabilistic Turing Machine, but now we have binary and uniformly distributed transitions and all computation paths of equal length. Given these constraints, this is essentially a #P function as the probability of any computation path is uniformly distributed: essentially we are counting accepting paths on a non-deterministic Turing Machine, modulo a straight normalization (divi-

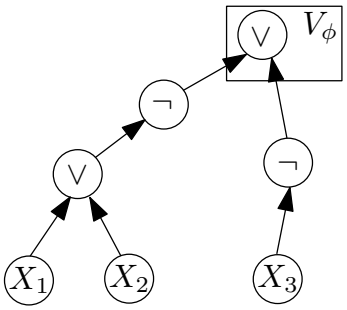

Figure 2: The probabilistic network corresponding to $\neg(x_1 \vee x_2) \vee \neg x_3$

sion by $(k!)^{p(|x|)}$) to obtain a probability rather than an integer. To be precise, there is a function f in #P, a constant k, and a polynomial p such that the probability $\Pr(\mathbf{h})$ is precisely $f(x)$ divided by $(k!)^{p(|x|)}$. □

3.2 Hardness

To prove hardness results for these three problems, we will use a proof technique due to Park and Darwiche [12]. Using this technique they proved that the decision variant of the problem of finding the most likely joint value assignment to a set of variables given *partial* evidence for the complement of that set (the PARTIAL MAP problem), is $\mathsf{NP}^{\mathsf{PP}}$-complete. In the proof, a probabilistic network $\mathcal{B}_\phi$ is constructed from a given Boolean formula ϕ with n variables. For each propositional variable x_i in ϕ, a binary stochastic variable X_i is added to $\mathcal{B}_\phi$, with possible values TRUE and FALSE and a uniform probability distribution. For each logical operator in ϕ, an additional binary variable in $\mathcal{B}_\phi$ is introduced, whose parents are the variables that correspond to the input of the operator, and whose conditional probability table is equal to the truth table of that operator. For example, the value TRUE of a stochastic variable mimicking the *and*-operator would have a conditional probability of 1 if and only if both its parents have the value TRUE, and 0 otherwise. The top-level operator in ϕ is denoted as V_ϕ. In Figure 2 the network $\mathcal{B}_\phi$ is shown for the formula $\neg(x_1 \vee x_2) \vee \neg x_3$.

Now, for any particular truth assignment $\mathbf{x}$ to the set of all propositional variables $\mathbf{X}$ in the formula ϕ we have that the probability of the value TRUE of V_ϕ, given the joint value assignment to the stochastic variables matching that truth assignment, equals 1 if $\mathbf{x}$ satisfies ϕ, and 0 if $\mathbf{x}$ does not satisfy ϕ. Without any given joint value assignment, the prior probability of V_ϕ is $\frac{\#_\phi}{2^n}$, where $\#_\phi$ is the number of satisfying truth assignments of the set of propositional variables $\mathbf{X}$. Note that the above network $\mathcal{B}_\phi$ can be constructed from ϕ in polynomial time.

Lemma 6. POSITIVE INFERENCE *is* NP*-hard.*

Proof. We reduce SATISFIABILITY to POSITIVE INFERENCE. Let ϕ be a SATISFIABILITY-instance and let $\mathcal{B}_\phi$ be the network as constructed above. Now,

$\Pr(V_\phi = \text{TRUE}) > 0$ if and only if there is a satisfying truth assignment to ϕ. □

Lemma 7. CERTAINTY INFERENCE *is* co-NP*-hard.*

Proof. We reduce TAUTOLOGY to CERTAINTY INFERENCE. Let ϕ be a TAUTOLOGY-instance and let $\mathcal{B}_\phi$ be the network as constructed above. Now, $\Pr(V_\phi = \text{TRUE}) = 1$ if and only if ϕ is a tautology[5]. □

Lemma 8. 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 network as constructed above. Now, $\Pr(V_\phi = \text{TRUE}) > \frac{1}{2}$ if and only if the majority of truth assignments satisfy ϕ. □

Lemma 9. EXACT INFERENCE *is* #P*-hard.*

Proof. We reduce #SAT to EXACT INFERENCE, using a parsimoniously polynomial-time many-one reduction, i.e., a reduction that preserves the number of solutions. Let ϕ be a #SAT-instance and let $\mathcal{B}_\phi$ be the network as constructed above. Now, $\Pr(V_\phi = \text{TRUE}) = \frac{l}{2^n}$ if and only if l truth assignments satisfy ϕ. □

3.3 Fixed parameter tractability

While a problem can be NP-hard in general, in some cases it can be nevertheless be solved in polynomial time when some *parameters* of the problem instance are bounded. If we may safely assume that these parameters $\{\kappa_1, \ldots, \kappa_m\}$ always have a low value, then the otherwise intractable problem becomes tractable again. A problem Π is called *fixed parameter tractable* for the set of parameters $\{\kappa_1, \ldots, \kappa_m\}$ if there exists an algorithm solving Π in time $\mathcal{O}(f(\kappa_1, \ldots, \kappa_m) \cdot n^c)$ for any computable function f and constant c; the parameterized problem is in the class FPT if and only if there exits a fixed parameter tractable algorithm solving it. However, if a problem is NP-hard for all (but finitely many) values of the parameter set, then the problem is denoted para-NP-hard for that parameter set. For example, CNF-SAT, parameterized by the number of literals k in each clause, is para-NP-hard, as CNF-SAT enjoys polynomial time algorithms only for $k < 3$ and is NP-hard for $k \geq 3$. Similar notions of intractability exist for other complexity classes like PP. Obviously, FPT $\subseteq$ para-NP; the inclusion is strict unless P = NP[6].

