Björklund, Andreas; Kaski, Petteri; Williams, Ryan

Solving systems of polynomial equations over GF(2) by a parity-counting self-reduction

Published in:
46th International Colloquium on Automata, Languages, and Programming, ICALP 2019

DOI:
10.4230/LIPIcs.ICALP.2019.26

Published: 01/07/2019

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Björklund, A., Kaski, P., & Williams, R. (2019). Solving systems of polynomial equations over GF(2) by a parity-counting self-reduction. In I. Chatzigiannakis, C. Baier, S. Leonardi, & P. Flocchini (Eds.), *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019* (pp. 1-13). [26] (Leibniz international proceedings in informatics; Vol. 132). Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing. https://doi.org/10.4230/LIPIcs.ICALP.2019.26

Solving Systems of Polynomial Equations over GF(2) by a Parity-Counting Self-Reduction

Andreas Björklund
Department of Computer Science, Lund University, Sweden

Petteri Kaski
Department of Computer Science, Aalto University, Finland

Ryan Williams
Department of Electrical Engineering and Computer Science & CSAIL, MIT, Cambridge, MA, USA

Abstract

We consider the problem of finding solutions to systems of polynomial equations over a finite field. Lokshtanov et al. [SODA'17] recently obtained the first worst-case algorithms that beat exhaustive search for this problem. In particular for degree-d equations modulo two in n variables, they gave an $O^*\big(2^{(1-1/(5d))n}\big)$ time algorithm, and for the special case $d = 2$ they gave an $O^*\big(2^{0.876n}\big)$ time algorithm.

We modify their approach in a way that improves these running times to $O^*\big(2^{(1-1/(2.7d))n}\big)$ and $O^*\big(2^{0.804n}\big)$, respectively. In particular, our latter bound – that holds for all systems of quadratic equations modulo 2 – comes close to the $O^*\big(2^{0.792n}\big)$ expected time bound of an algorithm empirically found to hold for *random* equation systems in Bardet et al. [*J. Complexity*, 2013]. Our improvement involves three observations:

1. The Valiant-Vazirani lemma can be used to reduce the solution-finding problem to that of counting solutions modulo 2.
2. The monomials in the probabilistic polynomials used in this solution-counting modulo 2 have a special form that we exploit to obtain better bounds on their number than in Lokshtanov et al. [SODA'17].
3. The problem of solution-counting modulo 2 can be "embedded" in a smaller instance of the original problem, which enables us to apply the algorithm as a subroutine to itself.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases equation systems, polynomial method

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.26

Category Track A: Algorithms, Complexity and Games

Funding *Andreas Björklund*: The Swedish Research Council grant VR 2016-03855 "Algebraic Graph Algorithms".
Petteri Kaski: 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".
Ryan Williams: The U.S. National Science Foundation under grants CCF-1741638 and CCF-1741615.

1 Introduction

We study the problem of finding a simultaneous root to a system of m polynomials

$$P_1(x) = 0\,, \quad P_2(x) = 0\,, \quad \ldots\,, \quad P_m(x) = 0 \tag{1}$$

over n variables $x = (x_1, x_2, \ldots, x_n)$. The computational tractability of this problem is known to dramatically depend on the domain of the variables and polynomial coefficients. Over the integers, the problem is undecidable, by Matiyasevich's celebrated solution of

46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 26; pp. 26:1–26:13

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Hilbert's tenth problem on the algorithmic decidability of Diophantine equations [10]. Over an algebraically closed field, the problem reduces via Hilbert's Nullstellensatz to deciding whether 1 belongs to the ideal generated by the polynomials, which for polynomials with rational coefficients can be decided in exponential space by computing a reduced Gröbner basis for the ideal [3, 4, 11]. Over the integers *modulo two* – our object of study in this paper – the problem is NP-complete even when the polynomials are severely constrained. Indeed, systems of polynomial equations modulo two enable the compact modeling of a versatile range of tasks. For example, one can easily represent k-CNFSAT formulas on n Boolean variables by expressing each clause in the formula as a degree-k polynomial [5]. A similar reduction from NAE3SAT proves that the problem remains NP-hard even in the case of quadratic equations modulo two. As the fastest known worst-case algorithms for k-CNF satisfiability (for example, see Moser and Scheder [13]) run in time $2^{n-\Omega(n/k)}$, it is a difficult challenge to design faster-than-$2^{n-\Omega(n/k)}$-time algorithms for solving systems of degree-k polynomial equations. Still it is interesting to ask precisely how much savings over the brute-force $O^*\big(2^n\big)$-time solution one can obtain, particularly in the case of quadratic equations, since the postulated hardness of solving quadratic systems modulo two forms the basis of several proposed cryptographic primitives, such as HFE proposed by Patarin [15] and UOV proposed by Kipnis, Patarin, and Goubin [8].

An $O^*\big(2^{0.792n}\big)$ expected-time algorithm for a system of $m = n$ quadratic polynomials over n variables modulo two was proposed by Bardet, Faugère, Salvy, and Spaenlehauer [1]. However, their algorithm only works for systems satisfying certain algebraic assumptions, and these assumptions were only experimentally verified to hold for the vast majority of such systems. Still, further refinements of the method makes it practical even for small systems [6], and the algorithmic ideas underlies the to date fastest known implementation we are aware of [14].

