Error-Correcting and Verifiable Parallel Inference in Graphical Models

Negin Karimi
Department of Computer Science
Aalto University
negin.karimi@aalto.fi

Petteri Kaski
Department of Computer Science
Aalto University
petteri.kaski@aalto.fi

Mikko Koivisto
Department of Computer Science
University of Helsinki
mikko.koivisto@helsinki.fi

Abstract

We present a novel framework for parallel exact inference in graphical models. Our framework supports error-correction during inference and enables fast verification that the result of inference is correct, with probabilistic soundness. The computational complexity of inference essentially matches the cost of w-cutset conditioning, a known generalization of Pearl's classical loop-cutset conditioning for inference. Verifying the result for correctness can be done with as little as essentially the square root of the cost of inference. Our main technical contribution amounts to designing a low-degree polynomial extension of the cutset approach, and then reducing to a univariate polynomial employing techniques recently developed for noninteractive probabilistic proof systems.

Introduction

Graphical Models and Inference. Probabilistic graphical models, Bayesian networks in particular (Pearl 1988), have become the main tool for representation and reasoning under uncertainty. They provide a compact way to specify multivariate probability distributions via conditional independences, along with relatively efficient means for *probabilistic inference*, or *belief updating*; that is, the task of computing the conditional marginal distribution of some query variables, given a configuration of some other variables.

Inference in graphical models is NP-hard, even if one allows approximations to within a given relative error (Dagum and Luby 1993). While polynomial-time algorithms are known for special cases, most notably for models where the underlying graph has bounded treewidth (Lauritzen and Spiegelhalter 1988; Zhang and Poole 1994; Dechter 1999), problem instances of larger treewidth remain a major computational challenge both in theory and in practice.

A Quest for Fast, Robust, and Verifiable Inference. In this paper, we present a new approach for exact inference in graphical models, motivated by two trends. First, the result of inference must admit *fast verification* so that the computational cost of verification is substantially less than the cost of inference—in essence, we want a *proof* that the result of the inference is correct so that the proof can be verified with less resources than it takes to execute inference—this is to enable both safety considerations as well as *delegation* of the inference task to a counterparty that need not be trusted. Second, computing architectures and platforms that enable massively parallel execution, from individual GPUs to service-providers with vast infrastructure, are becoming more widely available. This setting calls for inference algorithms that not only support fast verifiability of the result, but also support massive, efficient parallelization of inference that is robust against sporadic errors in computations and communications, which may be rare yet inevitable given large and increasing numbers of computing units.[1]

Error-Correcting Probabilistic Proof Systems. To meet the desiderata of parallelization, robustness, and fast verifiability for probabilistic inference, we adopt the framework of noninteractive probabilistic proof systems (Williams 2016). Specifically, we instantiate the "Camelot template" (Björklund and Kaski 2016), which has been applied to other fundamental graph problems such as graph coloring and counting small subgraphs (Kaski 2018). The key idea is to reduce the problem at hand to the task of computing values of a univariate polynomial

$$\hat{h}(Z) = \lambda_0 + \lambda_1 Z + \lambda_2 Z^2 + \ldots + \lambda_d Z^d \qquad (1)$$

at distinct points, so that the solution of the original problem (in our case, the result of inference) is immediate once this polynomial ("the proof") is available.

Accordingly, the process of *preparing* the proof is intrinsically parallelizable and amounts to computing sufficiently many point evaluations

$$(\zeta_1, \hat{h}(\zeta_1)), \quad (\zeta_2, \hat{h}(\zeta_2)), \quad \ldots, \quad (\zeta_e, \hat{h}(\zeta_e)) \qquad (2)$$

at distinct values $\zeta_1, \zeta_2, \ldots, \zeta_e$ so that the polynomial $\hat{h}$ becomes uniquely determined, a task that also admits natural *error-correction* by over-provisioning the evaluations.[2] Once a polynomial is available—for example, when it is

[1]For a study of hardware reliability and errors in a leadership supercomputing context, cf. Tiwari et al. (2015).

[2]Indeed, we recall from elementary algebra that a polynomial of degree at most d is uniquely determined from evaluations at any e

supplied to us by a counterparty to whom the task of preprar-ing the proof was delegated—it can be *verified* probabilistically by point evaluations at one or more randomly drawn points, which will always accept the correct polynomial and detect an incorrect polynomial with high probability. In essence, verification can be performed with *one* evaluation, whereas preparation takes at least $d+1$ evaluations, making proof verification substantially faster than proof preparation.

Our Contribution: A Proof System for Inference. When designing such a proof system for inference on graphical models, we need to solve a number of technical issues specific to the inference problem. While we defer a detailed technical development to later sections, an overview is appropriate here. Throughout the paper we work with the *factor graph* representation of probabilistic graphical models (Kschischang, Frey, and Loeliger 2001). Given a factor graph G as input, the inference problem asks us to marginalize over all but the designated query variables X_i, with $i \in B \subseteq \{1, 2, \ldots, n\}$, to obtain the joint distribution g.

Our technical contribution amounts to designing

1. a proof polynomial $\hat{h}$ that extends g so that each point probability $g(X_i = v_i : i \in B)$ in g is available as a point evaluation $\hat{h}(\zeta)$ of $\hat{h}$ at a specific point ζ, and
2. an evaluation algorithm that, given the factor graph G and a point ζ as input, computes the value $\hat{h}(\zeta)$.

The key idea underlying the evaluation algorithm is to transform G for a given ζ by means of fast local transformations to a factor graph $G_{Z=\zeta}$ that marginalizes to the single value $\hat{h}(\zeta)$. This structure in the evaluation algorithm enables us to relate the total resources invested in proof preparation to resources used by fundamental techniques for exact inference on G, namely Pearl's (1986) cutset conditioning and its generalization to w-cutsets (Shachter, Andersen, and Szolovits 1994; Rish and Dechter 2000; Bidyuk and Dechter 2007); a *w-cutset* is a set of variables which, when removed from G leave a graph of treewidth w.[3]

For a concise statement of our main result, denote by m the number of factors, by D the largest number of states per variable, and by $\|G\|$ the total size of the factor graph. Recall that a treewidth-w graph admits an elimination ordering of its vertices that enables solving the inference problem in $O(D^{w+1}m)$ arithmetic operations. Also, let us write $\tilde{O}(t)$ to hide factors polylogarithmic in t.