It is known that the inference problem in networks where the moral graph has bounded treewidth and the number of values per variable is bounded by a constant can be solved in linear time, using the clique tree propagation algorithm by Lauritzen and Spiegelhalter [9]. Hence, the inference problem is fixed parameter tractable for the set of parameters $\{tw, v\}$, where v denotes the maximal number of values per variable. The hardness construction discussed above induce para-PP-hardness of THRESHOLD INFERENCE for parameters $\{v, d_{\text{in}}\}$,

[5] Alternatively, when reducing from UNSATISFIABILITY, $\Pr(V_\phi = \text{TRUE}) = 0$ if and only if ϕ is a contradiction.

[6] There exists a whole hierarchy of intermediate parameterized complexity classes, namely the W-hierarchy, that is not used here; the reader is referred to, e.g., [4] for an overview of parameterized complexity.

where d_{in} denotes the maximal indegree of the variables, as all variables are binary and all have an indegree of 2. We will show, using a simple extension to the proof construct, that THRESHOLD INFERENCE is also para-PP-hard for the parameter[7] set $\{q\}$.

Lemma 10. THRESHOLD INFERENCE *is para-*PP*-hard for parameter set* $\{q\}$.

Proof. We reduce MAJSAT to THRESHOLD INFERENCE. Let ϕ be a MAJSAT-instance and let $\mathcal{B}_\phi$ be the network as constructed above. We fix q to be any rational number in $\langle 0, 1\rangle$. We now add a binary variable C_ϕ to our network, with V_ϕ as its only parent, and probability table $\Pr(C_\phi = \text{TRUE} \mid V_\phi = \text{TRUE}) = q + \epsilon$ and $\Pr(C_\phi = \text{TRUE} \mid V_\phi = \text{FALSE}) = q - \epsilon$ for an arbitrary small value ϵ such that $q - \epsilon > 0$ and $q + \epsilon < 1$. Now, $\Pr(C_\phi = \text{TRUE}) > q$ if and only if $\Pr(V_\phi = \text{TRUE}) > \frac{1}{2}$. Hence, THRESHOLD INFERENCE is PP-hard for every fixed q in $\langle 0, 1\rangle$. □

One can interpret these results as follows: while THRESHOLD INFERENCE can be solved tractably when the treewidth of the network and the number of values per variable are small, THRESHOLD INFERENCE remains intractable even when all variables are binary, even when all variables have indegree 2, and even when the threshold probability is arbitrarily close to 1. Observe that the problem degenerates to POSITIVE INFERENCE if the threshold equals 0; deciding whether the probability is exactly 0 or 1 degenerates to CERTAINTY INFERENCE and hence is co-NP-complete.

4 Conclusion

In this paper an overview is given of the computational complexity of POSITIVE INFERENCE, CERTAINTY INFERENCE, THRESHOLD INFERENCE, and EXACT INFERENCE, with a unifying construction for all hardness proofs (using reductions from the appropriate SATISFIABILITY variants) and full details on the membership proofs. In addition, a fixed parameter intractability result is derived from this construction: THRESHOLD INFERENCE is shown to remain para-PP-complete even if the threshold is arbitrarily close to 1. A fair amount of background on the PP and #P complexity classes and the Probabilistic (or Threshold) Turing Machines that are used to define them is given as preliminaries to the main results.

Acknowledgments

The author has been supported by the OCTOPUS project under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Embedded Systems Institute program. The author wishes to thank Todd Wareham and Lane Hemaspaandra for helpful discussions.

[7] We use a loose notion of the term *parameter* here, as parameters are formally natural numbers rather than rationals.

References

[1] J. L. Balcázar, R. V. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of the ACM*, 33:603–617, 1986.

[2] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2):393–405, 1990.

[3] R. Dechter. Bucket elimination: a unifying framework for probabilistic inference. In *Proceedings of the NATO Advanced Study Institute on Learning in Graphical Models*, pages 75–104, 1998.

[4] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer Verlag, Berlin, 1999.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, CA, 1979.

[6] J. T. Gill. Computational complexity of Probabilistic Turing Machines. *SIAM Journal of Computing*, 6(4), 1977.

[7] Y. Han, L. A. Hemaspaandra, and T. Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, 1997.

[8] F. V. Jensen and T. D Nielsen. *Bayesian Networks and Decision Graphs.* Springer Verlag, New York, NY, second edition, 2007.

[9] 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.

[10] M. L. Littman, S. M. Majercik, and T. Pitassi. Stochastic boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296, 2001.

[11] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

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

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

[14] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

[15] J. Simon. *On some central problems in computational complexity.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, NY, 1975.

[16] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.