Such a result highlights the potential vulnerability of cryptographic primitives whose security is based on the postulated hardness of solving random systems. However, from a theoretical viewpoint it is preferable to have rigorous proofs and algorithms that work efficiently on all inputs. The algorithm of Bardet et al. [1] is based on finding proofs of non-solvability, a so-called effective Nullstellensatz of finding low-degree polynomials H_i such that

$$\sum_i H_i(x')P_i(x', r) = 1,$$

for each specialisation r of the polynomials, i.e. after replacing a fixed subset of the variables to a restriction r. The algorithm then continues to look for solutions only in those specialisations r for which no proof was found. The search for proofs is formulated as a linear equation system whose dimensions depend on the bound of the degree in the proofs. The argument made in their paper is that for most equation systems with sufficiently more equations than variables, "small-degree" polynomials can be used in the proofs. Getting rigorous bounds on the degree appears to be a difficult problem, and in the worst case some systems will likely require large-degree proofs.

Lokshtanov, Paturi, Tamaki, Williams, and Yu [9] took a radically different approach and presented an $O^*\big(2^{0.876n}\big)$-time algorithm for quadratic equation systems modulo two that uses no algebraic assumptions at all and works for all quadratic systems. They also gave a general $O^*\big(2^{n-n/(5d)}\big)$ time algorithm for systems of equations of degree bounded by d.

Their approach uses the so-called "polynomial method": the entire system of equations is randomly replaced by a single "probabilistic" polynomial that has a "small" exponential number of terms, and is consistent with the system on exponentially many assignments. This polynomial is then evaluated quickly on many assignments using an FFT or fast matrix multiplication, gaining an advantage over exhaustive search.

We present a new algorithm that largely follows the approach of Lokshtanov et al. [9] but offers a few simplifications to their scheme. Most prominently, we will consider the problem of computing the parity of the number of solutions, rather than the decision problem directly. Whereas Lokshtanov et al. [9] apply a random parity sieve on monomials of a subset of the variables to implement a decision-to-parity reduction within the algorithm, our main observation is that this can be done on the system itself. That is, rather than explicitly being tailored into the algorithm, we can reduce decision to the parity problem by adding random affine equations to the system. This is based on the well-known theorem of Valiant and Vazirani [18] in complexity theory to isolate solutions to Boolean satisfiability by adding random equations. Our analysis in Section 2.5 is borrowed from theirs and is included here solely for the sake of completeness.

One immediate effect of our alternative parity-counting approach is that it reduces the need for random bits from exponential in n to merely polynomial in n. A more interesting gain is that our approach leads to faster algorithms, via two further observations. Our algorithm for quadratic systems runs in $O^*\big(2^{0.804n}\big)$ time, and for degree-d systems we provide a $O^*\big(2^{n-n/(2.7d)}\big)$ time algorithm. Our running time for quadratic equation systems in particular comes much closer to the $O^*\big(2^{0.792n}\big)$ running time of Bardet et al. [1]. To get a quantitive feeling of our incremental result, note:

1. We can solve quadratic systems modulo two with 9% more variables in about the same time as the algorithm of [9].
2. Our algorithm for degree-3 systems is faster than the one for degree-2 systems in [9].

Our first observation is that the seemingly more difficult problem of computing the parity of solutions apparently makes it easier to identify the structure of monomials in the probabilistic polynomials used in the polynomial method. Making use of this structure leads to better bounds on their number and (indirectly) on the total running time, and is the source of most of our improvement. Our second observation is that the parity-summation part in the method is identical to the original problem, leading to a self-reduction: we can use our algorithm as a subroutine to itself, again leading to a faster algorithm.

We present our algorithm for computing the parity of the number of solutions to a polynomial equation system in Section 3. We shall highlight the differences to the original decision algorithm by Lokshtanov et al. [9] as we go along. We begin by some preliminaries in the subsequent section.

2 Preliminaries

Here we review some notation and well-known facts. Let $\mathbb{F}_2$ denote the field of two elements; that is, integer arithmetic modulo two. For a non-negative integer n, we write $[n]$ for the set $\{1, 2, \ldots, n\}$. For a finite set D, we write 2^D for the power set of D, $\binom{D}{k}$ for the set of all k-element subsets of D, and $\binom{D}{\downarrow k} = \bigcup_{j=0}^{k} \binom{D}{j}$ for the set of all at-most-k-element subsets of D. Accordingly, we write $\binom{n}{\downarrow k} = \sum_{j=0}^{k} \binom{n}{j}$. For a function $f(n)$, the notation $O^*\big(f(n)\big)$ suppresses factors polynomial in n.

2.1 Yates's algorithm

Let us write I_ℓ for an $\ell \times \ell$ identity matrix. For an $s \times t$ matrix A and a non-negative integer n, the n^{th} Kronecker power of A factors into the sequence of matrices

$$A^{\otimes n} = \prod_{j=1}^{n} \left(I_s^{\otimes j-1} \otimes A \otimes I_t^{\otimes n-j} \right). \tag{2}$$

Each matrix $I_s^{\otimes j-1} \otimes A \otimes I_t^{\otimes n-j}$ is sparse with at most $s^{j-1} \cdot st \cdot t^{n-j} = s^j t^{n-j+1}$ nonzero entries. Thus, using sparse matrix–vector multiplication along the sequence (2), we may multiply the matrix $A^{\otimes n}$ with a given vector in at most

$$2\sum_{j=1}^{n} s^j t^{n-j+1} = \begin{cases} 2ns^{n+1} & \text{if } s = t, \\ \frac{2st(s^n - t^n)}{s-t} & \text{if } s \neq t \end{cases} \tag{3}$$

operations on scalars. This algorithm is known as Yates's algorithm [19].