Theorem 1 *Given a factor graph G, a w-cutset of size k, and a corresponding elimination ordering, the inference problem can be solved with $\tilde{O}(D^{k+1}m)$ evaluations of a proof polynomial, each requiring $\tilde{O}(D^{w+1}m + D^k + \|G\|k)$ arithmetic operations, and one interpolation that requires $\tilde{O}(D^{k+1}m)$ arithmetic operations.*

The total complexity in Theorem 1 is similar to the complexity of w-cutset conditioning, $O(D^k D^{w+1}m)$. Indeed, supposing the first term in the evaluation complexity dominates the other two—an assumption we will discuss soon—the total time complexity is about Dm times the complexity of w-cutset conditioning. Furthermore, like w-cutset conditioning, our algorithm can be parallelized to about D^k processors (fewer if not every variable in the cutset takes exactly D states), each doing $\tilde{O}(D^{w+1}m + D^k + \|G\|k)$ independent operations without communicating with the other processors.

Unlike w-cutset conditioning, Theorem 1 enables fast verification of the result. Indeed, once the result is ready, it can be verified with $\tilde{O}(D^{w+1}m + D^k + \|G\|k)$ operations using *one* processor—essentially, the larger the parameter D^k, the easier it is to verify the result compared with the cost of preparing the result. In essence, the result is a roughly factor-D^k-compressed algebraic digest (the proof polynomial) of the inference process, which enables the detection of an incorrect result with high probability.

The total complexity of the present framework compared with that of cutset conditioning depends on the choice of the cutset. From the previous bounds it follows that our framework is comparable to cutset conditioning essentially when $D^{w+1}m \geq D^k$. This is the case precisely in an application scenario of hard inference, where the factor graph does not have a near-tree-like topology that would enable a small cutset (a small value of k) to yield a near-tree-like residual (a small value of w) after its removal. Rather, our present framework targets the hard inference cases when no choice of a small cutset reduces w considerably below k. In the extreme, this is the case, for example, when the original factor graph has the maximum treewidth $n-1$. In this case the total complexity is $O(D^n m)$ and, by taking $k = w = \lceil n/2 \rceil$, we obtain cost $\tilde{O}(D^{\lceil n/2 \rceil + 1}m + \|G\|n)$ to verify the proof, which is essentially the square root of the total complexity. As such, our present contribution can be seen as an extension of Williams's (2016) celebrated noninteractive Merlin–Arthur proof system for #CNFSAT—the task of counting the number of satisfying assignments to a propositional formula in conjunctive normal form—to the setting of factor graphs and inference by contraction.

Finding a minimum-size w-cutset is an NP-hard problem (Bidyuk and Dechter 2004). Several heuristic algorithms have been proposed for the problem and its weighted variant that takes the exact domain sizes of the cutset variables into account (Larrosa and Dechter 2003; Fishelson and Geiger 2004; Bidyuk and Dechter 2003; 2004; 2007). One can employ any of these algorithms to select an appropriate cutset for the present algorithm (cf. Theorem 1).

To summarize, our contribution is a novel framework for exact inference in graphical models that is highly parallelizable, can withstand silent errors in underlying hardware during proof preparation, and enables safety and delegatability through fast verifiability with probabilistic soundness. Furthermore, the resource overhead for proof preparation is small compared with existing techniques that only perform inference with neither verifiability nor tolerance for errors.

distinct points when $e \geq d+1$; furthermore, any up to $(e-d-1)/2$ of these e evaluations may be in error and the polynomial is still uniquely determined.

[3]The treewidth of a factor graph is the treewidth of the graph whose vertices are the variables and two variables are adjacent if and only if they have a factor in common.

Related Work. Darwiche (2003) proposed a direct representation of the probability distribution of a Bayesian network as a multivariate polynomial where each monomial corresponds to a distinct value assignment to the variables. The representation enables answering inference queries in the "treewidth time" by evaluating and differentiating the polynomial. The work extends previous algorithms of similar complexity characteristics (Bertelè and Brioschi 1972; Lauritzen and Spiegelhalter 1988; Shafer and Shenoy 1990; Zhang and Poole 1994; Dechter 1999). Unlike the polynomial we introduce in the present work, Darwiche's representation supports neither (embarrassingly) parallel computations, error-correction, nor verification.

From the perspective of modern proof systems, the "Camelot template" (Björklund and Kaski 2016) in the present work can be traced back to earlier work on interactive proof systems (e.g., Shamir 1992; Goldwasser, Kalai, and Rothblum 2015; Goldreich 2018) as well as recent work on noninteractive proof systems in fine-grained complexity theory (e.g., Carmosino et al. 2016; Nederlof 2017; Williams 2016). For pointers on recent work in delegating general computation, we refer to Holmgren and Rothblum (2018) and Walfish and Blumberg (2015).

The key algorithmic enabler of the present framework is the fast algorithmic toolbox for computing with univariate polynomials (e.g., von zur Gathen and Gerhard 2013), as well as the possibility to interpolate a low-degree polynomial from partially erroneous evaluations in near-linear time in the size of the input (Gao 2003). The fundamental algorithmic building block in each case is the operation of fast polynomial multiplication, a task which in itself is easily parallelizable both in distributed-memory and shared-memory settings due to the FFT-like structure of the algorithms (Schönhage and Strassen 1971; Fürer 2009; Harvey and van der Hoeven 2019).

Factor Graphs and Inference

This section gives a concise development of the formalism of factor graphs and inference on factor graphs based on the operation of contracting factors.

Mathematical Preliminaries. In what follows we will assume some standard terminology in algebra (e.g., Lang 2002), and will work with the standard fast algorithmic toolbox for computing with univariate polynomials (e.g., von zur Gathen and Gerhard 2013).

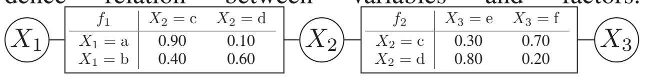
Factor Graphs. Let $X_1, X_2, \ldots, X_n$ be *variables* and let $f_1, f_2, \ldots, f_m$ be *factors*. Associated with each variable X_i there is a finite nonempty set $\mathcal{D}_i$, the *domain* of X_i. Each factor f_k is *incident* to a subset $S_k \subseteq U = \{1, 2, \ldots, n\}$ of the variables. For a set $S \subseteq U$, let us write $\prod_{i \in S} \mathcal{D}_i$, or simply $\mathcal{D}_S$, for the Cartesian product of the sets $\mathcal{D}_i$ with i ranging over S in the natural order of U. The Cartesian product over the empty set is assumed to be the set consisting of the empty set. Let $B \subseteq U$ be a set that indicates the *boundary* or *query* variables.