2.2 The fast zeta transform for the subset lattice

The matrix $\zeta = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ is invertible over any ring, with inverse $\zeta^{-1} = \left[\begin{smallmatrix} 1 & 0 \\ -1 & 1 \end{smallmatrix}\right]$. In particular, in the field $\mathbb{F}_2$ of two elements, we have $\zeta = \zeta^{-1}$. Let x and y be vectors whose components are indexed by the subsets in $2^{[n]}$. Then, the matrix–vector multiplication $y = \zeta^{\otimes n} x$ implements the linear map $x \mapsto y$ defined for all $B \subseteq [n]$ by $y_B = \sum_{A \subseteq B} x_A$. This map is the *zeta transform* for the lattice $(2^{[n]}, \subseteq)$. By (3) in Section 2.1, Yates's algorithm can be used to implement the zeta transform in $O(2^n n)$ operations. This algorithm is known as the *fast zeta transform*. The inverse transform is called the Möbius transform. In characteristic 2, these transforms coincide. The zeta transform remains invertible when the relevant vectors and matrices are restricted from $2^{[n]}$ to $\binom{[n]}{\downarrow d}$. The corresponding restriction of Yates's algorithm runs in $O\left(\binom{n}{\downarrow d} n\right)$ operations. See [2, 7] and the references therein for more on fast zeta transforms.

2.3 Polynomials modulo two: the monomial basis and the evaluation basis

Observe $x^2 = x$ holds for all $x \in \mathbb{F}_2$. Thus, WLOG, an n-variate polynomial $f = f(x_1, x_2, \ldots, x_n)$ in the polynomial ring $\mathbb{F}_2[x_1, x_2, \ldots, x_n]$ consists of only *multilinear* monomials indicated by a function $M_f : 2^{[n]} \to \mathbb{F}_2$ with

$$f = \sum_{Y \subseteq [n]} M_f(Y) \prod_{j \in Y} x_j \,.$$

Intuitively, $M_f(S)$ gives the coefficient of $\prod_{i \in S} x_i$ in the (unique) multilinear polynomial representing f. In particular, $\mathbb{F}_2[x_1, x_2, \ldots, x_n]$ is a 2^n-dimensional vector space over $\mathbb{F}_2$, and the function M_f, viewed as a vector with entries indexed by $2^{[n]}$, represents f in the *monomial basis*. We say that f has *degree* at most d if M_f vanishes outside $\binom{[n]}{\downarrow d}$. The monomial basis is the algebraic normal form of the Boolean function.

Associate each vector $x \in \mathbb{F}_2^n$ with the subset $X = \{i \in [n] : x_i = 1\} \subseteq [n]$. Define the *evaluation map* $E_f : 2^{[n]} \to \mathbb{F}_2$ for all $X \subseteq [n]$ by the rule $E_f(X) = f(x)$, where x is the vector corresponding to X. In what follows we often find it convenient to abuse notation slightly and write simply $f(X)$ in place of $E_f(X)$. Viewing the function E_f as a vector with entries indexed by $2^{[n]}$, we say that E_f represents f in the *evaluation basis*.

The monomial basis and the evaluation basis are related by the zeta transform. That is, for all $f \in \mathbb{F}_2[x]$, we have

$$E_f = \zeta^{\otimes n} M_f \,. \tag{4}$$

Indeed, for all $Z \subseteq [n]$ we have

$$E_f(Z) = f(Z) = \sum_{Y \subseteq Z} M_f(Y) \,.$$

By properties of the zeta transform, the basis-change identity (4) holds also when restricted from $2^{[n]}$ to $\binom{[n]}{\downarrow d}$; that is, when restricted from arbitrary polynomials to polynomials of degree at most d. Fast zeta transforms enable fast basis changes between the monomial basis and the evaluation basis as needed.

2.4 From finding to decision, from decision to parity-counting

The task of finding a solution to a given system of polynomial equations reduces to the task of deciding whether a given system has at least one solution. Indeed, assuming the system has a solution, we may try both values 0 and 1 to a selected variable, and focus on one assignment that indicates that the system after the substitution of the value has a solution. Thus, finding a solution takes at most $2n$ queries to a decision algorithm.

The task of deciding whether a given system of polynomial equations has a solution reduces to computing the parity of the number of such solutions by randomized isolation techniques. One elegant isolation technique is Valiant–Vazirani [18] affine hashing, which inserts $O(n)$ random linear equations into the system, without increasing the number of variables. For completeness, we recall affine hashing in Section 2.5. Thus, from here on, we consider the problem of counting the parity of solutions to a system of polynomial equations.

2.5 Valiant–Vazirani affine hashing

For completeness, this section recalls Valiant–Vazirani [18] affine hashing for isolating a unique solution (if any) by introducing a collection of random linear equations into to the system of polynomial equations. In particular, affine hashing does not increase the number of variables or the degree of the system, only the number of equations increases.

Let $\mathcal{S} \subseteq \{0,1\}^n$ be the set of solutions the system of polynomial equations. If $\mathcal{S}$ is empty there is nothing to isolate, so let us assume that $\mathcal{S}$ is nonempty in what follows. Let $k = 0, 1, \ldots, n$ be the unique integer such that $2^k \leq |\mathcal{S}| < 2^{k+1}$.

Draw independent uniform random values $\alpha_{ij} \in \{0,1\}$ for $i = 1, 2, \ldots, k+2$ and $j = 1, 2, \ldots, n$. For each $i = 1, 2 \ldots, k+2$, draw an independent uniform random value $\beta_i \in \{0,1\}$ and introduce the linear equation

$$\sum_{j=1}^{n} \alpha_{ij} x_j = \beta_i \tag{5}$$

into the system of polynomial equations.

Let us say that a solution $x \in \mathcal{S}$ *survives* if it satisfies every introduced equation (5). Let us write S_x for the event that x survives, and U_x for the event that x is the unique solution in $\mathcal{S}$ that survives. We want to control the probability

$$\Pr(U_x) = \Pr\bigl(S_x \cap \textstyle\bigcap_{y \in \mathcal{S} \setminus \{x\}} \bar{S}_y\bigr) \,.$$

By the union bound, we have

$$\Pr\big(S_x \cap \bigcap_{y\in\mathcal{S}\setminus\{x\}} \bar{S_y}\big) \geq \Pr(S_x) - \sum_{y\in\mathcal{S}\setminus\{x\}} \Pr\big(S_x \cap S_y\big)\,.$$

By independence of the events S_x and S_y for all $x, y \in \mathcal{S}$ with $x \neq y$, we have

$$\Pr(S_x) - \sum_{y\in\mathcal{S}\setminus\{x\}} \Pr\big(S_x \cap S_y\big) = \Pr(S_x)\left(1 - \sum_{y\in\mathcal{S}\setminus\{x\}} \Pr(S_y)\right).$$

By mutual independence of the $k+2$ equations, we have

$$\Pr(S_x) = \frac{1}{2^{k+2}}\,.$$

Hence,

$$\Pr(U_x) \geq \frac{1}{2^{k+2}}\left(1 - \frac{2^{k+1}}{2^{k+2}}\right) = \frac{1}{2^{k+3}}\,.$$

By mutual exclusiveness of the events U_x, we have

$$\Pr\big(\bigcup_{x\in\mathcal{S}} U_x\big) = \sum_{x\in\mathcal{S}} \Pr(U_x) \geq 2^k \frac{1}{2^{k+3}} = \frac{1}{8}\,.$$

From $(1-\frac{1}{8})^r \leq \exp(-\frac{r}{8}) \leq \epsilon$ we observe that $r = \lceil \ln \epsilon^{-1} \rceil$ independent repetitions will isolate a unique solution in $\mathcal{S}$ with probability at least $1-\epsilon$. Furthermore, we do not know the value of k, but we can exhaustively try out all the values $k = 0, 1, \ldots, n$ with $\epsilon = \frac{1}{n}$ so that a solution, if one exists, will be isolated and hence witnessed as odd parity in the solution space with high probability in total $O(n \log n)$ repetitions of the parity-counting algorithm.

3 A randomized reduction from parity-counting to itself

This section presents our technical contribution. All arithmetic in this section is over $\mathbb{F}_2$. Our task is to determine the parity of the number of solutions $x \in \{0,1\}^n$ to a given system of degree-d polynomial equations

$$P_1(x) = 0, \quad P_2(x) = 0, \quad \ldots, \quad P_m(x) = 0\,. \tag{6}$$

We present a randomized self-reduction that reduces (6) to multiple similar systems of degree at most d but over $\ell = \lambda n$ variables for a constant $0 < \lambda < 1$. Optimizing λ and applying the reduction recursively yields our main result.

3.1 Parity-counting as summation over the domain

We start with the elementary observation that determining the parity of the number of solutions to (6) amounts to computing the sum

$$I_F = \sum_{x\in\{0,1\}^n} F(x) \tag{7}$$

of the polynomial function

$$F(x) = (1 + P_1(x))(1 + P_2(x)) \cdots (1 + P_m(x))\,. \tag{8}$$

Indeed, for $x \in \{0,1\}^n$ we have $F(x) = 1$ if and only if x is a solution to (6), and otherwise $F(x) = 0$. For comparison, Lokshtanov et al. [9] used

$$J_F = \bigvee_{x \in \{0,1\}^{n-n'}} \left(\sum_{y \in \{0,1\}^{n'}} s_y \cdot F(x,y) \right) \tag{9}$$

with s_y independently and uniformly sampled scalars from $\{0,1\}$, with the observation that J_F is one with probability at least $1/2$ if the original system has a solution, and is always zero if the original system has no solutions. In the following, we show how to obtain a faster algorithm by dealing with I_F instead.

We do not know how to quickly evaluate I_F directly, but we will take an indirect and randomized approach to perform the summation.

3.2 Approximate the summand F by a low-degree probabilistic polynomial

The main difficulty in directly working with the polynomial F of (8) is its degree, which could be dm in general. As in Lokshtanov et al. [9], we construct a *probabilistic* polynomial $\tilde{F}$ with the property that for $0 < \epsilon \leq 1$ and for all $x \in \{0,1\}^n$ we have

$$\Pr_{f \in \tilde{F}} (F(x) = f(x)) \geq 1 - \epsilon \,. \tag{10}$$

We use the following construction generally credited to Razborov [16] and Smolensky [17]. For $i = 1, 2, \ldots, \lceil \log_2 \epsilon^{-1} \rceil$ and $j = 1, 2, \ldots, m$, draw an independent uniform random value $\rho_{ij} \in \{0,1\}$, and construct the polynomials

$$R_i(x) = \sum_{j=1}^{m} \rho_{ij} \cdot P_j(x) \,. \tag{11}$$

Let us now study the polynomial

$$f(x) = (1 + R_1(x))(1 + R_2(x)) \cdots (1 + R_{\lceil \log_2 \epsilon^{-1} \rceil}(x)) \,. \tag{12}$$

We easily observe that (10) holds. Furthermore, since each of the polynomials R_i has degree at most d, the degree of f is at most $d\lceil \log_2 \epsilon^{-1} \rceil$, rather than the degree $\Omega(dm)$ of F.