To avoid degenerate cases, we assume each variable is incident to at least one factor, each variable not in the boundary is incident to at least two factors, and $|\mathcal{D}_i| \geq 2$ for all $i \in U$.

Let $\mathbb{F}$ be a field, such as the field of rational numbers. Associate with each factor f_k a map $\mathcal{D}_{S_k} \to \mathbb{F}$. For a point $v \in \mathcal{D}_{S_k}$, we write $f_k(X_i = v_i : i \in S_k)$, or simply $f_k(v)$, for the value of the map at v.

The variables and their domains, the boundary, and the factors and their associated maps together constitute a *factor graph* G. Let us write $\|G\|$ for the *total size* of G, given by $\sum_{k=1}^{m} |\mathcal{D}_{S_k}|$.

Example. Below we illustrate a small factor graph; the variables are drawn as circles and the factors as boxes, with edges indicating the incidence relation between variables and factors.



f_1	$X_2 = \mathrm{c}$	$X_2 = \mathrm{d}$
$X_1 = \mathrm{a}$	0.90	0.10
$X_1 = \mathrm{b}$	0.40	0.60

f_2	$X_3 = \mathrm{e}$	$X_3 = \mathrm{f}$
$X_2 = \mathrm{c}$	0.30	0.70
$X_2 = \mathrm{d}$	0.80	0.20

Assuming the boundary $B = \{1, 3\}$ above, the factor graph represents the product of the 2×2 matrices given in the factors f_1 and f_2; let us next define this precisely.

The Inference Problem for Factor Graphs. Here and in what follows all arithmetic will take place in the field $\mathbb{F}$. Associated with a factor graph G is a map $g : \mathcal{D}_B \to \mathbb{F}$, the map *represented by* G, defined for all $v \in \mathcal{D}_B$ by

$$g(v) = \sum_{w \in \mathcal{D}_{U \setminus B}} \prod_{k=1}^{m} f_k(v, w)\,, \tag{3}$$

where for conciseness we have abbreviated

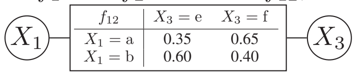
$$g(v) = g\big(X_i = v_i : i \in B\big)$$

and

$$f_k(v, w) = f_k\big(X_i = v_i, X_j = w_j : i \in S_k \cap B, j \in S_k \setminus B\big)\,.$$

In what follows we will tacitly use such abbreviations.

Example. Below we illustrate the product matrix $g = f_{12}$ represented by the factor graph in our example above. The factor graph below also serves to illustrate the contraction of f_1 and f_2 to obtain f_{12}, discussed in the next section.



f_{12}	$X_3 = \mathrm{e}$	$X_3 = \mathrm{f}$
$X_1 = \mathrm{a}$	0.35	0.65
$X_1 = \mathrm{b}$	0.60	0.40

For a factor graph G given as input, the *inference problem* is to compute a complete table of values for the map g. We observe that the map g evaluates to a single scalar value if the boundary B is empty.

Contraction for Inference. Let G be a factor graph and let $1 \leq a \neq b \leq m$. The operation of *contracting* the factors f_a and f_b in G is as follows. For each $i \in U$, let $T_i = \{1 \leq k \leq m : i \in S_k\}$. We say that i is *internal* to the contraction if $i \notin B$ and $T_i \subseteq \{a, b\}$. Let the set $I_{ab} \subseteq U$ consist of all i that are internal to the contraction and observe that $I_{ab} \subseteq S_a \cup S_b$ by our assumption on nondegeneracy. Delete f_a and f_b from G and introduce the factor f_{ab} that is incident to $(S_a \cup S_b) \setminus I_{ab}$ and defined for all $v \in \mathcal{D}_{(S_a \cup S_b) \setminus I_{ab}}$ by

$$f_{ab}(v) = \begin{cases} \sum_{w \in \mathcal{D}_{I_{ab}}} f_a(v, w) f_b(v, w) & \text{if } I_{ab} \neq \emptyset; \\ f_a(v) f_b(v) & \text{if } I_{ab} = \emptyset. \end{cases} \tag{4}$$

Finally, delete the variables X_i with $i \in I_{ab}$ from the factor graph to obtain G_{ab}, the factor graph obtained from the factor graph G by contracting f_a and f_b.

The *cost* of the contraction is $|\mathcal{D}_{S_a \cup S_b}|$. The *cost* of a sequence of contractions on a factor graph is the sum of the costs of the contractions in the sequence. The *cost* of G is the minimum cost of a sequence of contractions that starts from G and results in a factor graph with a single factor. This single factor is associated with the map g in (3) independently of the sequence of contractions performed. Indeed, this property is an immediate consequence of (3), (4), and our assumption on nondegeneracy.

Remark. The contraction operation captures the key step of essentially all standard approaches to exact inference. In variable elimination (Bertelè and Brioschi 1972; Zhang and Poole 1994), for instance, eliminating a variable X_i by summing over its values corresponds to contracting, in a sequential manner, all factors f_k that include the variable in their arguments, i.e., $i \in S_k$; in bucket elimination (Dechter 1999), the set $\{f_k : i \in S_k\}$ is called a bucket. Message-passing algorithms (Lauritzen and Spiegelhalter 1988; Shafer and Shenoy 1990) also perform elimination operations, just twice as many: while the basic elimination algorithm induces a clique tree, in which it eliminates variables from leaf nodes towards a fixed root node, message passing algorithms do the computations in both directions for each edge of the tree; this allows answering multiple inference queries without repeating computations.

A Proof System with Error Correction

This section starts the development of our novel probabilistic proof system for inference on factor graphs that can correct errors in proof preparation. This section defines the proof polynomial and develops its key properties, with the algorithms for preparation and verification of the proof postponed to subsequent sections.

We begin with a first reduction that enables us to subsequently work over finite fields, in particular with modular arithmetic modulo word-bit-length prime moduli.

First Reduction: Chinese Remaindering. Let us first recall a standard reduction via the Chinese Remainder Theorem from inference on rational-valued factor graphs to inference on multiple factor graphs over prime fields; that is, factor graphs where the arithmetic is over the integers modulo a prime. Here we assume that the rational numbers are represented in a radix-point number system with radix R for an integer $R \geq 2$. This is the case, for example, when the factors are represented with standard floating-point numbers.

More precisely, we say that a rational number ρ *admits representation using* d *digits in radix* R if there exists an integer e, a sign $\sigma \in \{-1, 1\}$, and $s_1, s_2, \ldots, s_d \in \{0, 1, \ldots, R-1\}$ such that

$$\rho = \sigma R^e \big(s_1 R^{-1} + s_2 R^{-2} + \ldots + s_d R^{-d}\big)\,.$$

Let G be a factor graph over the rational numbers such that every value ρ in every factor admits representation using d digits in radix R. Suppose furthermore that the exponent e of every value is in the range $\ell \leq e \leq u$. (For example, when the values of the factors are probabilities, we have $0 \leq e \leq 1$.) Since $0 \leq \sum_{j=1}^{d} s_j R^{-j} \leq 1 - R^{-d}$, we have that $R^{d-\ell}\rho$ is an integer with $|R^{d-\ell}\rho| \leq R^{d+u-\ell}$. Accordingly, let us transform G into a new factor graph $G_{\mathbb{Z}}$ by multiplying every value of every factor of G by $R^{d-\ell}$. Then, every factor of $G_{\mathbb{Z}}$ is integer-valued and by (3) we have that $G_{\mathbb{Z}}$ represents the map $g_{\mathbb{Z}}$ with $g_{\mathbb{Z}} = R^{(d-\ell)m} g$. Furthermore, from (3) we have that every value of $g_{\mathbb{Z}}$ is an integer with absolute value at most $M = R^{(d+u-\ell)m} \prod_{j \in U} |\mathcal{D}_j|$. Let $p_1, p_2, \ldots, p_k$ be distinct prime numbers with $p_1 p_2 \cdots p_k \geq 2M + 1$, and let us write $G_{\mathbb{Z}_{p_j}}$ for the factor graph obtained from $G_{\mathbb{Z}}$ by working modulo p_j in the arithmetic. By the Chinese Remainder Theorem, we can reconstruct $g_{\mathbb{Z}}$ and hence g if we have solved $G_{\mathbb{Z}_{p_j}}$ to obtain $g_{\mathbb{Z}_{p_j}}$ for each $j = 1, 2, \ldots, k$.

In what follows we can thus without loss of generality assume that $\mathbb{F}$ is a finite field, implemented in practice with word-bit-length primes and modular arithmetic using, for example, Montgomery multiplication (Montgomery 1985).

The Proof Polynomial. We now proceed with the detailed technical definition of the proof polynomial $\hat{h}$, and then develop its key properties. Let G be a factor graph and let C be a set with $B \subseteq C \subseteq U$. Intuitively, C indicates the cutset of variables in G that will be Vandermonde-conditioned for polynomial extension and removed from the factor graph when later evaluating the proof polynomial; we postpone a technical discussion of the precise relationship to the w-cutset conditioning algorithm until later sections.

Next, introduce a new polynomial indeterminate Y_i for each $i \in C$. For each $i \in C$, select an arbitrary injective map $\eta_i : \mathcal{D}_i \to \mathbb{F}$. Associate with each factor f_k and each $w \in \mathcal{D}_{S_k \setminus C}$ the multivariate polynomial

$$\hat{f}_k\big(Y_i, X_j = w_j : i \in S_k \cap C, j \in S_k \setminus C\big) \in \mathbb{F}\big[Y_i : i \in C\big]$$

that (i) satisfies for each $v \in \mathcal{D}_{S_k \cap C}$ the evaluation identity

$$\begin{aligned} &\hat{f}_k\big(Y_i = \eta_i(v_i), X_j = w_j : i \in S_k \cap C, j \in S_k \setminus C\big) \\ &\quad = f_k\big(X_i = v_i, X_j = w_j : i \in S_k \cap C, j \in S_k \setminus C\big) \end{aligned} \tag{5}$$

and (ii) for all $i \in C$ we have

$$\deg_{Y_i} \hat{f}_k \leq \begin{cases} |\mathcal{D}_i| - 1 & \text{if } i \in S_k \cap C; \\ 0 & \text{if } i \in C \setminus S_k\,. \end{cases} \tag{6}$$

The polynomials $\hat{f}_k$ are unique and can be constructed, for example, by Lagrange interpolation from (5).

Let us extend (3) into a polynomial $\hat{g} \in \mathbb{F}\big[Y_i : i \in C\big]$ by defining

$$\hat{g}(Y_i : i \in C) = \sum_{w \in \mathcal{D}_{U \setminus C}} \prod_{k=1}^{m} \hat{f}_k(Y, w)\,. \tag{7}$$

Select an arbitrary injective map $\tau : \mathcal{D}_C \to \mathbb{F}$ and let Z be a polynomial indeterminate. For each $i \in C$, let $\hat{\ell}_i \in \mathbb{F}[Z]$ be the unique polynomial that satisfies (i) for all $v \in \mathcal{D}_C$ the evaluation identity

$$\hat{\ell}_i\big(Z = \tau(v)\big) = \eta_i(v_i) \tag{8}$$

and (ii) $\deg_Z \hat{\ell}_i \le |\mathcal{D}_C| - 1$. This polynomial can be found, for example, by applying Lagrange interpolation to (8).

Finally, let $\hat{h} \in \mathbb{F}[Z]$ be the polynomial defined by the substitution

$$\hat{h}(Z) = \hat{g}\big(Y_i = \hat{\ell}_i(Z) : i \in C\big) . \tag{9}$$

We say that $\hat{h}$ is the *proof polynomial* associated with G and our choices for C, τ, and η_i for $i \in C$.

Properties of the Proof Polynomial. Let us now establish that the proof polynomial $\hat{h}$ enables us to access g, the map represented by G, through point evaluations. From (9), (8), (7), (6), and (5), we immediately conclude that for all $v \in \mathcal{D}_C$ we have

$$\hat{h}\big(Z = \tau(v)\big) = \sum_{w \in \mathcal{D}_{U \setminus C}} \prod_{k=1}^{m} f_k(v, w) . \tag{10}$$

Accordingly, if we split $v \in \mathcal{D}_C$ into $v = (s, t)$ with $s \in \mathcal{D}_B$ and $t \in \mathcal{D}_{C \setminus B}$, we have for each $s \in \mathcal{D}_B$ the evaluation identity

$$g(s) = \sum_{t \in \mathcal{D}_{C \setminus B}} \hat{h}\big(Z = \tau(s, t)\big) . \tag{11}$$

That is, access to $\hat{h}$ in (9) enables us to access the map g thus solve the inference problem on G.