3.3 Sum the parts of multiple independent approximations

Suppose we replace the summands $F(x)$ in the computation of $I_F = \sum_x F(x)$ with $f(x)$'s from (12). By doing so we have reduced the degrees of the summands, but we also introduced a difficulty in the process: the summands $f(x)$ may introduce errors in the computation of I_F. We resolve this issue by drawing a sample of $s = O(n)$ independent Razborov–Smolensky approximations $f_1, f_2, \ldots, f_s \in \tilde{F}$, and then sum each of these, *in parts*.

Let us define precisely what we mean. Suppose our summand is g. Let us view g as the set function $g : 2^{[n]} \to \{0,1\}$ defined over the set of subsets of $[n]$. Let $A, B \subseteq [n]$ be disjoint with $A \cup B = [n]$. Think of A and B as a partition of n variables (indexed by $[n]$) into two parts; a subset $X \subseteq A$ will be construed as a 0-1 assignment to the variables in A, and a subset $Z \subseteq B$ will be construed as a 0-1 assignment to the variables in B. We will compute (7) as

$$I_F = \sum_{Z \in 2^B} \sum_{X \in 2^A} F(X \cup Z).$$

This is similar to Lokshtanov et al. [9] with $A = [n']$ in (9), except we take the actual sum modulo 2 over 2^B assignments, instead of a disjunction over the assignments.

For every $Z \subseteq B$ (construed as a 0-1 assignment to the variables in B), define the function

$$g|_A^{Z\to B} : 2^A \to \{0,1\}$$

for all $X \subseteq A$ by

$$g|_A^{Z\to B}(X) = g(X \cup Z)\,.$$

That is, $g|_A^{Z\to B}$ is the *part of* g obtained from fixing the variables in B to the 0-1 assignment g by Z. Each part of g is a polynomial over the variables in A.

Summing g in parts now amounts to computing the sums of all parts

$$I_{g|_A^{Z\to B}} = \sum_{X\subseteq A} g(X \cup Z) \quad \text{for each} \quad Z \subseteq B\,.$$

Once we have the sums of all parts, we can easily compute the overall sum:

$$I_g = \sum_{X\subseteq[n]} g(X) = \sum_{Z\subseteq B} I_{g|_A^{Z\to B}}\,.$$

However, we will *not* sum up the parts' sums directly. Indeed, we are summing potentially erroneous approximations of the true summands, and obtaining the (full) sum of a potentially erroneous approximation is not what we want. What we want, with high probability, is the sum of the true summands.

3.4 Correct the sum of each part by "scoreboarding"

Recall that we proposed to work with a sample of $s \le O(n)$ independent polynomials $f_1, f_2, \ldots, f_s \in \tilde{F}$ that approximate the true summand F. Suppose we have summed each approximation, in parts, to obtain the summand of each part of each approximation. That is, for each $Z \subseteq B$ we have the *scoreboard* of s sums

$$I_{f_1|_A^{Z\to B}} = \sum_{X\subseteq A} f_1(X\cup Z), \quad I_{f_2|_A^{Z\to B}} = \sum_{X\subseteq A} f_2(X\cup Z), \quad \ldots, \quad I_{f_s|_A^{Z\to B}} = \sum_{X\subseteq A} f_s(X\cup Z).$$

Each of these s sums is $\{0,1\}$-valued. Assuming s is odd, we take the unique majority value across the scoreboard and set, for every $Z \subseteq B$,

$$\tilde{I}_Z = \text{Majority}\big(I_{f_1|_A^{Z\to B}}, I_{f_2|_A^{Z\to B}}, \ldots, I_{f_s|_A^{Z\to B}}\big) = \begin{cases} 1 & \text{if } \sum_{j=1}^s I_{f_j|_A^{Z\to B}} > \frac{s}{2}\,, \\ 0 & \text{if } \sum_{j=1}^s I_{f_j|_A^{Z\to B}} < \frac{s}{2}\,. \end{cases} \tag{13}$$

Consider now the true summand F and a sum of its part $I_{F|_A^{Z\to B}}$. We can control the probability of error $\Pr[\tilde{I}_Z \neq I_{F|_A^{Z\to B}}]$ as follows. First, set $\epsilon = 2^{-(|A|+2)}$ and use the union bound with (10) to conclude that, for all $Z \subseteq B$ and $j = 1, 2, \ldots, s$, we have

$$\Pr\big[I_{F|_A^{Z\to B}} = I_{f_j|_A^{Z\to B}}\big] \ge 1 - 2^{|A|}\cdot\epsilon \ge \frac{3}{4}\,. \tag{14}$$

Consequently, the approximate summands f_j are bounded in degree by at most

$$\Delta = \lceil \log_2 \epsilon^{-1} \rceil d = (|A| + 2)d\,. \tag{15}$$

Second, recalling that $f_1, f_2, \ldots, f_s$ are independent, we can use a standard Chernoff bound to control the error from scoreboarding. When T is a sum of independent identically distributed random variables, for all $0 \leq \delta \leq 1$ it holds that [12]

$$\Pr\big[T \leq (1-\delta)\mathbb{E}[T]\big] \leq \exp\big(-\tfrac{\delta^2 \mathbb{E}[T]}{2}\big)\,. \tag{16}$$