Let us conclude this section by deriving an upper bound for the degree of $\hat{h}$. From (9), (8), and (6), we observe that

$$\deg_Z \hat{h} \le \big(|\mathcal{D}_C| - 1\big) \sum_{k=1}^{m} \sum_{j \in S_k \cap C} \big(|\mathcal{D}_j| - 1\big) . \tag{12}$$

Evaluating the Proof Polynomial

Next we proceed to develop a fast evaluation algorithm for computing values of $\hat{h}$. The following two subsections develop technical preliminaries towards this end, after which we proceed with the development of the algorithm.

Fast Algorithms for Univariate Polynomials. Let us recall the basic toolkit for computing with univariate polynomials (von zur Gathen and Gerhard 2013). Two univariate polynomials of degree at most d with coefficients over $\mathbb{F}$ can be multiplied in $O\big(\mathrm{M}(d)\big)$ arithmetic operations in $\mathbb{F}$, with $\mathrm{M}(d) = d \log d \log\log d$. A similar arithmetic bound holds for polynomial quotient and remainder. Interpolation from d given points and evaluation of a given degree d polynomial at e given points run in $O\big(\mathrm{M}(d) \log d\big)$ and $O\big(\mathrm{M}(d) + \mathrm{M}(e) \log e\big)$ operations in $\mathbb{F}$, respectively. Accordingly, decoding a Reed–Solomon code of degree d from e given evaluations (with at most $(e - d - 1)/2$ erroneous evaluations for correct decoding) can be done in $O\big(\mathrm{M}(e) \log e\big)$ operations (Gao 2003).

Evaluation and Interpolation of Polynomials. We will need the following basic facts about evaluation and interpolation of polynomials. Recall that the *Vandermonde matrix* for points $\xi_0, \xi_1, \ldots, \xi_d \in \mathbb{F}$ is the $(d+1) \times (d+1)$ matrix

$$V_\xi = V_{\xi_0, \xi_1, \ldots, \xi_d} = \begin{bmatrix} 1 & \xi_0 & \cdots & \xi_0^d \\ 1 & \xi_1 & \cdots & \xi_1^d \\ \vdots & \vdots & & \vdots \\ 1 & \xi_d & \cdots & \xi_d^d \end{bmatrix} , \tag{13}$$

which is invertible if and only if $\xi_0, \xi_1, \ldots, \xi_d$ are distinct. Let $\pi = \begin{bmatrix} \pi_0 & \pi_1 & \cdots & \pi_d \end{bmatrix}^\top \in \mathbb{F}^{(d+1)\times 1}$ be the vector of coefficients for a polynomial $p(X) = \sum_{j=0}^{d} \pi_j X^j \in \mathbb{F}[X]$, and let $p(\xi) = \begin{bmatrix} p(\xi_0) & p(\xi_1) & \cdots & p(\xi_d) \end{bmatrix}^\top \in \mathbb{F}^{(d+1)\times 1}$ be the corresponding vector of evaluations of p. Then we have the evaluation–interpolation identities

$$p(\xi) = V_\xi \pi \quad \text{and} \quad \pi = V_\xi^{-1} p(\xi) . \tag{14}$$

These identities extend to multivariate polynomials (evaluated/interpolated on a set of points with the structure of a Cartesian product) by taking a Kronecker product of Vandermonde matrices, with one matrix for each polynomial indeterminate. This fundamental fact admits a crisp representation in the language of factor graphs, which will form the crux of our evaluation algorithm, described next.

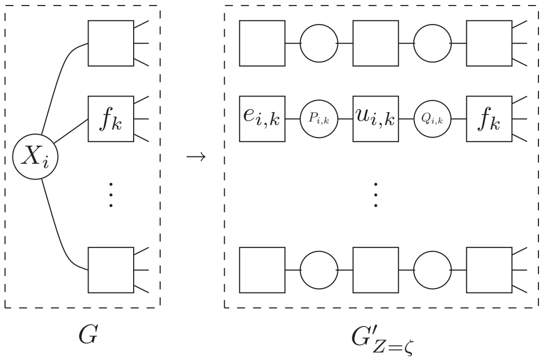
The Evaluation Algorithm. This section develops a fast algorithm for the following problem. Suppose we are given as input a factor graph G, a set C with $B \subseteq C \subseteq U$, the injective maps $\eta_i : \mathcal{D}_i \to \mathbb{F}$ for $i \in C$, the injective map $\tau : \mathcal{D}_C \to \mathbb{F}$, and a point $\zeta \in \mathbb{F}$. Our task is to compute the value of the proof polynomial $\hat{h}$ in (9) at $Z = \zeta$.

Our algorithm for computing $\hat{h}(Z = \zeta)$ is as follows. First, for each $i \in C$, we compute the value $\hat{\ell}_i(\zeta) = \hat{\ell}_i(Z = \zeta) \in \mathbb{F}$ by applying a fast univariate polynomial interpolation algorithm to the desired evaluations given by (8) to recover $\hat{\ell}_i(Z)$ in coefficient form, and the using Horner's rule to obtain the evaluation at $Z = \zeta$. Since each $\hat{\ell}_i(Z)$ has degree at most $|\mathcal{D}_C|$, the total number of arithmetic operations in $\mathbb{F}$ executed in this step is

$$O\big(|C|\, \mathrm{M}(|\mathcal{D}_C|) \log |\mathcal{D}_C|\big) . \tag{15}$$

Next, we transform the factor graph G into a factor graph $G'_{Z=\zeta}$ whose corresponding map has value $\hat{h}(Z = \zeta)$. The key idea is to implement the interpolation from f_k to $\hat{f}_k$ in (5) by inserting (inverses of) Vandermonde matrices (13) with points $\eta_i(\mathcal{D}_i)$ as factors into the factor graph, and then joining each such factor with a factor that represents a Veronese vector $\begin{bmatrix} 1 & \hat{\ell}_i(\zeta) & \cdots & \hat{\ell}_i(\zeta)^{|\mathcal{D}_i|-1} \end{bmatrix}^\top$ so that the resulting factor graph represents evaluation of $\hat{h}$ at $Z = \zeta$.

The precise transformation starting from G will perform a local modification on the factor neighborhood of each variable X_i with $i \in C$ in turn; for convenience we first illustrate what happens in the neighborhood of X_i:



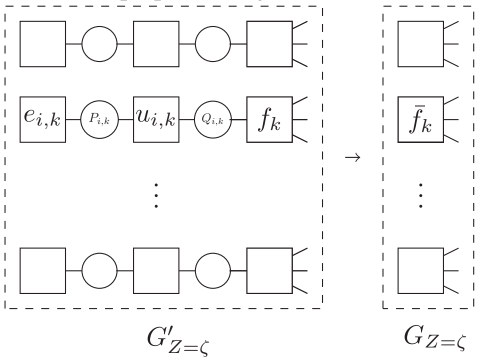
Observe in particular that X_i itself gets removed from the factor graph as a result of the transformation.

Now let us proceed with the detailed description of the transformation. For each variable X_i with $i \in C$, we modify the factor graph as follows. For each factor f_k with incidence $i \in S_k$, introduce two new variables, $P_{i,k}$ and $Q_{i,k}$, as well as two new factors, $e_{i,k}$ and $u_{i,k}$, into the factor graph. Let the variable $P_{i,k}$ have domain $\{0, 1, \ldots, |\mathcal{D}_i| - 1\}$ and the variable $Q_{i,k}$ have domain $\mathcal{D}_i$. Remove the incidence of X_i with f_k and replace it with the incidence of $Q_{i,k}$ with f_k, noting that this is a well-defined operation since both X_i and $Q_{i,k}$ have domain $\mathcal{D}_i$. Let $Q_{i,k}$ be incident also to $u_{i,k}$. Let $P_{i,k}$ be incident with $e_{i,k}$ and $u_{i,k}$. Define the maps associated with $e_{i,k}$ and $u_{i,k}$ for all $c \in \{0, 1, \ldots, |\mathcal{D}_i| - 1\}$ and $v_i \in \mathcal{D}_i$ by

$$e_{i,k}(c) = \hat{\ell}_i(\zeta)^c \quad \text{and} \quad u_{i,k}(c, v_i) = \left[V^{-1}_{\eta_i(\mathcal{D}_i)}\right]_{v_i,c}, \quad (16)$$

where $\left[V^{-1}_{\eta_i(\mathcal{D}_i)}\right]_{v_i,c}$ refers to the entry at row v_i, column c of the inverse of the Vandermonde matrix $V_{\eta_i(\mathcal{D}_i)}$ developed over the points $\eta_i(v_i) \in \mathbb{F}$ with v_i ranging over $\mathcal{D}_i$. We observe that the inverse of $V_{\eta_i(\mathcal{D}_i)}$ exists because η_i is injective. Once all the incidences of X_i with a factor f_k are transformed in this way, we remove the variable X_i from the factor graph. Indeed, at this point X_i is incident to no factor, so we can safely remove X_i and proceed to consider the next i, if any. When all i have been considered, set the boundary B to the empty set. The resulting factor graph is $G'_{Z=\zeta}$ and its associated map has the value $\hat{h}(Z = \zeta)$, which follows by (16), (14), (9), (7), and (5).

Let us next preprocess the factor graph $G'_{Z=\zeta}$ by contractions to obtain the factor graph $G_{Z=\zeta}$. Again a local illustration of the preprocessing transformation is convenient:



In precise terms, starting with the factor graph $G'_{Z=\zeta}$, for each $i \in C$ and each $1 \le k \le m$ with $i \in S_k$, first contract the factors $e_{i,k}$ and $u_{i,k}$, then contract the result of the first contraction with the factor f_k to obtain the factor $\bar{f}_k$. The cost of the first contraction is $|\mathcal{D}_i|^2$ and the cost of the second contraction is at most $|\mathcal{D}_{S_k}|$. Thus, the entire preprocessing sequence of contractions has cost at most

$$\sum_{i \in C} \sum_{1 \le k \le m : i \in S_k} \left(|\mathcal{D}_i|^2 + |\mathcal{D}_{S_k}|\right), \quad (17)$$

which can be further lowered to $\tilde{O}\big(|C| \cdot \|G\|\big)$ arithmetic operations by using repeatedly fast univariate polynomial interpolation in place of explicitly constructing the factors $u_{i,k}$ and contracting them; in what follows, we will assume that this faster design is used.

Once the preprocessing is complete, each factor $\bar{f}_k$ has domain $\mathcal{D}_{S_k \setminus C}$. Let us write $G_{Z=\zeta}$ for the factor graph resulting from these contractions.

Finally, the evaluation algorithm proceeds to contract factors in $G_{Z=\zeta}$ so that only a single factor with value $\hat{h}(Z = \zeta)$ remains. This concludes our description of the algorithm.

We observe that the computational cost of the evaluation algorithm is governed by the cost of its three parts: (i) the cost of preparing the new factors in $G'_{Z=\zeta}$, bounded by (15); (ii) the cost preprocessing $G'_{Z=\zeta}$ to $G_{Z=\zeta}$, bounded by (17); and (iii) the cost of contracting $G_{Z=\zeta}$. Our goal next is to control the cost (iii) in terms of the cost of the w-cutset conditioning algorithm.

The Cutset Conditioning Algorithm Recalled. At this point it will be convenient to briefly review the operation of the cutset conditioning algorithm. Let G be a factor graph and let C indicate the cutset of variables with $B \subseteq C \subseteq U$. For each $r = r_C \in \mathcal{D}_C$ in turn, start with the factor graph G, and execute the following local transformation. For each factor f_k in G such that $S_k \cap C$ is nonempty, replace the factor f_k with the factor $\underline{f}_k$ incident to $S_k \setminus C$ and defined for all $v \in \mathcal{D}_{S_k \setminus C}$ by the rule

$$\underline{f}_k(v) = f_k(v, r). \quad (18)$$

Finally, delete the variable X_i for each $i \in C$ from the factor graph and set the boundary to the empty set. Let us write $G_{X_C = r_C}$ for the resulting factor graph. Writing g for the map represented by G, from (3) and (18) we observe that $G_{X_C = r_C}$ contracts to the value $g(r_C)$. That is, by iterating over all the possible choices of $r_C \in \mathcal{D}_C$, a task which can be executed in parallel on multiple processors, and contracting each $G_{X_C = r_C}$ to a single factor, we recover g.

We observe that the total cost of the cutset conditioning algorithm is essentially $|\mathcal{D}_C|$ times the cost of contracting the factor graph $G_{X_C = r_C}$, where this contraction cost is independent of the choice of r_C.