Take $T_Z = \sum_{j=1}^{s} I_{f_j|_A^{Z\to B}}$. From (14) we have

$$\begin{aligned} \mathbb{E}[T_Z \,|\, I_{F|_A^{Z\to B}} = 1] &\geq \tfrac{3}{4}s\,, \\ \mathbb{E}[s - T_Z \,|\, I_{F|_A^{Z\to B}} = 0] &\geq \tfrac{3}{4}s\,, \end{aligned} \tag{17}$$

Recalling that we assume s to be odd, from (16) and (17) with $\delta = 1/3$ we thus have

$$\begin{aligned} \Pr\big[T_Z > \tfrac{s}{2} \,\big|\, I_{F|_A^{Z\to B}} = 1\big] &\geq 1 - \exp\big(-\tfrac{s}{24}\big)\,, \\ \Pr\big[s - T_Z > \tfrac{s}{2} \,\big|\, I_{F|_A^{Z\to B}} = 0\big] &\geq 1 - \exp\big(-\tfrac{s}{24}\big)\,. \end{aligned} \tag{18}$$

Set $s = 48n + 1$ and use (18) in (13) to conclude that

$$\Pr\big[\tilde{I}_Z = I_{F|_A^{Z\to B}}\big] \geq 1 - 2^{-2n}\,.$$

Taking a union bound over all $Z \subseteq B$, we have

$$\Pr\big[I_F = \textstyle\sum_{Z\subseteq B} \tilde{I}_Z\big] \geq 1 - 2^{-n}\,.$$

That is, error-correction by scoreboarding enables us to recover the sum $I_F = \sum_x F(x)$ with only exponentially small error probability. All that now remains is to sum in parts.

3.5 Summing the parts of a low-degree polynomial

As before, we view the summand as a set function $f : 2^{[n]} \to \{0,1\}$ and assume that the underlying polynomial representing f has degree at most Δ. For each $Z \subseteq B$, we want to produce the sum $I_{f|_A^{Z\to B}} = \sum_{X\subseteq A} f(X \cup Z)$.

Our strategy for summation will rely on the fact that these sums are linearly dependent, and fast basis changes via fast zeta transforms will enable fast summation of parts.

To witness the linear dependence, let us put to use the fact that f is low-degree. Let $M_f : 2^{[n]} \to \{0,1\}$ be the representation of f in the monomial basis (M_f maps monomials to coefficients). For all $W \subseteq [n]$, we thus have

$$f(W) = \sum_{U\subseteq W} M_f(U)\,.$$

Now recall that f has degree at most Δ if and only if M_f vanishes on subsets of size greater than Δ. That is, the representation in the monomial basis is sparse; we seek to express our sums in this basis.

Toward this end, let us study summing a part from the perspective of the coefficients M_f rather than f. For each $Z \subseteq B$, we have

$$I_{f|_A^{Z\to B}} = \sum_{X\subseteq A} f(X \cup Z) = \sum_{X\subseteq A} \sum_{U\subseteq X\cup Z} M_f(U) = \sum_{Y\subseteq Z} M_f(A \cup Y)\,. \tag{19}$$

The last equality in (19) follows because *every monomial (viewed as a subset) not containing A will cancel modulo two*, because it contributes to the sum an even number of times. This

property of contributing monomials being known to contain A is the key difference from the approach of Lokshtanov et al. [9]; the property is lost if we take a disjunction (as in (9)) instead of a sum modulo 2 (as in (7)). It will yield a smaller upper bound on the number of monomials.

Let us now get some corollaries of (19). First, since only monomials that contain A contribute to the sums $I_{f|_A^{Z\to B}}$, knowledge of only these monomials is sufficient information to compute the sum $I_{f|_A^{Z\to B}}$ for each $Z \subseteq B$. Second, we know that M_f vanishes on all subsets of size greater than Δ. Since each monomial must contain A, we only have to consider subsets from B of size at most $\delta = \Delta - |A|$, compared to $\delta = \Delta$ used in Lokshtanov et al. [9].

These two corollaries yield the following three-step strategy for summing the parts:

(i) Compute $I_{f|_A^{B\to Z}} = \sum_{X\subseteq A} f(X \cup Z)$ for each set $Z \in \binom{B}{\downarrow\delta}$.
By (19), we thus have $\binom{|B|}{\downarrow\delta}$ equations for the $\binom{|B|}{\downarrow\delta}$ unknowns $M_f(A \cup Y)$, for $Y \in \binom{B}{\downarrow\delta}$.

(ii) Solve the equations for the values $M_f(A \cup Y)$ for $Y \in \binom{B}{\downarrow\delta}$ (the coefficients of f as a polynomial).

(iii) Use the solved values to produce the sum $I_{f|_A^{B\to Z}}$ for each $Z \subseteq B$.

Observe that the only actual summations are made in (i). The step (iii) produces the sums in batch from the values obtained in (ii). To solve the equations in (ii), we use the fast zeta transform over the sets in $\binom{B}{\downarrow\delta}$. For the step (iii), use the fast zeta transform over 2^B with the knowledge that $M_f(A \cup Y)$ vanishes in 2^B outside $Y \in \binom{B}{\downarrow\delta}$.