The Cost of the Evaluation Algorithm. An immediate but significant observation is now that, for an identical choice of the cutset C in the evaluation algorithm and in the cutset conditioning algorithm, respectively, the factor graphs $G_{Z=\zeta}$ and $G_{X_C = r_C}$ have identical contraction cost. Indeed, the two factor graphs are otherwise identical expect in terms of the values of the maps associated with the factors.

It follows that an analysis of the cost (iii) in the evaluation algorithm reduces to an analysis of the cost of the cutset conditioning algorithm.

We can now establish the coarse-grained upper bound for the cost of the evaluation algorithm in Theorem 1. Let $D = \max_{i\in U} |\mathcal{D}_i|$ and $k = |C|$. We observe that (15), which controls the cost of part (i) of the algorithm, is bounded by $\tilde{O}(D^k)$. The cost of part (ii) is controlled by (17) and bounded by $\tilde{O}\big(|C|\cdot\|G\|\big)$. Finally, the cost of part (iii) agrees with the cost of contraction in the w-cutset conditioning algorithm, which is bounded by $\tilde{O}(D^{w+1}m)$. This establishes the bound $\tilde{O}\big(D^{w+1}m + D^k + |C|\cdot\|G\|\big)$ in Theorem 1.

Proof Preparation and Verification

Now that we have developed the evaluation algorithm for computing values of the proof polynomial, let us analyse proof preparation and proof verification in more detail, including completing the proof of Theorem 1.

Preparing and Error-Correcting the Proof. From (12) and (2) we have that independent parallel runs of the evaluation algorithm at e distinct points with $e \geq d+1$ for

$$d = \big(|\mathcal{D}_C| - 1\big)\sum_{k=1}^{m}\sum_{j\in S_k\cap C}\big(|\mathcal{D}_j| - 1\big) \qquad (19)$$

suffice to uniquely determine the proof polynomial $\hat{h}$, even in the presence of at most $(e-d-1)/2$ erroneous evaluations. Furthermore, from such evaluations we can recover the coefficient form (1) of the polynomial as well as identify the erroneous evaluations in $O\big(\mathrm{M}(e)\log e\big)$ operations in $\mathbb{F}$ (Gao 2003).

From (19) we have the more coarse-grained upper bound

$$d \leq |\mathcal{D}_C|\cdot m|C|\max_{j\in C}|\mathcal{D}_j|\,,$$

whose right-hand-side can in turn can be bounded by $\tilde{O}(D^{|C|+1}m)$ with $D = \max_{i\in U}|\mathcal{D}_i|$. In particular, we have that $\tilde{O}(D^{|C|+1}m)$ evaluations of the proof polynomial suffice to reconstruct the coefficient form (1). This establishes the claim on the number of evaluations in Theorem 1. The proof of Theorem 1 is now almost complete; what remains is to develop the procedure for verifying the proof polynomial.

Verifying the Proof. Let us now review how the proof can be verified with probabilistic soundness using randomized polynomial identity testing, which is a well-known technique and presented here for completeness of exposition only. Suppose that we are given as input a factor graph G, a cutset C with $B\subseteq C\subseteq U$, the injective maps $\eta_i : \mathcal{D}_i \to \mathbb{F}$ for $i\in C$, the injective map $\tau : \mathcal{D}_C \to \mathbb{F}$, and a polynomial $\hat{h}' \in \mathbb{F}[Z]$ of degree at most d in coefficient form

$$\hat{h}'(Z) = \lambda'_0 + \lambda'_1 Z + \lambda'_2 Z^2 + \ldots + \lambda'_d Z^d\,. \qquad (20)$$

We seek to verify whether $\hat{h}'$ equals $\hat{h}$, where $\hat{h}$ is the proof polynomial (9) associated with G, C, η_i, τ.

First, without loss of generality we can assume that the degree of both $\hat{h}'$ and $\hat{h}$ is bounded by d with (19). Indeed, if this is not the case, we must have $\hat{h}' \neq \hat{h}$ and thus we can immediately reject the given $\hat{h}'$ as a bad proof.

Next, let us perform the following randomized test on $\hat{h}'$. Draw a uniform random element $\zeta\in\mathbb{F}$. Use Horner's rule on the coefficient reprensentation (20) to recover the value $\hat{h}'(\zeta)\in\mathbb{F}$ in $O(d)$ arithmetic operations in $\mathbb{F}$. Next, use the evaluation algorithm on $G, C, \eta_i, \tau, \zeta$ to obtain the value $\hat{h}(\zeta)\in\mathbb{F}$ of the true proof polynomial $\hat{h}$ at ζ. Finally, test whether $\hat{h}'(\zeta) = \hat{h}(\zeta)$ and either accept or reject $\hat{h}'$ as the correct proof polynomial $\hat{h}$ accordingly.

Let us now analyze the test. It is immediate that $\hat{h}'(\zeta) = \hat{h}(\zeta)$ holds always when $\hat{h}' = \hat{h}$. That is, the correct proof always passes the test. When $\hat{h}' \neq \hat{h}$, we observe that the difference polynomial $\hat{h}' - \hat{h}$ is a not-identically-zero polynomial of degree at most d, and thus has at most d roots by the Fundamental Theorem of Algebra. Furthermore, in this case we have $\hat{h}'(\zeta) = \hat{h}(\zeta)$ if and only if ζ is a root of the difference polynomial $\hat{h}' \neq \hat{h}$, and this happens with probability at most $d/|\mathbb{F}|$ over the uniform random choice of $\zeta\in\mathbb{F}$. Assuming that $d \lll |\mathbb{F}|$, a single test will thus result with high probability in the outcome $\hat{h}'(\zeta)\neq\hat{h}(\zeta)$ and thus the detection of the bad proof.[4] Furthermore, r independent repetitions of the test can be used to amplify the probability of detecting a bad proof to at least $1-(d/|\mathbb{F}|)^r$. This completes the proof of Theorem 1.

Let us stress that while preparing the proof takes $d+1$ evaluations with d given in (19), verifying the proof takes *one* evaluation (or r evaluations when amplified) and so is considerably more efficient than proof preparation.

Accessing the Result of Inference. Once the proof polynomial $\hat{h}$ is available in coefficient form (1) and has been verified, the map $g : \mathcal{D}_B \to \mathbb{F}$ represented by G can be recovered using fast polynomial batch evaluation on the points $\tau(\mathcal{D}_C)\subseteq\mathbb{F}$ to recover the values $\hat{h}(\tau(s,t))$ for all $(s,t)\in\mathcal{D}_B\times\mathcal{D}_{C\setminus B}$. In particular, by (11) we can then recover $g(s) = \sum_{t\in\mathcal{D}_{C\setminus B}}\hat{h}(\tau(s,t))$ for all $s\in\mathcal{D}_B$. This process is dominated by the cost of batch-evaluating the degree-at-most d polynomial $\hat{h}$, and takes $O\big(\mathrm{M}(d) + \mathrm{M}\big(|\mathcal{D}_C|\big)\log|\mathcal{D}_C|\big)$ operations in $\mathbb{F}$. A slightly more coarse-grained bound is $\tilde{O}(D^{|C|+1}m)$ operations.

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement 338077 "Theory and Practice of Advanced Search and Enumeration" as well as from the Academy of Finland, Grant 316771.

[4]Here it should be perhaps stressed that a bad proof will be detected with high probability even when the bad proof has been adversarially prepared by an all-powerful adversary.

References

Bertelè, U., and Brioschi, F. 1972. *Nonserial Dynamic Programming*. Academic Press.

Bidyuk, B., and Dechter, R. 2003. An empirical study of w-cutset sampling for Bayesian networks. In Meek, C., and Kjærulff, U., eds., *UAI '03, Proc. 19th Conference in Uncertainty in Artificial Intelligence*, 37–46. Morgan Kaufmann.

Bidyuk, B., and Dechter, R. 2004. On finding minimal w-cutset. In Chickering, D. M., and Halpern, J. Y., eds., *UAI '04, Proc. 20th Conference in Uncertainty in Artificial Intelligence*, 43–50. AUAI Press.

Bidyuk, B., and Dechter, R. 2007. Cutset sampling for Bayesian networks. *J. Artif. Intell. Res.* 28:1–48.

Björklund, A., and Kaski, P. 2016. How proofs are prepared at Camelot: Extended abstract. In Giakkoupis, G., ed., *Proc. 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016*, 391–400. ACM.

Carmosino, M. L.; Gao, J.; Impagliazzo, R.; Mihajlin, I.; Paturi, R.; and Schneider, S. 2016. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In Sudan, M., ed., *Proc. 2016 ACM Conference on Innovations in Theoretical Computer Science*, 261–270. ACM.

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

Darwiche, A. 2003. A differential approach to inference in Bayesian networks. *J. ACM* 50(3):280–305.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artif. Intell.* 113(1-2):41–85.

Fishelson, M., and Geiger, D. 2004. Optimizing exact genetic linkage computations. *J. Comput. Biol.* 11(2/3):263–275.

Fürer, M. 2009. Faster integer multiplication. *SIAM J. Comput.* 39(3):979–1005.

Gao, S. 2003. A new algorithm for decoding Reed–Solomon codes. In Bhargava, V. K.; Poor, H. V.; Tarokh, V.; and Yoon, S., eds., *Communications, Information, and Network Security*. Springer. 55–68.

Goldreich, O. 2018. On doubly-efficient interactive proof systems. *Found. Trends Theor. Comput. Sci.* 13(3):158–246.

Goldwasser, S.; Kalai, Y. T.; and Rothblum, G. N. 2015. Delegating computation: Interactive proofs for muggles. *J. ACM* 62(4):27:1–27:64.

Harvey, D., and van der Hoeven, J. 2019. Integer multiplication in time $O(n \log n)$. Manuscript https://hal.archives-ouvertes.fr/hal-02070778 .

Holmgren, J., and Rothblum, R. 2018. Delegating computations with (almost) minimal time and space overhead. In Thorup, M., ed., *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, 124–135. IEEE Computer Society.

Kaski, P. 2018. Engineering a delegatable and error-tolerant algorithm for counting small subgraphs. In Pagh, R., and Venkatasubramanian, S., eds., *Proc. Twentieth Workshop on Algorithm Engineering and Experiments, ALENEX 2018*, 184–198. SIAM.

Kschischang, F. R.; Frey, B. J.; and Loeliger, H. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory* 47(2):498–519.

Lang, S. 2002. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, third edition.

Larrosa, J., and Dechter, R. 2003. Boosting search with variable elimination in constraint optimization and constraint satisfaction problems. *Constraints* 8(3):303–326.

Lauritzen, S. L., and Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* 50(2):157–194.

Montgomery, P. 1985. Modular multiplication without trial division. *Math. Comp.* 44(170):519–521.

Nederlof, J. 2017. A short note on Merlin-Arthur protocols for subset sum. *Inform. Process. Lett.* 118:15–16.

Pearl, J. 1986. Fusion, propagation, and structuring in belief networks. *Artif. Intell.* 29(3):241–288.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Rish, I., and Dechter, R. 2000. Resolution versus search: Two strategies for SAT. *J. Autom. Reason.* 24(1/2):225–275.

Schönhage, A., and Strassen, V. 1971. Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)* 7:281–292.

Shachter, R. D.; Andersen, S. K.; and Szolovits, P. 1994. Global conditioning for probabilistic inference in belief networks. In de Mántaras, R. L., and Poole, D., eds., *UAI '94: Proc. Tenth Annual Conference on Uncertainty in Artificial Intelligence*, 514–522. Morgan Kaufmann.

Shafer, G. R., and Shenoy, P. P. 1990. Probability propagation. *Ann. Math. Artif. Intell.* 2(1):327–351.

Shamir, A. 1992. IP = PSPACE. *J. ACM* 39(4):869–877.

Tiwari, D.; Gupta, S.; Gallarno, G.; Rogers, J.; and Maxwell, D. 2015. Reliability lessons learned from GPU experience with the Titan supercomputer at Oak Ridge leadership computing facility. In Kern, J., and Vetter, J. S., eds., *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015*, 38:1–38:12. ACM.

von zur Gathen, J., and Gerhard, J. 2013. *Modern Computer Algebra*. Cambridge University Press, third edition.

Walfish, M., and Blumberg, A. J. 2015. Verifying computations without reexecuting them. *Commun. ACM* 58(2):74–84.

Williams, R. R. 2016. Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation. In Raz, R., ed., *31st Conference on Computational Complexity, CCC 2016*, volume 50 of *LIPIcs*, 2:1–2:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Zhang, N. L., and Poole, D. L. 1994. A simple approach to Bayesian network computations. In *Proc. Tenth Canadian Conference on Artificial Intelligence*, 171–178.