3.6 Summing a part reduces back to parity-counting

Let us now define in detail what it means to sum a part in step (i). First, let us parameterise the partition A, B of $[n]$. For $1 \le \ell \le n$, set

$$|A| = \ell \qquad \text{and} \qquad |B| = n - |A| = n - \ell\,.$$

By (15), we have

$$\lceil \log_2 \epsilon^{-1} \rceil = |A| + 2 = \ell + 2.$$

Thus our polynomials have degree $\Delta \le (\ell + 2)d$, and

$$\delta = \Delta - |A| = (|A| + 2)d - |A| = (d - 1)\ell + 2d\,.$$

Recall that the given input consists of the polynomials $P_1, P_2, \ldots, P_m$ in (6). Using the polynomials $P_1, P_2, \ldots, P_m$, the algorithm draws s samples, where each sample is an independent collection of Razborov–Smolensky polynomials $R_1, R_2, \ldots, R_{\ell+2}$ constructed using (11). (Recall the R_i's are simply random linear combinations of the P_j's.) Each collection forms one of the approximate summands $f_j = \prod_i (1 + R_i)$ of (12). However, in our algorithm these approximate summands f_j are never constructed in explicit form: in Lokshtanov et al. [9] they are constructed explicitly, which leads to a worse running time.

Rather, we observe that we can access a part $f_j|_A^{Z\to B}$ for $Z \in \binom{B}{\downarrow\delta}$ by making the substitution $Z \to B$ directly into the Razborov–Smolensky polynomials $R_1, R_2, \ldots, R_{\ell+2}$ that define f_j, without constructing f_j itself. In particular, after the substitution, the polynomials have variables $x_A = (x_j : j \in A)$. In notation, we construct by the substitution $Z \to B$ the polynomials

$$Q_1(x_A) = R_1|_A^{Z\to B}(x_A)\,, \quad Q_2(x_A) = R_2|_A^{Z\to B}(x_A)\,, \quad \ldots, \quad Q_{\ell+2}(x_A) = R_{\ell+2}|_A^{Z\to B}(x_A)\,.$$

Since the R_i's are just linear combinations of polynomials of degree at most d, these polynomials also have degree at most d, and are over $\ell = |A|$ variables. Thus, computing the sum $I_{f|_A^{Z\to B}}$ of the part $f|_A^{Z\to B}$ is exactly the task of summing over $x_A \in \{0,1\}^{|A|}$ the polynomial

$$f|_A^{Z\to B}(x_A) = (1 + Q_1(x_A))(1 + Q_2(x_A)) \cdots (1 + Q_{\ell+2}(x_A)) \,.$$

Recalling (8), this is exactly the task of determining the parity of the number of solutions $x_A \in \{0,1\}^{|A|}$ to the system of polynomial equations

$$Q_1(x_A) = 0 \,, \quad Q_2(x_A) = 0 \,, \quad \ldots, \quad Q_{\ell+2}(x_A) = 0 \,.$$

This completes our randomized reduction from parity-counting to itself: we have reduced parity counting for a system of degree-d polynomials in n variables to $\binom{|B|}{\downarrow\delta} = \binom{n-\ell}{\downarrow(d-1)\ell+2d}$ calls to parity counting a system of degree-d polynomials in ℓ variables.

To compare, in Lokshtanov et al. [9], such a self-reduction seems not to be possible when one uses (9) instead of (7). A full $O^*(2^{|A|})$ time summation was used in their paper.

3.7 Running time analysis

Let us now analyze the running time as a function of the number of variables n and the reduction parameter ℓ with $1 \le \ell \le n$. Let us write $T(n,m)$ for an upper bound for the worst-case running time when the input consists of at most m polynomials of degree at most d in at most n variables. Similarly, let us write $S(n,m)$ for an upper bound for the worst-case space complexity.

Let us now recall the structure of the self-reduction, then analyze its recursive application. The reduction first builds the $s = 48n + 1$ approximate summands (via the constituent polynomials $R_1, R_2, \ldots, R_{\ell+2}$) and then works to complete $2^{|B|} = 2^{n-\ell}$ scoreboards, each recording the sum (over the integers to enable majority-voting) of summation of s parts. The summation of parts proceeds across the scoreboards, one entire approximate summand at a time using steps (i), (ii), and (iii). In step (i), we recursively perform summations for $\binom{|B|}{\downarrow\delta} = \binom{n-\ell}{\downarrow(d-1)\ell+2d}$ parts, each via the constituent polynomials $Q_1, Q_2, \ldots, Q_{\ell+2}$ of degree at most d over ℓ variables. In step (ii), we run the fast zeta transform over $\binom{B}{\downarrow\delta}$ to recover the monomials of the summand for all $A \cup Y$ with $Y \in \binom{B}{\downarrow\delta}$. In step (iii), we run the fast zeta transform over 2^B to recover all sums of parts for one approximate summand. This procedure is repeated for each of the approximate summands, updating the scoreboard as we go. Once the scoreboards are complete, the algorithm takes the majority vote in each scoreboard, and returns the parity of the majority votes.

The space complexity of the reduction can be upper-bounded via the recursive scoreboards and the representation of the polynomials in the monomial basis, with

$$S(n,m) \le S(\ell, \ell+2) + O\big(2^{n-\ell} \log s + m\tbinom{n}{\downarrow d}\big) \,. \tag{20}$$

Indeed, the zeta transforms at each level of recursion require only space $O(2^{n-\ell})$.

The time complexity of the reduction can be upper-bounded via the brute-force base case $T(n,m) \le O(2^n nm\binom{n}{\downarrow d})$ and the recurrence

$$T(n,m) \le s\tbinom{n-\ell}{\downarrow(d-1)\ell+2d} T(\ell,\ell+2) + O\big(s\big(\tbinom{n-\ell}{\downarrow(d-1)\ell+2d}\big)\big(n+(\ell+2)m\tbinom{n}{\downarrow d}\big)+2^{n-\ell}(n-\ell)\big)\big) \,. \tag{21}$$

The first term accounts for the parity self-reduction; the second term accounts for the fast zeta transforms.

Let us now assume that $d = 2, 3, \ldots$ is a fixed constant. We will run the reduction (21) recursively for $D = D(d)$ levels, and then switch to the brute-force base case. The parameter ℓ at each level is set by means of a constant $\lambda = \lambda(d)$ with $0 < \lambda < \frac{1}{2d-1}$ so that $\ell = \lfloor \lambda n \rfloor$ where n is the number of variables in the input to the level. Let

$$H(\rho) = -\rho \log_2 \rho - (1-\rho)\log_2(1-\rho)$$

be the binary entropy function and recall that we have $\binom{k}{\lfloor u \rfloor} \leq 2^{kH(u/k)}$ for all $1 \leq u \leq \frac{k}{2}$.

Since $\ell \leq \frac{n}{2d-1}$, the sums of binomial coefficients in (21) can be upper-bounded as follows:

$$\tbinom{n-\ell}{\downarrow(d-1)\ell+2d} \leq n^{2d+1}\tbinom{n-\ell}{(d-1)\ell} \leq n^{2d+2}2^{n(1-\lambda)H\left(\frac{(d-1)\lambda}{1-\lambda}\right)}.$$

Assuming that we run the recursion for $D = D(d)$ levels and then use brute force, we observe that there exists a constant $C = C(d) > 0$ such that we have

$$T(n,m) = O(mn^C(1 + 2^{\tau(1)n}))\,,$$

where $\tau(\lambda^k)$ is a parameter defined for $k = 0, 1, \ldots, D-1$ by

$$\tau(\lambda^k) = \lambda^k \max\bigl((1-\lambda)H\bigl(\tfrac{(d-1)\lambda}{1-\lambda}\bigr) + \tau(\lambda^{k+1}),\, 1-\lambda\bigr) \tag{22}$$

and

$$\tau(\lambda^D) = \lambda^D\,. \tag{23}$$

Recalling that $1 + \lambda + \lambda^2 + \ldots = \frac{1}{1-\lambda}$, and choosing a large enough D, we have that

$$\tau(1) \leq 1-\lambda \quad \text{whenever} \quad H\bigl(\tfrac{(d-1)\lambda}{1-\lambda}\bigr) < 1-\lambda\,.$$

Thus, for any $d \geq 2$ we can select $\lambda = 1/(2.7d)$ to obtain the running time $O^*\bigl(2^{(1-1/(2.7d))n}\bigr)$. For $d = 2$, we can select $\lambda = 0.196774680497$ to obtain the running time $O^*\bigl(2^{0.803225n}\bigr)$.

References

1. Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic Boolean systems. *J. Complexity*, 29(1):53–75, 2013. doi:10.1016/j.jco.2012.07.001.
2. Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto, Jesper Nederlof, and Pekka Parviainen. Fast zeta transforms for lattices with few irreducibles. *ACM Trans. Algorithms*, 12(1):Art. 4, 19, 2016.
3. Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal.* PhD thesis, Department of Mathematics, University of Innsbruck, 1965.
4. David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015. doi:10.1007/978-3-319-16721-3.
5. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of problems in games, graphs and algebraic equations. *Discrete Applied Mathematics*, 1(1-2):15–30, 1979. doi:10.1016/0166-218X(79)90012-X.
6. Antoine Joux and Vanessa Vitse. A crossbred algorithm for solving Boolean polynomial systems. Cryptology ePrint Archive, Report 2017/372, 2017. URL: https://eprint.iacr.org/2017/372.
7. Petteri Kaski, Jukka Kohonen, and Thomas Westerbäck. Fast Möbius inversion in semimodular lattices and ER-labelable posets. *Electron. J. Combin.*, 23(3):Paper 3.26, 13, 2016.

8 Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 1999. doi:10.1007/3-540-48910-X_15.

9 Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating Brute Force for Systems of Polynomial Equations over Finite Fields. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2190–2202. SIAM, 2017. doi:10.1137/1.9781611974782.143.

10 Yuri V. Matiyasevich. *Hilbert's Tenth Problem*. Foundations of Computing Series. MIT Press, Cambridge, MA, 1993.

11 Ernst W. Mayr. Some complexity results for polynomial ideals. *J. Complexity*, 13(3):303–325, 1997. doi:10.1006/jcom.1997.0447.

12 Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, Cambridge, second edition, 2017.

13 Robin A. Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 245–252. ACM, 2011. doi:10.1145/1993636.1993670.

14 Ruben Niederhagen, Kai-Chun Ning, and Bo-Yin Yang. Implementing Joux-Vitse's Crossbred Algorithm for Solving MQ Systems over GF(2) on GPUs. Cryptology ePrint Archive, Report 2017/1181, 2017. URL: https://eprint.iacr.org/2017/1181.

15 Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996. doi:10.1007/3-540-68339-9_4.

16 A. A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

17 Roman Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

18 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

19 F. Yates. *The Design and Analysis of Factorial Experiments*. Imperial Bureau of Soil Science, Harpenden, 1937.